Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Белоглазов М.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук, доцент
_____ Кислун О.А.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерна інженерія"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Белоглазову Максиму Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN*

2. Керівник роботи *Кислун Олег Андрійович, канд. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 35-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Белоглазов М.В. Дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Метою розробки є дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Об'єктом дослідження є процес проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Предметом дослідження є методи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

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 10/11.

Програму розроблено в середовищі Delphi 10.4 Sydney.

Ключові слова: комп'ютерна інженерія, NGN

ABSTRACT

Belohlazov M.V. Research and software implementation of the architecture design system of corporate networks built on wireless NGN technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the system of designing the architecture of corporate networks built on wireless NGN technology.

The purpose of the development is the research and software implementation of the architecture design system of corporate networks built on wireless NGN technology.

The object of research is the process of designing the architecture of corporate networks built on wireless NGN technology.

The subject of research is the methods of designing the architecture of corporate networks built on wireless NGN technology.

Research methods are based on telecom theory methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system for designing the architecture of corporate networks built on wireless NGN technology.

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.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Delphi 10.4 Sydney environment.

Keywords: computer engineering, NGN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БНМ	–	балансування навантаження мережі, Network Load Balancing
ЕЦП	–	електронний цифровий підпис
ІВК	_	інфраструктура відкритих ключів
ІТ	_	інформаційні технології
ЛОМ	_	локальна обчислювальна мережа
ПЗ	_	програмне забезпечення
СКЗІ	_	система комплексного захисту інформації
УК	_	управляючі компоненти
УЦ	_	удостовірюючий центр
IGMP	_	Internet Group Management Protocol
NLB	_	Network Load Balancing, балансування мережного навантаження
PKI	_	Public Key Infrastructure
VPN	_	віртуальна приватна мережа

ВСТУП

Актуальність теми. При всій ефективності й швидкодії бездротових технологій вони не можуть обійтися без провідної мережі NGN. Якщо при її побудові будуть допущені прорахунки, то стрімке зростання трафіку Wi-Fi може привести до виникнення вузьких місць у кабельній системі. Поява нових стандартів на бездротові мережі NGN привносить додаткові складності.

Донедавна Wi-Fi у корпоративних будинках уважався додатковим преміальним сервісом – він пропонувався відвідувачам і іноді використовувався співробітниками. Однак сьогодні наявність бездротової мережі NGN стало обов'язковою вимогою, пропонованою до офісних комплексів, аеропортів, торгових центрів, спортивних арен і інших місць масового скупчення людей.

Співробітники компаній, клієнти й орендарі хочуть мати доступ до необхідних даних і послуг незалежно від того, де вони перебувають і який тип мобільного пристрою використовують. Зростаюча залежність від бездротового зв'язку привела до швидкого поширення підходу «Принеси свій власний пристрій» (Bring Your Own Device, BYOD). У міру того як бездротові пристрої й додатки стали оперувати все більшими обсягами даних і мати потребу у все більшій пропускній здатності й більше швидкому відгуку, у багатьох мережах стали виникати проблеми із продуктивністю.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

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

– Огляд існуючих систем проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– Дослідження системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– Програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Об'єктом дослідження є процес проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Предметом дослідження є методи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– Розроблено вітчизняний продукт проектування архітектури корпоративних мереж побудованих на технології бездротової NGN, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Зіштовхуючись із такими тенденціями, як BYOD, IT-керівники змушені розробляти нові стратегії для інтеграції бездротових рішень із ядром мережі NGN. При цьому найчастіше виявляється, що принципи розміщення бездротових точок доступу (ТД) і розгортання кабельної інфраструктури необхідно переглядати. Хоча за рівнем пропускної здатності виділені провідні з'єднання завжди будуть перевершувати поділювані бездротові мережі NGN, значні успіхи, досягнуті в області підвищення швидкості передачі даних і продуктивності радіотехнологій, відкривають нові можливості при проектуванні мережі NGN.

IT-відділи компаній зіштовхуються сьогодні з дуже непростими завданнями. При всій ефективності й швидкодії бездротових технологій вони не можуть обійтися без провідної мережі NGN, кабельна інфраструктура якої була б адекватна завданню підтримки WLAN. Якщо при її побудові будуть допущені прорахунки, то стрімке зростання трафіку Wi-Fi може привести до виникнення вузьких місць у кабельній системі. Поява нових стандартів на бездротові мережі NGN привносить додаткові складності, до яких IT-відділи повинні бути готові

Розвиток корпоративних систем бездротового доступу

Організація Wi-Fi Alliance визначає Wi-Fi як будь-які «продукти для бездротової локальної мережі NGN (WLAN), які засновані на стандартах IEEE 802.11». У цей час найбільший інтерес представляють два стандарти: 802.11n і недавно ратифікований 802.11ac.

IEEE 802.11n. Ратифікований в 2007 році (відповідні продукти почали виводитися на ринок в 2009 році) стандарт IEEE 802.11n припускає використання каналів шириною 20 або 40 Мгц у діапазоні 2,4 або 5 Ггц і забезпечує швидкість до 600 Мбіт/с. Він став першим стандартом WLAN, у якому підтримується

технологія MIMO, що дозволяє передавати декілька радіопотоків на різні антени, що дозволяє знизити негативний вплив перешкод, підвищити швидкість передачі й надійність.

IEEE 802.11ac. У червні 2013 року IEEE ратифікував стандарт 802.11ac, новітній на даний момент варіант технології Wi-Fi. Він забезпечує кращі характеристики й пропускну здатність: агрегована швидкість передачі даних може доходити до 6,9 Гбіт/с, що більш ніж у десять разів більше, ніж швидкість систем стандарту 802.11n. Удосконалена технологія припускає використання частот у діапазоні 5 Ггц і більше складних методів модуляції для надання більшого числа незалежних каналів.

Нова технологія стрімко ввірвалася на ринок. Уже 19 червня 2013 року – у той самий день, коли було оголошено про завершення роботи над 802.11ac, – Wi-Fi Alliance підтвердив сумісність із цим стандартом 19 продуктів: маршрутизаторів, точок доступу, мікрочипів і смартфонів. Як правило, наша сертифікаційна програма служить одним з факторів, що сприяють широкому поширенню технології на ринку, але з настільки стрімким прийняттям нового стандарту раніше не зіштовхувалися.

1.2 Область застосування

Відповідно до дослідження, проведеному в 2015 році виданням Information Week, більше 45% IT-керівників планують впроваджувати системи 802.11ac у своїх корпоративних мережах. А недавній прогноз ABI Research пророкує, що в самий найближчий час 40% смартфонів будуть підтримувати нову версію Wi-Fi. На думку експертів Wi-Fi Alliance, в 2017 році пристрою 802.11ac становлять більшість ринків продуктів Wi-Fi.

Захоплене прийняття 802.11ac і його високих швидкостей чревате проблемами для розподільних мереж, які у свій час проектувалися для підключення точок доступу 802.11n. У самої низькошвидкістної конфігурації –

1×1 MIMO – 802.11ac підтримує швидкість 866 Мбіт/с, що на 44% більше, ніж у випадку самого швидкісного варіанта 802.11n. А коли використовується схема 8×8 MIMO, нова технологія здатна забезпечити швидкість 6,9 Гбіт/с. Крім того, доступність нових каналів тепер дозволяє розміщати точки доступу з більшою щільністю й завдяки цьому надавати послуги Wi-Fi з розширеним функціоналом.

Отже, устаткуванню 802.11ac може не вистачити пропускної здатності провідних каналів 1000Base. Навіть орієнтовані на роботу в найменш швидкісному режимі пристрою 802.11ac можуть бути оснащені двома портами 1000Base з функцією агрегування каналів. Очікується, що в майбутньому відповідні продукти стануть оснащуватися портами 10GBase або двома або навіть чотирма портами 1000Base з метою підтримки мультигігабітного підключення.

Новий рівень щільності й швидкості систем Wi-Fi міняє правила гри відносно проектування розподільної кабельної інфраструктури. IT-фахівці повинні звернути особливу увагу на те, як і де планується встановлювати нові точки доступу, і забезпечити побудова такої кабельної інфраструктури, що була б у стані передавати генеруємий трафік.

Пристрої IEEE 802.11ac стануть, імовірно, першим додатком, запити якого перевищать продуктивність широко використовуваних сьогодні кабельних систем Категорії 6. Першим, але точно не останнім. З огляду на швидкість, з якою розвиваються бездротові технології, проектована сьогодні кабельна інфраструктура повинна бути досить гнучкої й мати запас ємності, щоб підтримувати нові системи протягом довгого часу.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

У наш час вибір структури мережі NGN здійснюється емпіричним шляхом. Як правило, це роблять фахівці, ґрунтуючись на власному досвіді. Таким чином, процес вибору структури обчислювальної мережі NGN залежить від конкретної людини (компанії). На початковому етапі, коли число обчислювальних мереж було невелике, такий метод вибору структури мережі NGN був прийнятний. Стрімкий прогрес обчислювальних і телекомунікаційних засобів привів до різкого збільшення попиту на корпоративні мережі NGN. Відсутність загальнодоступних методик і алгоритмів вибору структур мереж NGN помітно гальмує їхній розвиток. Потрібен системний підхід до даної проблеми. Наявність ефективних загальнодоступних методик дозволило б у значній мірі скоротити часові й вартісні витрати на проектування мережі NGN, зробити доступним рішення цього складного завдання більше широкому колу фахівців, що безсумнівно приведе до якісного стрибка в розвитку обчислювальних мереж NGN.

Ціль методології створення мереж NGN полягає в організації процесу побудови мережі NGN і забезпеченні керування цим процесом для того, щоб гарантувати виконання вимог, як до самої мережі NGN, так і до характеристик процесу розробки. Основні завдання, рішення яких повинна забезпечувати методологія побудови мереж NGN:

– забезпечувати створення мереж NGN, що відповідають пропонованим до них вимогам по автоматизації ділових процесів і які відповідають цілям і завданням організації;

– гарантувати створення системи із заданою якістю в заданий термін і в рамках бюджету;

– підтримувати зручну дисципліну супроводу, модифікації й нарощування системи, щоб інформаційна система (ІС) могла відповідати швидко, що змінюються вимогам, роботи компанії, забезпечити використання заділу в області інформаційних технологій;

– забезпечувати створення ІС, що відповідають вимогам відкритості, переносимості й масштабованості..

Методологія повинна забезпечувати зниження складності процесу створення мережі NGN за рахунок повного й точного опису цього процесу й застосування сучасних методів і технологій створення мережі NGN на всьому життєвому циклі мережі NGN – від задуму до реалізації.

Існуючі підходи й методології проектування

У цей час не існує загальноприйнятих підходів і методологій проектування мереж NGN. Для того, щоб було простіше розібратися в існуючих підходах і методологіях проектування мереж NGN пропонується наступна класифікація. Методології побудови комп'ютерних мереж NGN умовно можна розбити на два класи. Перший клас містить у собі методології, основою яких є використання набору стандартних рішень при побудові мереж NGN (під стандартними рішеннями маються на увазі рішення, пропоновані відомими компаніями – Cisco, HP, і т.д.). Даний клас методологій характеризується відносно низьким рівнем витрат на проектування, однак отримана мережа може не повною мірою відповідати пропонованим вимогам. Мережі NGN, побудовані з використанням методологій другого класу, містять крім стандартних рішень ще й унікальні розробки, які дозволяють максимально адаптувати мережу до структури бізнес-процесів підприємства. Нижче наведено два описи методологій, що належать кожному із класів.

При такому підході мережа NGN представляється у вигляді багатошарової піраміди. У підставі піраміди, що представляє корпоративну мережу NGN,

лежить шар комп'ютерів – центрів зберігання й обробки інформації, і транспортна підсистема, що забезпечує передачу інформаційних пакетів між комп'ютерами. Над транспортною системою працює шар мережних операційних систем, що організує роботу додатків у комп'ютерах і надає через транспортну систему ресурси свого комп'ютера в загальне користування. Над операційною системою працюють різні додатки, цей клас системних додатків звичайно виділяють в окремий шар мережі NGN. На наступному рівні працюють системні сервіси, які, користуючись СУБД, як інструментом для пошуку потрібної інформації серед мільйонів і мільярдів байт, збережених на дисках, надають кінцевим користувачам цю інформацію в зручній для ухвалення рішення формі. До цих сервісів відноситься служба WWW, система електронної пошти й багато які інші.

Верхній рівень мережі NGN представляють спеціальні програмні системи, які виконують завдання, специфічні для даного підприємства. Прикладами таких систем можуть служити системи автоматизації банку, організації бухгалтерського обліку, і т.п. Кінцева мета мережі NGN втілена в прикладних програмах верхнього рівня [1]. Хоча шари цієї піраміди зв'язані й впливають один на одного, звичайно кожний шар проектується досить автономно фахівцями й фірмами відповідного профілю.

Основна особливість такого підходу – використання набору стандартних готових рішень як будівельних блоків для створення мережі NGN. Подання мережі NGN у вигляді багатошарової піраміди не дає чіткої відповіді як будувати мережа NGN. Даний підхід має право на існування й найбільш ефективний при побудові нескладних мереж NGN.

Недоліки й переваги даної методології

Недоліки:

– Використання готових рішень, які можуть погано сполучатися один з одним.

– При проектуванні недостатньо враховується інформаційна структура підприємства.

– Готова мережа NGN не завжди повністю відповідає пропонованим вимогам.

Переваги:

– Скорочення часу розробки, витрат на проектування й, як наслідок, зниження вартості мережі NGN.

– Мережі NGN, побудовані з використанням такого підходу, можуть задовольнити в тому або іншому ступені більшість замовників.

Інформаційний інжиніринг і реінжиніринг бізнес проектів

Пропоновані в цих принципово нових підходах методи дозволили описувати, аналізувати й проектувати структуру й діяльність корпорацій подібно технічним системам. Кожний із цих підходів породив свій клас методологій, що володіють загальними характеристиками. Методологія будується на основі ітераційної спіральної моделі життєвого циклу мережі NGN. Принциповою особливістю методології є те, що, охоплюючи всі етапи життєвого циклу мережі NGN, вона робить основний упор на підтримку початкових етапів створення мереж NGN (формування вимог до мережі NGN, що точно відповідають цілям і завданням організації). Відповідно до підходу інформаційного інжинірингу, якому можна визначити як застосування взаємозалежного набору формальних технологій (моделей) для планування, аналізу, проектування й створення ІС на рівні корпорацій або окремих її частин [2], процес створення мережі NGN будується як процес побудови й розвитку моделей. Таким чином, фундамент пропонованої методології становлять ітераційна спіральна модель життєвого циклу ІС і комплекс систем, що розвиваються, погоджених моделей.

Ітераційна спіральна модель життєвого циклу ІС

Методологія описує процес створення й супроводу ІС у вигляді життєвого циклу (ЖЦ) мережі NGN, представляючи його у вигляді послідовності стадій, кожна з яких розбита на етапи, і виконуваних на них процесів. Для кожного етапу

визначаються послідовність виконуваних робіт, одержувані результати й т.д. Такий формальний опис ЖЦ мережі NGN дозволяє спланувати й організувати процес колективної розробки й забезпечити керування цим процесом. Він включає стадії аналізу, проектування, розробки, тестування й інтеграції, впровадження, супроводи й розвитку мережі NGN.

Процес створення мережі NGN – це процес побудови й послідовного перетворення погоджених моделей на всіх етапах ЖЦ. За допомогою CASE-засобів моделі створюються, перетворяться й контролюються. Основними результатами на кожному етапі ЖЦ є моделі обумовлених на даному етапі об'єктів (організації, вимог до мережі NGN і додаткам і т.д.).

Комплекс систем, що розвиваються, погоджених моделей

Методологія визначає процес створення мереж NGN як процес побудови й послідовного розвитку систем погоджених моделей [3].

Початком процесу створення ІС є моделі бізнес-процесів організації. Із цих моделей може бути отримана більшість найважливіших вимог до ІС. Це фундаментальне положення методології дозволяє абсолютно об'єктивно підійти до виробітку вимог і проектуванню ІС. Створюється система моделей опису вимог до мережі NGN, що потім перетвориться в систему моделей, що описують проект мережі NGN. Формуються моделі архітектури ІС, вимог до програмного забезпечення (ПЗ) і інформаційному забезпеченню (ІЗ); формується архітектура ПЗ й ІЗ, виділяються корпоративні БД і окремі додатки, формуються моделі вимог до додатків і проводиться їхня розробка, тестування й інтеграція.

Недоліки й переваги даної методології

Недоліки:

– Мережа NGN, спроектована з використанням даної методології, буде дорожче ніж мережа NGN, побудована із застосуванням інших методологій, оскільки процес створення більше тривалий і трудомісткий.

Переваги:

– Дана методологія дає гарантію, що побудована мережа NGN буде повністю відповідати пропонованим до неї вимогам.

– Мережа NGN є відбиттям бізнес-процесів підприємства, що дозволяє використовувати неї з максимальною ефективністю.

Cisco Packet Tracer

Cisco Packet Tracer – це багатофункціональна програма моделювання мереж, що дозволяє студентам експериментувати з поводженням мережі й оцінювати можливі сценарії. Будучи невід'ємною частиною комплексного середовища навчання Мережної академії, Packet Tracer надає функції моделювання, візуалізації, авторської розробки, атестації й співробітництва, а також полегшує викладання й вивчення складних технологічних принципів.

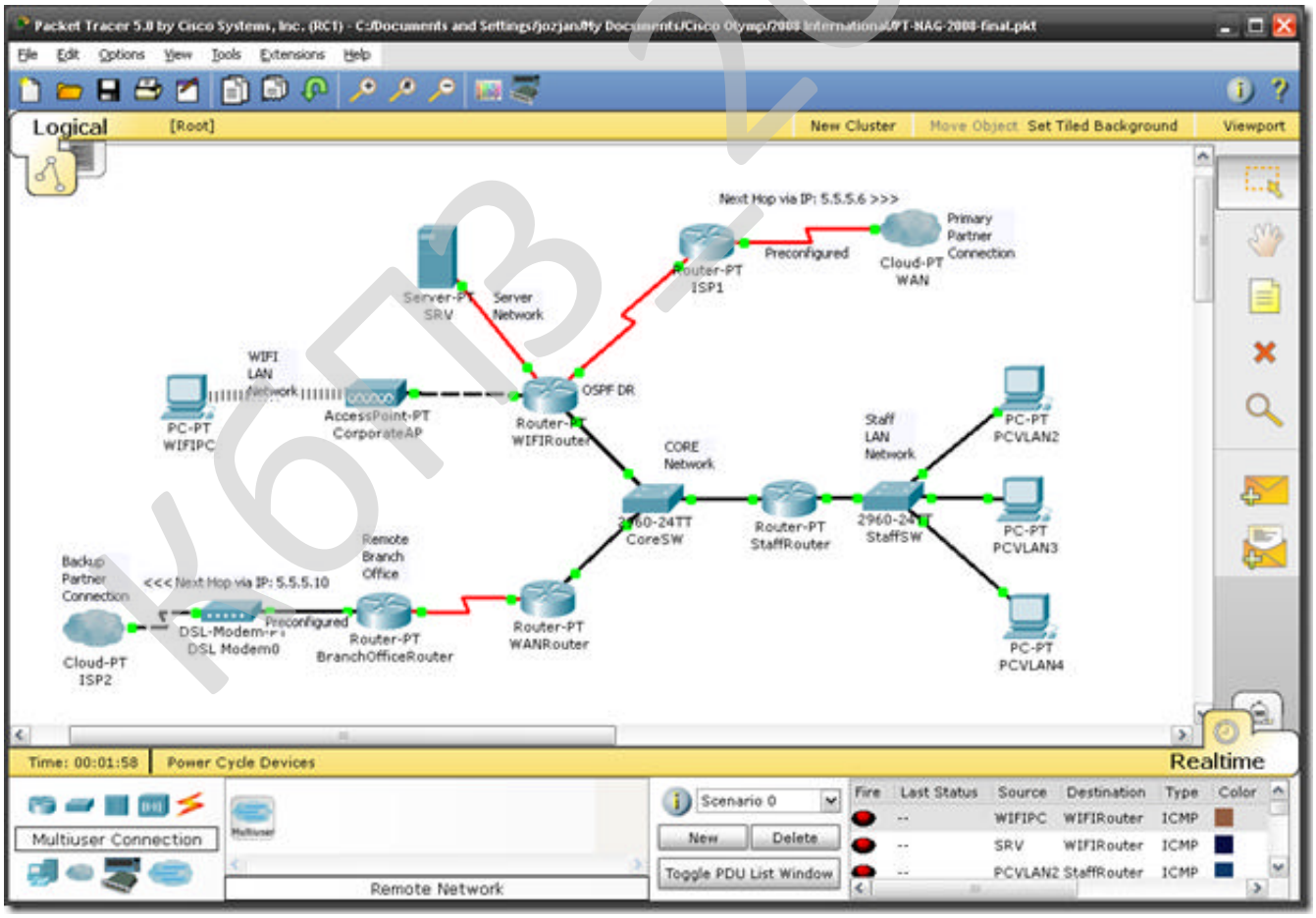

Рисунок 2.1 – Інтерфейс користувача Cisco Packet Tracer

Packet Tracer доповнює фізичне встаткування класу, дозволяючи студентам створювати мережі із практично необмеженою кількістю пристроїв, підтримуючи нагромадження практичного досвіду, тягу до відкриттів і розвиток навичок усунення неполадок. Навчальне середовище на основі імітаційних моделей допомагають студентам розвивати навички XXI століття, такі як прийняття рішень, творче й критичне мислення й рішення завдань. Packet Tracer доповнює програми Мережних академій, що дозволяє викладачам легко описати й показати складні технічні принципи й проекти мережних систем.

ПЗ Packet Tracer безкоштовно надається для інструкторів, студентів, випускників і адміністраторів Мережної академії, зареєстрованих як користувачів NetSpace.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

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

Delphi 10.4 Sydney

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

виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично

масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

Розроблювачі VCL застосунків тепер можуть використовувати 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 мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 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

Стандарти на підключення бездротових точок доступу

Розвиток технологій бездротового зв'язку стало спонукальним фактором для повторного аналізу, а в деяких випадках і для перегляду існуючих стандартів, що визначають топологію й інші аспекти кабельних систем, використовуваних для підключення точок доступу. Розглянемо деякі із цих стандартів.

Багато стандартів, що стосуються бездротових мереж, концентруються на питаннях підключення точок доступу до загальної мережі NGN об'єкта. Сутужніше всього вибрати розмір стільники таким чином, щоб побудована мережа WLAN забезпечувала ефективну підтримку додатків IEEE 802.11n і 802.11ac, а кабельна розподільна мережа гарантувала достатню продуктивність сьогодні й у майбутньому.

В 2004 році міжнародні організації ISO і IEC представили технічний звіт TR-24704, що сфальцьований на питаннях «універсальної кабельної інфраструктури для підключення бездротових точок доступу усередині приміщення». Складений у період становлення корпоративних систем Wi-Fi, він містить чудове передбачення росту можливостей сучасних бездротових технологій.

Установлюючи стандарти кабельної інфраструктури для бездротового доступу, звіт TR-24704 пропонує оптимальну схему розміщення точок бездротового доступу. Як модель мережі NGN пропонується масив із прилягаючий друг до друга шестикутних стільників (осередків). Крім цього, передбачається використання всеспрямованих антен, із круговим випромінюванням у кожної соте. При правильній розстановці досягається мінімальне й одночасно оптимальне перекриття сусідніх стільників.

Для досягнення більше високої ємності зона покриття кожного стільника обмежений радіусом 12 м. TR-24704 рекомендує розміщати телекомунікаційні розетки якнайближче до центра стільники. Це забезпечує максимальну гнучкість при установці окремих точок доступу й найкраще покриття.

Незабаром після появи документа TR-24704, Асоціація TIA представила свої рекомендації з реалізації кабельної інфраструктури для підключення бездротових точок доступу. Документ TIA TSB-162 припускає створення сітки із квадратних осередків зі стороною 18 м. Такий підхід більше відповідає типовому плануванню в Північній Америці, він спрощує проектування й інсталяцію. Точки доступу розміщаються при можливій довжині шнура підключення до 13 м. Передбачаючи прийняття стандарту IEEE 802.11ac, укладачі нової редакції цього документа рекомендували використовувати кабельну проводку Категорії 6A, щоб забезпечити достатню пропускну здатність і підтримку систем Power-over-Ethernet (PoE) зі збільшеною потужністю.

До кінця 2012 року Асоціація TIA прийняла ще один стандарт розраховуючи на високу щільність розміщення точок доступу. Документ TIA-4966 призначений для забезпечення бездротового покриття в освітніх установах, але викладені в ньому рекомендації корисні для будь-яких великих приміщень або будинків з високою концентрацією бездротових клієнтів.

При оснащенні великих відкритих приміщень рекомендується при виборі щільності розміщення точок доступу враховувати передбачуване число користувачів. Плануючи покриття для приміщень із декількома секціонованими зонами, необхідно виходити з їхньої площі. У типовому офісному будинку рекомендується одна точка доступу на кожні 230 м^2. На менш «дружніх» для радіосигналу об'єктах, наприклад у гуртожитках, пропонується встановлювати по одній точці доступу на кожні 150 м^2.

Планування бездротової мережі NGN доступу

Незважаючи на різницю в рекомендаціях, які пропонуються в зазначені вище стандартах, кожний з них може допомогти IT-фахівцям у проектуванні

ефективної WLAN з високою ємністю. Але, звичайно, для цього потрібно й більше глибоке розуміння специфіки даного процесу.

Щоб повною мірою використовувати переваги таких технологій, як 802.11ac, необхідно враховувати електромагнітну обстановку на об'єкті, рівень перешкод і їхніх джерел, майбутню потребу в ємності мережі NGN, вимоги до кабельної проводки й забезпечення електроживлення. У рамках цієї статті ми не зможемо дати вичерпний аналіз розглянутої тут теми, але постараємося представити основні рекомендації з побудови кабельної інфраструктури й викласти загальні ідеї планування й реалізації ефективної мережі NGN точок доступу.

В ідеалі розробка кабельної мережі NGN і аналіз радіопокриття повинні здійснюватися в комплексі – це дозволить забезпечити максимальну ємність і гнучкість для задоволення поточних і майбутніх потреб кінцевих користувачів. На практиці структури, описані в документах TIA TSB-162-a й ISO/IEC TR 24704, мають явні переваги, особливо для застосування в нових будинках.

У новому будинку кабельна сітка (pre-cabled grid) для підключення точок доступу й структурована кабельна система для IT- і іншого встаткування можуть розроблятися й інсталюватися в те саме час. Після того як кабельна сітка змонтована, бездротова інфраструктура на об'єкті може бути встановлена в будь-який час, причому незручності для співробітників будуть мінімальними. При визначенні місця для установки точок доступу рекомендується проводити радіочастотне обстеження, що дозволить оптимізувати розташування ТД у кожної конкретної соти (осередку).

Радіочастотне планування

Для нових інсталяцій більшість виробників устаткування WLAN рекомендують проводити радіочастотне обстеження об'єкта. В умовах, коли запити до ємності мінімальні й немає яких-небудь спеціальних вимог, досить лише оцінити умови поширення радіохвиль. В інших випадках може знадобитися більше ретельний аналіз, у тому числі питань, пов'язаних з перешкодами,

продуктивністю й безпекою. Оскільки новий стандарт 802.11ac припускає істотне збільшення числа одиниць мережного встаткування й обсягу трафіку, виконання ретельного аналізу об'єкта стає дуже важливим – необхідно переконатися в тому, що мережа буде повністю відповідати як поточним, так і майбутнім вимогам до бездротового зв'язку.

Використовуючи подібний інструментарій, наприклад Wi-Fi-аналізатор AirMagnet компанії Fluke Networks, можна змоделювати характеристики бездротової мережі NGN на конкретному об'єкті, оптимізувавши розміщення точок доступу. Ця програма дозволяє ввести план об'єкта, указати місця установки точок доступу й побачити картину радіопокриття в порівнянні з реальної. Крім того, вона забезпечує генерацію звітів і може бути інтегрована із засобами керування WLAN, що робить її використання ідеальним інструментом для проектування бездротової мережі NGN.

Ослаблення сигналу може бути пов'язане з рядом факторів, включаючи наявність поглинаючих матеріалів, таких як шафи й устаткування, або структурних перешкод – бетонних стін або сталевих перебирань. Це пояснює різке падіння потужності сигналу на стороні об'єкта, протилежної тої, де встановлена точка доступу.

Гарне покриття забезпечене практично скрізь, де раніше (при наявності тільки однієї точки доступу) сигнал була слабкий. Таке розташування типово для рішень 802.11n, коли точки доступу використовують канали, що не перекриваються. Забезпечуючи непогане радіопокриття, така конфігурація має обмежені можливості для більше інтенсивного використання Wi-Fi.

Перший крок у плануванні високопродуктивної мережі NGN – поділ об'єкта на сітку осередків (стільників) відповідно до рекомендацій TIA TSB-162-a або ISO / IEC TR 24704. Цю сітку можна пристосувати й для задоволення потреби в кабельній проводці з боку інших систем будинку. Такий інтегрований «відкритий» підхід (з певною волею вибору місця розміщення телекомунікаційних розеток) до реалізації СКС має як короткострокові, так і

довгострокові переваги. При використанні сітки із квадратними осередками (згідно TSB-162-A) положення й щільність точок доступу можна міняти відповідно до заповнюваності конкретної області. Наприклад, у зоні з переговірними кімнатами буде потрібно кілька точок доступу, а в лабораторії можна обійтися їхнім мінімальним числом.

Зверніть увагу на те, що фактичне розміщення точок доступу ґрунтується на поточному використанні простору, тоді як грамотно спроектована кабельна проводка повинна забезпечити можливість майбутніх змін. По завершенні формування сітки осередків варто вибрати кабельну проводку, здатну підтримати майбутні потреби.

Попереднє планування всієї проводки може значно спростити інсталяцію, а також наступну модернізацію, технічне обслуговування й ремонт мережі NGN.

Розміщення й підключення точок доступу

Крім моделювання радіочастотного середовища й планування ємності, при проектуванні розміщення й підключення точок доступу необхідно продумати фізичну доступність, організацію розподільної мережі NGN і електроживлення, а також загальну естетику інсталяції.

Фізична доступність. Грамотний вибір місця установки допоможе заощадити на обслуговуванні й модернізації й у цілому знизити експлуатаційні витрати. Дуже часто доступ до встаткування бездротового зв'язку ускладнений. Нерідко ТД розміщають над фальшпотолком або в закритих шафах, що вимагає прокладки кабелів у подпотолочному просторі. Але монтаж на стіну або на стелю все-таки переважніше. Це дозволяє завжди бачити індикатори стану ТД.

Треба ретельно підійти й до вибору місця розміщення телекомунікаційної розетки. Гарна доступність розетки, її розташування поруч із точкою доступу дозволять легко провести тестування лінії зв'язку й швидко відключити точку доступу для обслуговування або ремонту.

Розподільна мережа. У мережі NGN традиційної топології телекомунікаційна розетка, використовувана для підключення ТД, прямо зв'язана

горизонтальним кабелем з комутаційною панеллю, що перебуває в телекомунікаційній кімнаті на тому же поверсі. Альтернативний варіант – зонна архітектура, що може забезпечити легку установку, високу гнучкість і потенційно більше низькі експлуатаційні витрати.

У зонній моделі кабелі прокладаються від апаратної кімнати до конкретних виділених зон у будинку. Фіксована проводка закінчується на рівні точки консолідації (Consolidation Point, CP), звідки кабелями здійснюється підключення телекомунікаційних розеток для ТД. Такий підхід забезпечує максимальну гнучкість при розміщенні першої розетки в кожному осередку, залишаючи необхідні ресурси для підключення додаткових розеток. Це може бути надзвичайно корисно в ході модернізації існуючих мереж: при грамотно обраному місці установки точки консолідації довгі джгути кабелів, що йдуть із телекомунікаційної кімнати, можуть бути зафіксовані в кабельних важкодоступних каналах. При наявності фіксованої проводки інсталятори одержать «волю маневру» при прокладці подовжувальних кабелів для підключення до точки консолідації телекомунікаційних розеток, які можуть використовуватися для обслуговування ТД або іншого встаткування інтелектуального будинку.

Електроживлення. Якщо точки доступу будуть підключатися до звичайної електромережі NGN, необхідно погодити всі дії зі службою головного енергетика, що у випадку високоплотних інсталяцій з використанням пленумного простору зробити досить складно. Із цієї причини більшість точок доступу сконструйовано для одержання електрики по слабкострумової СКС – за технологією PoE. Для забезпечення надійної роботи встаткування (особливо тих точок доступу, які встановлюються в закритих місцях або будуть експлуатуватися в складних кліматичних умовах) рекомендується перевірити канал PoE.

Вимоги до живлення нових точок доступу стандарту 802.11ac можуть виявитися більше твердими. Проектувальникам варто розглянути можливість

прокладки щонайменше двох кабелів Категорії 6А до кожної точки доступу. Крім резервування каналу зв'язку, це допоможе організувати резервне живлення, і точка доступу зможе функціонувати навіть у випадку виходу з ладу одного з комутаторів або інжекторів PoE, до яких вона підключена.

Естетичні міркування. Бажано зробити все так, щоб кабельна проводка для точок доступу, включаючи телекомунікаційні розетки й шнури підключення, була схована або хоча б не впадала в око. Для цього намагаються задіяти елементи структури будинку, але коли неприпустимо торкатися унікальної архітектури й/або дизайн інтер'єра, доводиться шукати вихід з положення. У кожному конкретному випадку деталі розміщення елементів інфраструктури повинні обговорюватися із клієнтами.

Вибір типу кабельної проводки

З погляду топології кабельні канали для підключення точок доступу відносяться до горизонтальної підсистеми – як і звичайна офісна проводка, прокладена від телекомунікаційної кімнати до телекомунікаційних розеток. У більшості сучасних інсталяцій усередині приміщень використовується витопарна проводка Категорії 6. Ця категорія забезпечує необхідну пропускну здатність для розподільної мережі NGN і може підтримувати PoE Plus (IEEE 802.3at) – оновлений стандарт для систем PoE з максимальною потужністю живлення до 25,5 Вт. Для підключення нових, більше швидкісних рішень 802.11ac варто віддавати перевагу Категорії 6А.

Більшість точок доступу оснащені портами Ethernet RJ-45, сумісними з витопарним кабелем. Проте деякі пристрої мають порти для багатомодового оптичного волокна із з'єднувачами LC. Як правило, вони використовуються в точках доступу, призначених для установки поза приміщеннями або коли довжина кабелів повинна перевищувати 100 м. У цьому випадку для забезпечення достатньої пропускної здатності до кожної ТД необхідно прокласти щонайменше дві пари волокон ОМ3 або більше високі класи. При підключенні по волокну для точок доступу прийде організувати локальне електроживлення.

Інсталяція в нових будинках і модернізація існуючих проектів

Очевидно, що при розгортанні бездротової мережі NGN у новому будинку, де ще не встановлені внутрішні перегородки, є набагато більше можливостей, ніж при модернізації старих проектів. У цьому випадку має сенс зосередитися на оптимізації покриття й забезпеченні більшої гнучкості для майбутнього росту й різних перетворень. Бажано керуватися схемою сітки осередків і рекомендаціями з розміщення точок доступу, викладеними в TIA TSB-162-a й ISO/IEC TR 24704. У проекті може передбачатися зонна архітектура або прокладка кабелю в центр кожного осередку, але треба мати на увазі, що зонна схема забезпечує максимальну гнучкість для наступних модернізацій і оптимізацію витрат на внесення змін. Якщо перегородки й меблі вже встановлені, проектувальникам варто провести радіочастотне обстеження для визначення оптимального місця розміщення кожної точки доступу.

При інсталяції мережі NGN у вже експлуатованому будинку потрібні ретельна оцінка об'єкта й індивідуальний підхід до підключення точок доступу. Планування повинне ґрунтуватися на докладній інформації про радіочастотну обстановку й особливості прокладки кабелів у вже використовуваних приміщеннях. Таким чином, настійно рекомендуємо IT-фахівцям заздалегідь виконати радіочастотний аналіз із обліком планованої пропускної здатності. Особливу увага необхідно звернути на ті області, де потрібне спеціальне покриття або передбачається ріст ємності.

Після завершення аналізу починається розробка кабельної стратегії для розміщення телекомунікаційних розеток і точок доступу. При цьому треба передбачити й урахувати наступне:

– доступ до стельового простору для організації кабельних каналів від телекомунікаційних розеток до кожної точки доступу;

– наявність і близькість розподільних кабельних каналів для запланованих підключень точок доступу;

– архітектурні й естетичне рішення для маскування точок доступу й кабельної інфраструктури;

– особливості радіочастотного покриття.

Рекомендована в TIA TSB-162-a або TR 24704 схема осередків є кращою, але не завжди реалізованої, особливо в модернизуємих середовищах. Якщо наявні розподільні кабельні канали заповнені або не підтримують цю схему, буде потрібно прокладка додаткових каналів. У цьому випадку з метою досягнення найкращого покриття й підтримки майбутнього росту варто вибрати зонну архітектуру.

Рекомендації

IT-фахівцям доводиться виконувати свою роботу в умовах підривного росту числа бездротових пристроїв, що серйозно міняє вимоги до провідних і бездротових мереж. Упоратися із цим ростом допоможе підтримка більше високої швидкості бездротового доступу. Новий стандарт IEEE 802.11ac надає гігабітну швидкість бездротовим клієнтам і вимагає мультигігабітних швидкостей для каналів підключення точок доступу до комутатора доступу або контролеру.

Сучасні бездротові точки доступу мають розширені можливості, але й висувають підвищені вимоги до електроживлення. Із прийняттям нового стандарту IEEE 802.3 на технологію PoE до прикінцевих пристроїв по СКС можна буде підводити до 60 Вт. Щоб бути готовим до цих нововведень, фахівці CommScope рекомендують наступне:

– прокласти щонайменше два кабелі Категорії 6A до кожної точки доступу, переважно з використанням зонної архітектури;

– прокласти щонайменше чотири кабелі Категорії 6A розраховуючи на ТД до кожної зонної коробки, щоб забезпечити додаткову ємність для кожної точки доступу або можливість установки додаткових точок доступу із внесенням мінімальних змін у конфігурацію мережі NGN;

– при плануванні прокладки кабелів використовувати описану вище схему осередків, що дозволить легко підключати точки доступу, маючи волю маневру у виборі місця її установки;

– передбачити прокладку багатомодового волокна, коли необхідно забезпечити швидкість вище 10 Гбіт/с або установка здійснюється поза приміщеннями при довжині каналу понад 100 м;

– змішане використання старих і нових бездротових точок доступу обмежити перехідним етапом, оскільки застарілі версії можуть знизити продуктивність мережі NGN;

– провести дослідження об'єкта й оцінку потенційної продуктивності з урахуванням функціональності точок доступу й клієнтів для оптимізації розміщення (і програмування) цих точок і, відповідно, прокладки кабелів.

З моменту появи в 1997 році першого стандарту 802.11 на WLAN ця технологія стрімко розвивалася. Усього за 16 років пікові швидкості передачі даних збільшилися з 2 Мбіт/с до 6,9 Гбіт/с (у стандарті IEEE 802.11ac). За увесь час було розроблено п'ять версій первісного стандарту, а середній проміжок часу між їхнім прийняттям склав усього 42 місяця. І немає ніяких підстав думати, що тенденція збільшення швидкості і ємності WLAN сповільниться після недавнього прийняття IEEE 802.11ac.

Важливо не тільки гарантувати здатність обраних мережних компонентів підтримувати поточні потреби компанії в бездротовому зв'язку – необхідний потенціал для подальшої модернізації в найближчому майбутньому. Серед різних компонентів бездротової мережі NGN кабельна інфраструктура являє собою найбільш складний об'єкт для модернізації.

Щоб не було потрібно міняти кабельну інфраструктуру щораз, коли буде обновлятися бездротова технологія, варто вибрати кабельну систему з довгострокової – переважно на 20 років – гарантією підтримки додатків, що дозволить бути впевненим у тому, що проводка забезпечить належну підтримку додатків, специфікованих відомими організаціями по стандартизації, протягом багатьох років.

3.2 Розробка структурної схеми

Сучасний ринок зв'язку перебуває на такому етапі, коли оператори мають сприятливу можливість обійти всі труднощі конвергенції, властивим мережам минулих років, і перейти прямо до мереж наступного покоління на базі технології, що одержала робочу назву NGN – «New Generation Network».Для того щоб зробити цей прорив і приєднатися до числа високотехнологічних операторів, необхідні нові рішення в області створення й надання високопродуктивних послуг. NGN – технологія побудови мережі – призначена для надання послуг передачі даних і голосових сервісів. Вона знімає цілий ряд обмежень і бар'єрів, що існують зараз, і в цьому полягає її економічна продуктивність.

До нових можливостей відносяться:

– перехід від принципу з'єднання «точка – точка» до принципу «кожний з кожним»;

– використовується універсальний характер обслуговування різних додатків (ми це бачимо на прикладі Інтернету, VPN);

– абстрагування користувачів від технологій реалізації послуг зв'язку (принцип чорного ящика) і безпрецедентна гнучкість одержання необхідного набору, обсягу і якості послуг;

– повна прозорість взаємин між продавцем і покупцем послуг.

В основу технології покладена концепція перебудови суспільства на принципах повнозв’язності, коли всі інформаційні ресурси стають загальнодоступними в будь-якому середовищі, вони можуть бути доставлені незалежно від того, де перебуває людина. Зародок концепції повнозв’язності з'явився в Інтернеті, що дало поштовх величезної популярності мережі. Користувач, входячи в Інтернет, одержує доступ до усього світу.

NGN з погляду споживача представляється хмарою або «чорним ящиком», тому що споживачеві не цікаво знати стеки, протоколи й т.д. Вона дозволяє

надавати такі послуги, як широкополосний доступ до Інтернету (100 Мбіт/сек), пакетну телефонію, VPN, «відео за запитом» і виділені широкополосні канали.

Таким чином, NGN – мережа зв'язку наступного покоління (Next step generation) – гетерогенна мультисервісна мережа, що забезпечує передачу всіх видів медіатрафіку й розподілене надання необмеженого спектра телекомунікаційних послуг з можливістю їхнього додавання, редагування, розподіленої тарифікації. Мережа підтримує передачу трафіку з різними вимогами до якості обслуговування й забезпечує підтримку зазначених вимог.

Пакетні технології обробки дозволяють запропонувати користувачеві в такій мережі прозорі автоматизовані принципи розрахунків за приєднання, розрахунку за вхідний, вихідний трафік, уводити платежі за ініціалізацію, транзит і термінацію трафіку, розраховувати сигнальний трафік, виділяючи складову трафіку, пропущеного від стороннього оператора .

У новій мережі застосовується передова технологія маршрутизації «Riverstone». На відміну від традиційних мереж у структурі NGN утворений додатковий шар – керування комутацією транспортної мережі. Він організується за допомогою програмних комутаторів – «SoftSwitch», які повинні підтримувати трансляцію основних протоколів VoIP у протоколи традиційних мереж. Тому для сполучення пакетних і традиційних телефонних мереж «SoftSwitch» повинен відповідати наступним вимогам:

– працювати із протоколами сигналізацій різної архітектури й взаємодіяти з медіашлюзами, що забезпечують передачу голосової, сигнальної інформації, даних, IP-телефонії й інших видів трафіку;

– підтримувати вся розмаїтість сигналізацій – ОКС-7, DSS1, ВКС і ін., оскільки з погляду телефонної мережі він є транзитним комутатором і пунктом сигналізації ОКС-7;

– підтримувати всі протоколи IP-телефонії (H.323, MGCP, H.248, SIP) і здійснювати їхню конвертацію з одного протоколу в інший, тому що для

пакетних мереж він є пристроєм керування медіашлюзами й контролером сигналізацій.

Таким чином, устаткування програмної комутації в NGN відіграє роль універсального програмно-апаратного комплексу, конвертера сигналізації, що перетворить протоколи сигналізації як у мережі з комутацією каналів: ОКС-7, DSS1, V5, CAS, так і в мережі пакетної комутації – протоколи IP-телефонії: H 323, SIP, MGCP, MEGACO/ H.248.

Програмний комутатор – це програмно-апаратний комплекс, «мозок» нової мережі, призначений для керування обробкою телефонних викликів, що відбуваються в різних мережах, у тому числі в мережах з комутацією пакетів, він акумулює весь інтелект мережі, а інші елементи, розташовані на периферії, позбавлені інтелекту й повністю підконтрольні програмному комутатору, що в цілому сприяє кращій керованості й масштабованості мережі.

Тому застосування програмних комутаторів з погляду побудови мережі є технологічною інвестицією в майбутнє, тому що ОКС-7 підтримує сьогодні роботу в ТфОП, і тільки острівці нових пакетних мереж сигналізуються із застосуванням IP-Протоколів.

Топологія мережі. Якщо представити архітектуру NGN у вигляді набору площин, то внизу виявиться площина абонентського доступу, що базується на трьох середовищах передачі: металевому кабелі, оптоволокні й радіоканалах, далі йде площина комутації – комутації каналів і/або комутації пакетів. У зазначеній площині перебуває й структура мультисервісних вузлів доступу. Над ними розташовуються програмні комутатори («SoftSwitch»), складові площина програмного керування. Вище перебувають площини інтелектуальних послуг і експлуатаційного керування.

Така структура більше реалістична, і рисунок 3.1 відповідає сьогоднішнім реаліям операторів, які працюють в умовах, коли 5-10% абонентів бажають одержати найсучасніші послуги широкополосного доступу, а деяка частина абонентів чекає установки звичайного телефону.

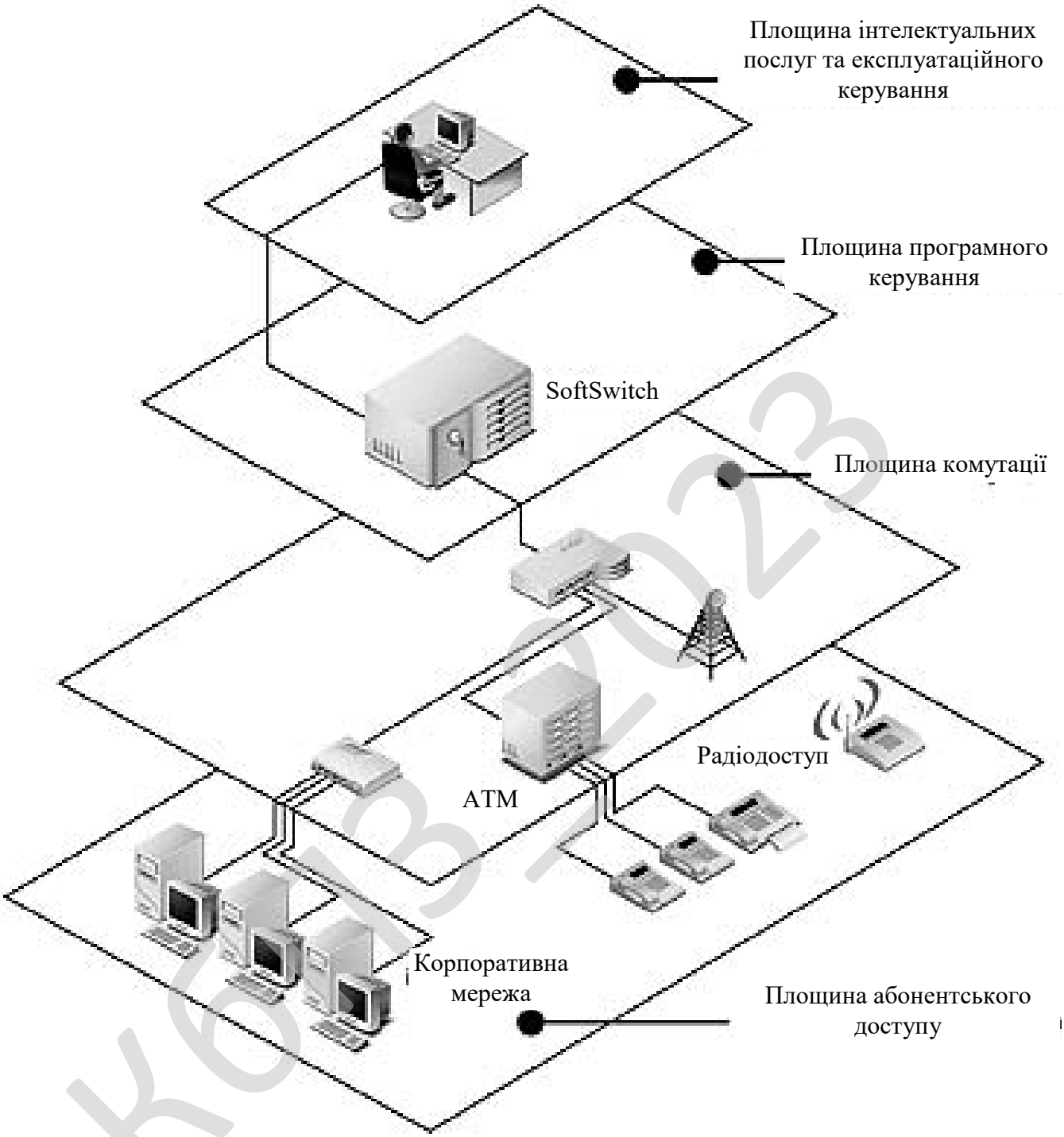

Рисунок 3.1 – Структурна схема системи

Транспортна мережа є опорною мережею в багатошаровій архітектурі телекомунікаційної мережі із шарами, що надбудовуються вільно, послуг, тому вона повинна дуже надійно працювати, інакше всі накладені послуги «зваляться».

Транспортна мережа повинна бути високопродуктивної й будуватися на основі оптико-волоконних ліній зв'язку, що дозволяє забезпечити більшу швидкість обміну (до 100 Мбіт/сек.) і уникнути заторів і колізій при маршрутизації потоків.

Існує й інше бачення топології мережі: так, «Alcatel» визначає наступні шари архітектури NGN: доступу, шлюзів (підтримують стикування з мережами рухливого зв'язку, ТфОП і іншими), транспорту, керування, додатків, експлуатаційного керування.

Базові принципи побудови мультисервісних мереж за технологією NGN

Необхідно відзначити, що немає чітко сформульованих вимог до мережі NGN. Поки сформульовані деякі базові принципи, на які необхідно орієнтуватися при введенні мультисервісних мереж наступного покоління.

Як відзначалося вище, ядром мережі нового покоління є програмний комутатор – мозок мережі, що акумулює весь інтелект мережі, а інші елементи, розташовані на рівнях доступу, шлюзів, транспорту, позбавлені інтелекту й повністю підконтрольні програмному комутатору, що в цілому сприяє кращій керованості й масштабованості мережі.

У випадку переваги на мережі аналогового встаткування, морально й фізично застарілого, можливо модернізувати мережу, використовуючи програмний комутатор у якості розподіленого телефонного концентратора, що дозволить домогтися зниження витрат на будівництво й експлуатацію мережі за рахунок того, що замість двох роздільних мереж (з комутацією каналів і пакетів) буде будуватися одна мережа, функції вузлових (транзитних) АТС буде виконувати програмний комутатор, з'явиться можливість більше економічної організації нових послуг.

Вузол доступу

Устаткування програмної комутації для забезпечення доступу (шлюзи) застосовується в декількох випадках.

Найбільше вигідно застосовувати шлюзи в сполученні з технологіями широкополосного доступу ADSL, які дозволяють передавати великі обсяги інформації по мідних лініях, ефективно використовувати мідну абонентську інфраструктуру. Разом технології NGN і DSL дозволяють довести до споживача по одній мідній парі не один телефонний канал, а високошвидкісний канал передачі даних плюс до декількох десятків телефонних каналів одночасно. Це головна перевага, що ми можемо сьогодні одержати на рівні доступу.

При модернізації мережі необхідно планувати модернізацію застарілого парку встаткування, забезпечивши його можливістю надавати базовий набір послуг, і почати впроваджувати встаткування програмної комутації на транзитному рівні. Групи абонентів, які відразу потраплять на комутатор NGN, одержать базовий набір плюс додаткові послуги: широкополосний доступ, відео за запитом, мультимедійні послуги й т.д. У цьому випадку можна використовувати програмний комутатор у якості розподіленого телефонного концентратора, що дозволить домогтися зниження витрат на будівництво й експлуатацію мережі за рахунок того, що замість двох роздільних мереж (з комутацією каналів і пакетів) буде будуватися одна мережа, а функції вузлових (транзитних) АТС буде виконувати програмний комутатор, з'явиться можливість більше економічної організації нових послуг.

Доцільно не просто збільшувати номерну ємність, а ставити з боку абонентського доступу трохи абонентських концентраторів, що мають можливість включатися по ІКМ-трактах у мережу з комутацією каналів і по IP-Протоколах у мережу з комутацією пакетів. Подібний підхід дозволить захистити на довгий час інвестиції й у той же час акуратно, еволюційно розвивати свої мережі: вирішити насущні проблеми по збільшенню номерної ємності, поступово замінити АТС декадно-крокового типу, зосереджуючи інвестиції переважно в мережі абонентського доступу. Інтерфейс V.5.2, по якому встаткування доступу стикується із ТфОП, що працює за принципом комутації каналів, дозволить тому

ж самому встаткуванню в майбутньому працювати в стику з маршрутизаторами пакетних мереж.

По-друге, необхідно заглядати принаймні на п'ять років уперед і здобувати встаткування мультисервісного абонентського доступу, що зможе працювати в майбутніх мережах з комутацією пакетів.

Поступово нове пакетне транспортне середовище буде розширюватися й заміщати аналогові сегменти, так що в результаті ми перейдемо до NGN на всій мережі – від середовища передачі до середовища послуг – з таким інтерфейсом, що дозволить клієнтові одержувати доступ до будь-яких додатків. Проведені нами попередні розрахунки показують, що витрати на реконструкцію мереж на основі NGN порівнянні з рівнем витрат, які б спричинило послідовне виконання всіх етапів модернізації мережі за класичною схемою: модернізація SDH-мережі, цифровізація мережі (будівництво нових і вдосконалювання наявних вузлів комутації), а потім ще й побудова накладеної мережі передачі даних, яка б знадобилася в цьому випадку для надання нових послуг і доступу до мережі Інтернет.

В основі вибору технології й концепції перспективного розвитку, модернізації або розвитку мережі підприємства зв'язки повинні бути присутнім економічні мотиви. У мультисервісних мережах ці мотиви виражаються в наданні нових послуг і зниженні витрат на їхнє формування завдяки унікальній можливості побудови технологічної інфраструктури з розподіленою комутацією й гнучкою уніфікованою структурою керування.

Концепція мультисервісної мережі досить складна й немає ніяких стандартів на її використання. Тому проектування подібних систем являє собою високе «технологічне мистецтво» системного інтегратора.

Технологія NGN може бути реалізована тільки за допомогою механізмів пакетної передачі й технологій програмної комутації, які є основними елементами мультисервісної мережі.

Використання встаткування «SoftSwitch» надає реальні можливості автоматизації процесу імпорту даних про представлені послуги в існуючі на підприємствах автоматизовані системи комплексних розрахунків (АСКР).

Саме головне в NGN – облік стану балансу між її вартістю, надійністю і якістю надаваних послуг.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. На ній показано, що структура системи складається з наступних блоків:

– Блок збирання статистичного навантаження на вузли корпоративної мережі побудованої на технології бездротової NGN.

– Блок аналізу статистичного навантаження.

– Блок балансування навантаження у корпоративній мережі побудованій на технології бездротової NGN.

– Блок формування звітів.

– Блок формування сигналів керування навантаженням.

Створений у результаті роботи над роботою, прототип центра, що засвідчує, – СКЗІ можна віднести до корпоративного рішення (варіант інсорсингу). Перевагою цього програмно-апаратного комплексу є його масштабованість, що дозволяє включати розроблену систему як у жорстко задану ієрархічну структуру, так і будувати на його базі мережні моделі із кросс-сертифікацією.

Для рішення завдання вибору числа й розміщення елементів управляючих центрів (УЦ), представленого у вигляді інформаційно-обчислювального комплексу, запропоновано використовувати евристичні алгоритми, засновані на методах локальної оптимізації й теорії масового обслуговування. Альтернативний підхід до рішення завдання дискретного математичного програмування, заснований на теорії графів, неприйнятний через високу

обчислювальну складність при великій кількості вузлів у корпоративній мережі побудованій на технології бездротової NGN. Наприклад, у корпоративній мережі побудованій на технології бездротової NGN з 10 вузлів існує 2^{45} варіантів розташування ліній зв'язку, включаючи безліч тривіальних випадків. Якщо припустити, що аналіз кожного варіанта становить 1 секунду, то на дослідження буде потрібно більш ніж $9*10^8$ років.

Рисунок 3.2 – Функціональна схема системи

На змістовному рівні завдання побудови оптимальної корпоративної мережі побудованої на технології бездротової NGN з УЦ формулюється в такий спосіб. Виходячи із заданих значень інтенсивності запитів абонентів з

урахуванням вводимих допущень і обмежень визначити оптимальні за критерієм мінімуму наведених витрат функціональні параметри корпоративної мережі побудованої на технології бездротової NGN: число й розміщення програмно-апаратних модулів УЦ у пунктах корпоративної мережі побудованої на технології бездротової NGN, співвіднести групи абонентів з обслуговуючими їх УК УЦ, ємність УК УЦ і каналів зв'язку. При цьому повинні дотримуватися обмеження на якість обслуговування: середній час обробки повідомлення й імовірність своєчасного обслуговування не повинні перевищувати граничних значень.

Як критерій оптимізації обрані наведені витрати на канал зв'язку й вузол корпоративної мережі побудованої на технології бездротової NGN.

Як модель, що інтерпретує інформаційний потік у корпоративній мережі побудованій на технології бездротової NGN з УЦ, розглядається дві корпоративної мережі побудованої на технології бездротової NGN масового обслуговування (СеМО):

– багатофазна СеМО з відмовами й повторними викликами;

– багатофазна СеМО комбінованою комутацією.

Завдання пошуку оптимальної структури корпоративної мережі побудованої на технології бездротової NGN успішно вирішуються з використанням евристичних алгоритмів, заснованих на методах локальної оптимізації зі спрямованим перебором варіантів структури корпоративної мережі побудованої на технології бездротової NGN.

На першому етапі вся безліч абонентських пунктів розбивається на групи, розташовувані в окремих зонах обслуговування УК УЦ. У процесі роботи алгоритму число УК УЦ (відповідно й чисто зон) змінюється від мінімального (наприклад, рівного одиниці), до максимального. Для кожного значення шукається локальне оптимальне розміщення УК УЦ. Абонентські місця приєднуються до найближчих УК УЦ.

Далі по черзі вибирається найкраще місце розташування кожного УК УЦ у межах своєї зони. Найкращий пункт для розміщення УК УЦ у межах своєї зони вибирається не із всіх можливих варіантів, а з пунктів підмножини A^*, число елементів якого істотно менше в порівнянні з безліччю A. Додатково прискорити процес перебору дозволяє уведення околиць, у які входять до десяти варіантів можливого розташування УК УЦ. Після УК УЦ $д$-й зони процес повторюється із УК УЦ першої зони. Якщо, починаючи з будь-якого УК УЦ, $д$ раз не вдалося поліпшити значення оптимизуємого функціонала, то процес вибору розміщення $д$ УК УЦ у фіксованих зонах припиняється, тому що досягається локальний екстремум.

При фіксованій кількості УК УЦ рішення завдання закінчується, коли перестановка УК УЦ не приводить до мінімізації оптимізуємого функціонала.

Після того як корпоративна мережа побудована на технології бездротової NGN із УК УЦ сформована (знайдене місце розташування всіх УК УЦ і зроблене прикріплення абонентів), обчислюється значення параметрів оптимізації (середній час обслуговування T и ймовірність своєчасного обслуговування Q). Знайдені значення рівняються із граничними величинами (T_b, Q_L). Якщо обмеження не виконуються, то поточна структура виключається з розгляду.

Оптимальна структура корпоративної мережі побудованої на технології бездротової NGN відповідає мінімальному значенню функціонала серед отриманих величин при різному числі УК УЦ.

У цьому розділі основним питанням є поліпшення якості функціонування корпоративної мережі побудованої на технології бездротової NGN УЦ шляхом адаптивного балансування навантаження між його керуючими компонентами.

Серед аналізованих підходів по керуванню навантаженням і динамічним керуванням потоками інформації (метод рельєфів, матричний і хвильовий алгоритми, ігровий і ймовірносно-ігровий методи) для імітаційного моделювання системи балансування навантаження на УК УЦ обраний матричний метод.

Аналіз вироблявся по наступних ознаках:

– вид аналізованої інформації;

– тип реакції на вхідний потік;

– тип системи керування;

– критерій оптимізації;

– обсяг службової інформації, що пересилається;

– часові витрати;

– складність реалізації.

Недоліком матричного методу для розглянутої прикладної області є критерій оптимізації у вигляді довжини шляхи комутації. Це можна усунути шляхом формування іншого критерію оптимізації, адаптованого під роботу корпоративній мережі побудованій на технології бездротової NGN з УЦ.

Матричний метод балансування навантаження орієнтований на алгоритми знаходження найкоротших шляхів. Даний метод дозволяє знайти довжини найкоротших шляхів між всіма вузлами корпоративної мережі побудованої на технології бездротової NGN одночасно, ґрунтуючись на застосуванні операцій над матрицями відстаней. У розробленій модифікації методу пропонується замінити параметр довжини зв'язку інтегральним критерієм, розрахованим раніше.

У цьому випадку структуру корпоративної мережі побудованої на технології бездротової NGN можна представити у вигляді матриці адаптивного балансування навантаження корпоративної мережі побудованої на технології бездротової NGN, де показник оптимальності визначається як результат розподілу інтегрального критерію z-ого вузла (n_i) на показник доступності лінії зв'язку з h-ним вузлом.

Запропонована в роботі модифікація матричного підходу полягає в наступному:

– пропонується замінити параметр довжини зв'язку інтегральним критерієм;

– на кожному кроці побудови матриці балансування в додатковій матриці запам'ятовуються проміжні транзитні вузли. Це дозволяє по результатом багатокрокового перерозподілу навантаження видавати інструкції для УК УЦ по маршрутизації вступні на них повідомлень.

Запропоноване рішення по поліпшенню якості функціонування корпоративної мережі побудованої на технології бездротової NGN з УЦ рівнялося з існуючими моделями якості обслуговування (Quality of Service). За результатами порівняння зроблений вивід про неможливість рішення завдання перерозподілу навантаження між УК УЦ винятково засобами цих моделей.

Результатом моделювання є вказівки для УК УЦ, сформовані за допомогою програмного модуля розробленої системи, засобами якого вузли можуть перенаправляти надлишкове навантаження на аналогічні компоненти. З наведених результатів моделювання можна зробити вивід, що розроблена модель динамічного балансування навантаження корпоративної мережі побудованої на технології бездротової NGN враховує основні характеристики роботи УЦ, дозволяючи оптимально перерозподілити вхідний інформаційних потік на її керуючих компонентах. У результаті балансування не тільки знімається надлишкове завантаження із УК УЦ, але й відбувається перерозподіл вхідного потоку з недоступних вузлів.

Матриця доступності, використовувана при моделюванні, дозволяє врахувати залежність від якості лінії зв'язку, що актуально, як при роботі через Інтернет-провайдерів у корпоративній мережі побудованій на технології бездротової NGN загального користування, так і у випадку корпоративній мережі побудованій на технології бездротової NGN корпоративного УЦ, що діє у ЛОМ або розподіленої VPN-корпоративної мережі побудованої на технології бездротової NGN.

Модифікація матричного підходу знаходження оптимальних для прийняття надлишкового навантаження вузлів і введення інтегрального показника, дозволяють сформувати нотації для перемикання навантаження не

тільки на вигідний УК УЦ, але й організувати маршрут перерозподілу, що складає з декількох транзитних вузлів.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. Після початку роботи розробленого ПЗ ми потрапляємо до головного блоку системи звідки через ланку дій відбувається наступне:

– Інтерфейс ПЗ.

– Налаштування ПЗ.

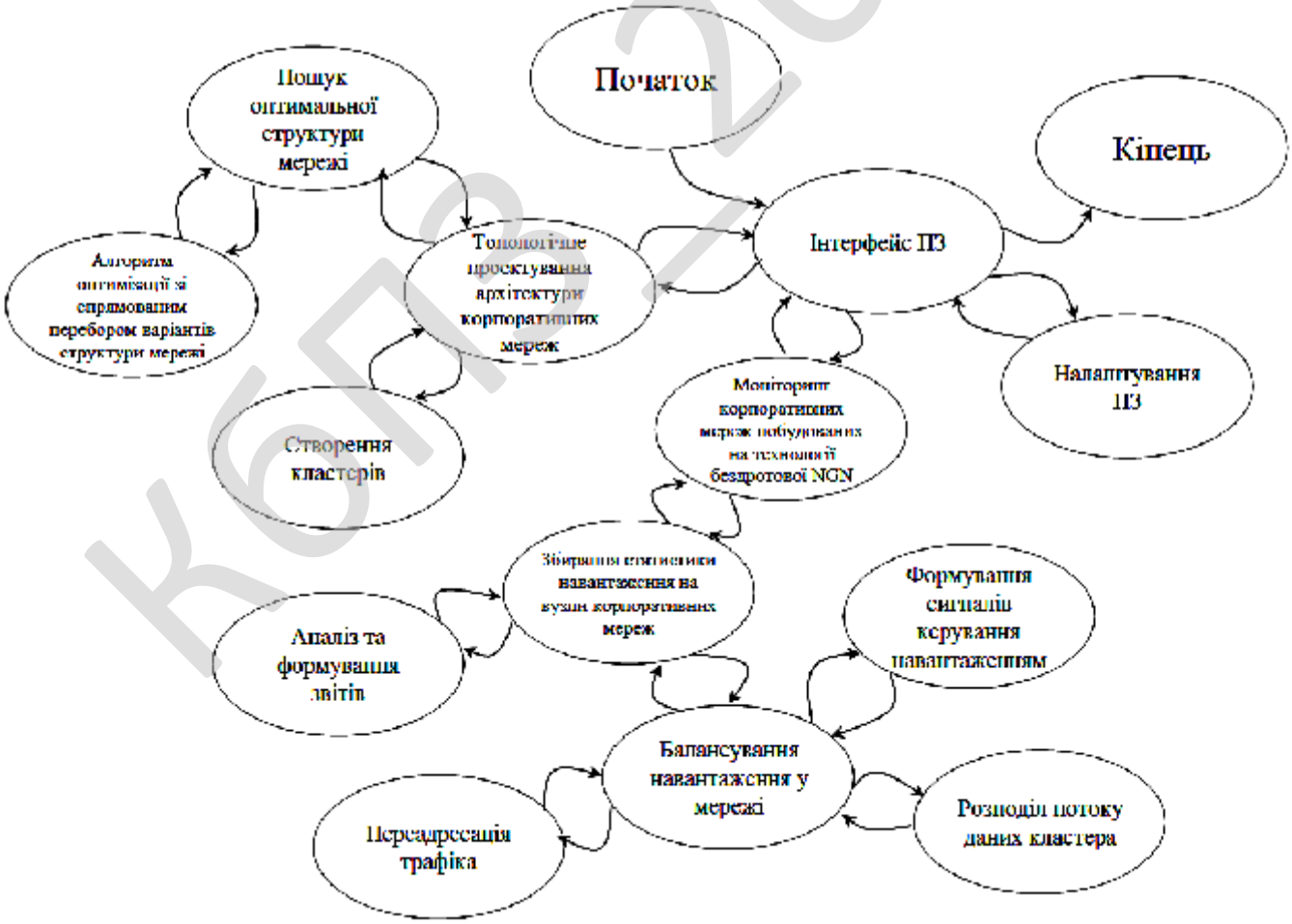

Рисунок 3.3 – Діаграма взаємодії процесів

– Топологічне проектування архітектури корпоративних мереж.

– Пошук оптимальної структури мережі.

– Алгоритм оптимізації зі спрямованим перебором варіантів структури мережі.

– Створення кластерів.

– Моніторинг корпоративних мереж побудованих на технології бездротової NGN.

– Збирання статистики навантаження на вузли корпоративних мереж.

– Аналіз та формування звітів.

– Балансування навантаження у мережі.

– Переадресація трафіка.

– Формування сигналів керування навантаженням.

– Розподіл потоку даних кластера.

Таким чином, розглянувши опис системи, функціональну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

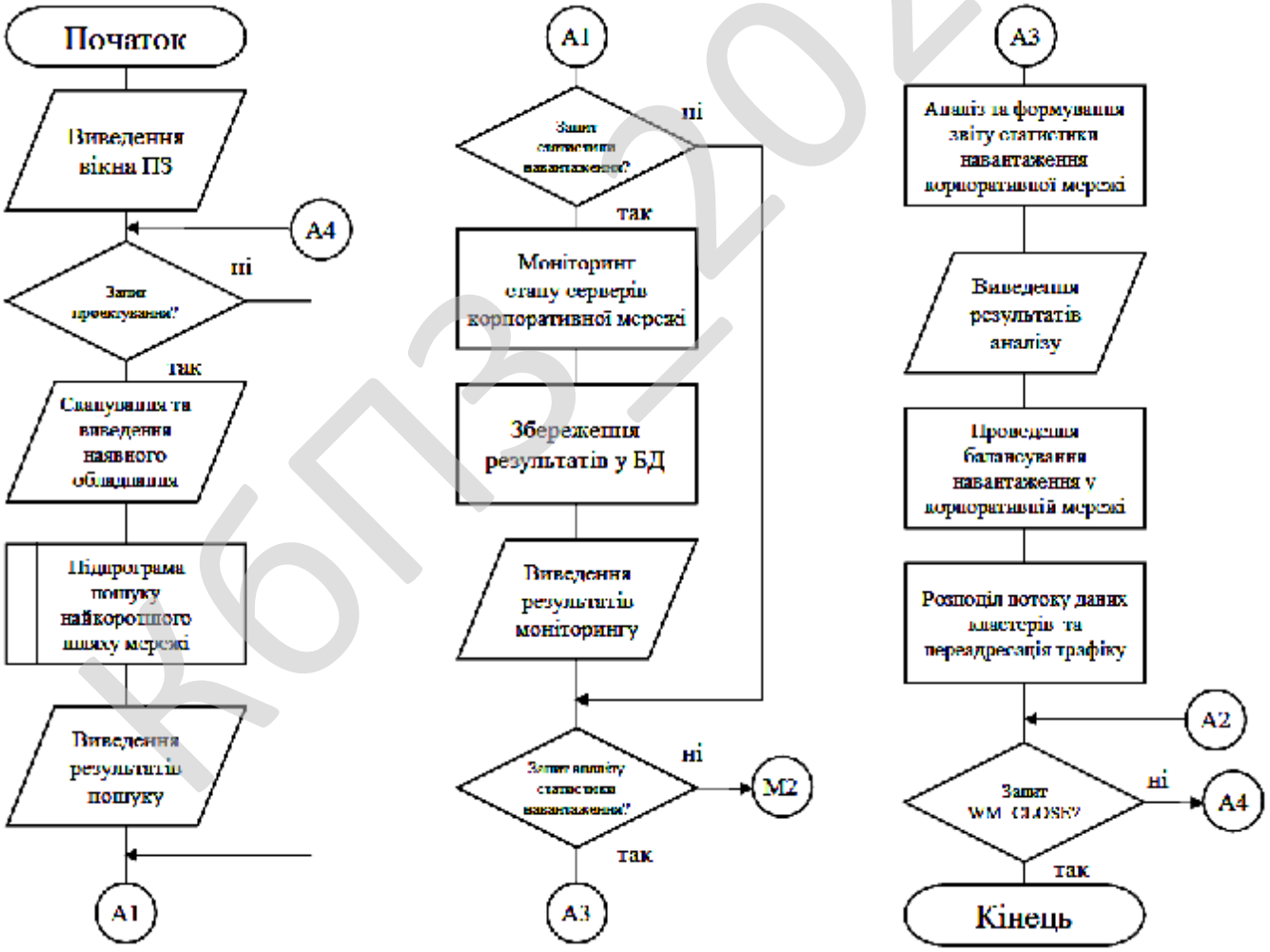

Рисунок 4.1 – Блок схема основної програми

З якої видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

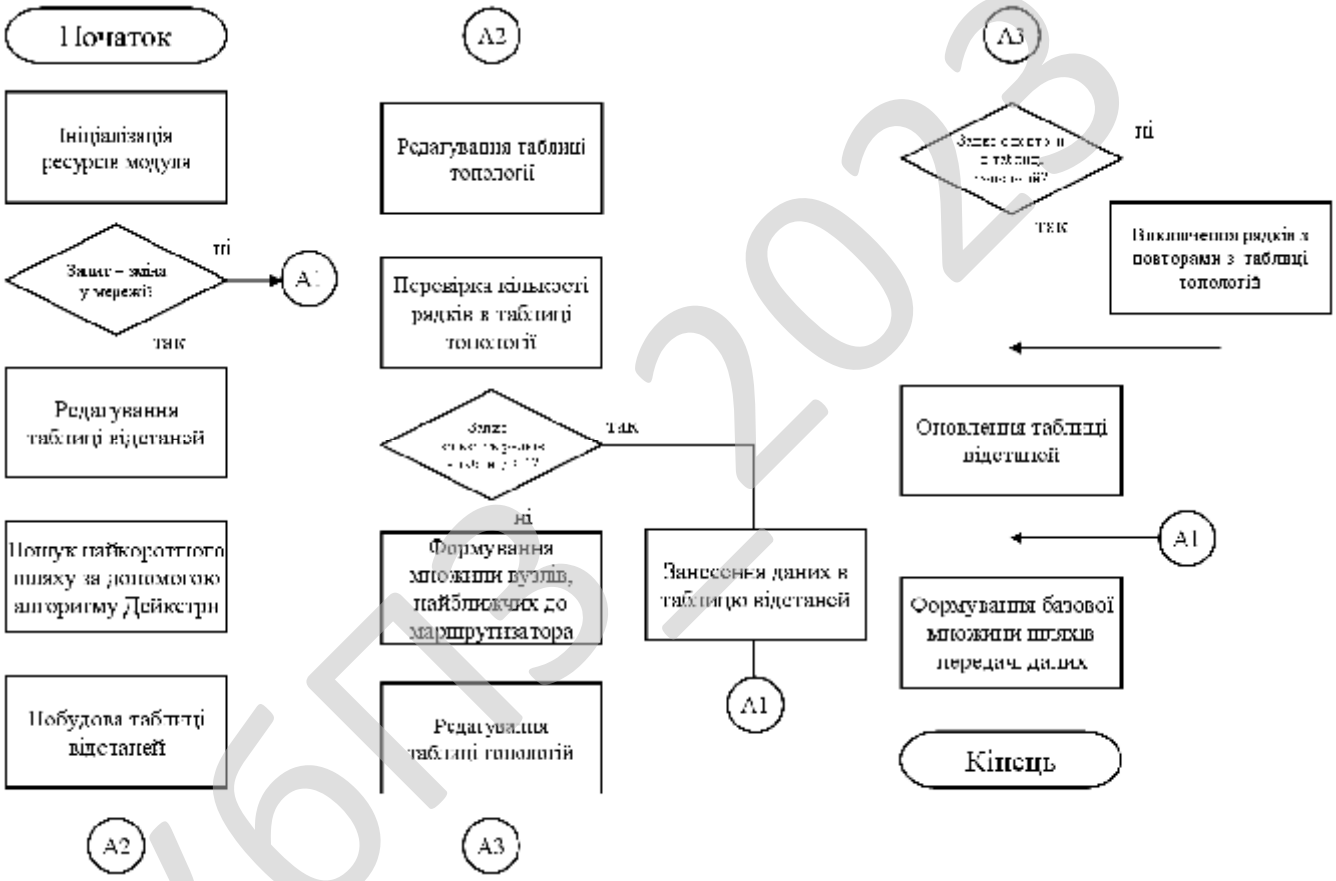

Рисунок 4.2 – Блок схема підпрограми

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем я враховував, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те,

що при розробці програми слід надати особливу увагу модулю проектування архітектури корпоративних мереж.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

Опис алгоритмів функціонування системи

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

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.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

Опис функцій ПЗ проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

```
       procedure TForm1.CClick(Sender: TObject);
       var
// об’ява змінних
        N: TNetworkNeighborhood;
        List: TStringList;
        i, j: Integer;
        ListViewItem: TListItem;
        WorkgroupNode, ComputerNode: TTreeNode;
       Begin
// початок
        Screen.Cursor:=crHourGlass;
        try
// Ініціалізація об’ єктів та сканування мережі
         N:=TNetworkNeighborhood.Create;
         try
// Отримання списку всіх комп’ ютерів в мережі
          Memo1.Lines.Clear;
          N.ListComputers(Memo1.Lines);
// Отримання списку всіх робочих груп і комп’ ютерів в мережі,
// відсортованих в алфавітному порядку
          List:=TStringList.Create;
          ListView1.Items.Clear;
          try
           N.ListNetwork(List);
           for i:=0 to List.Count – 1 do begin
            ListViewItem:=ListView1.Items.Add;
            ListViewItem.Caption:=List[i];
            ListViewItem.ImageIndex:=Integer(List.Objects[i]);
           end;
          finally
           List.Free;
```

```
          end;
// Побудова дерева робочих груп і комп’ ютерів в мережі
          TreeView1.Items.Clear;
          for i:=0 to N.Count – 1 do begin
           WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
           WorkgroupNode.ImageIndex:=1;
           WorkgroupNode.SelectedIndex:=1;
           for j:=0 to (N.Objects[i] as TStrings).Count – 1 do begin
            ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode,
                          (N.Objects[i] as TStrings).Strings[j]);
            ComputerNode.ImageIndex:=0;
           end;
          end;
// Отримання IP адрес комп’ ютерів
         GetIPAddresses(N, Memo2.Lines);
         finally
          N.Free;
         end;
         TreeView1.FullExpand;
        finally
         Screen.Cursor:=crDefault;
        end;
       end;
```

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

```
       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 сервісе              }
          ( 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
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою Threefish – в криптографії симетричний блоковий криптоалгоритм, розроблений автором Blowfish та Twofish, американським криптографом Брюсом Шнайером 2008 року для використання в хеш-функції Skein і як універсальну заміну наявним блоковим шифрам. Основними принципами розробки шифру були: мінімальне використання пам'яті, необхідна для використання в хеш-функції стійкість до атак, простота реалізації та оптимізація під 64-розрядні процесори.

Структура алгоритму

Threefish має дуже просту структуру і може бути використаний для заміни алгоритмів блочного шифрування, бувши швидким і гнучким шифром, що працюють в довільному режимі шифрування. Threefish S-блоки не використовує, заснований на комбінації інструкцій виключаючого або, складання і циклічного зсуву.

Як і AES, шифр реалізований у вигляді підстановочно-перестановочної мережі на оборотних операціях, не будучи шифром мережі Фейстел.

Алгоритм передбачає використання tweak-значення, свого роду вектора ініціалізації, дозволяючи змінювати таким чином значення виходу, без зміни ключа, що має позитивний ефект як для реалізації нових режимів шифрування, так і на криптостойкости алгоритму.

Як результат думки авторів, що кілька складних раундів часто гірше ніж застосування великого числа простих раундів, алгоритм має нетрадиційно велику кількість раундів – 72 або 80 при ключі 1024 біт, проте, за заявою творців, його швидкісні характеристики випереджають AES приблизно вдвічі. Варто зауважити, що через 64-бітної структури шифру, дана заява має місцево лише на 64-розрядної архітектури. Тому, Threefish, як і Skein [1], заснований на ньому, на 32-розрядних процесорах показує значно гірші результати ніж на «рідному» обладнанні.

Ядром шифру є проста функція «MIX», перетворювальна два 64-бітових беззнакових числа, в процесі якої відбувається складання, циклічний зсув (ROL / ROR), і додавання по модулю 2 (XOR) .

Нижче представлений код MIX-функції для Threefish-1024 [2]:

```
<syntaxhighlight lang="C">
// Константи для циклічного зсуву
int R16 [8] [8] = {
  {55, 43, 37, 40, 16, 22, 38, 12},
  {25, 25, 46, 13, 14, 13, 52, 57},
  {33, 8, 18, 57, 21, 12, 32, 54},
  {34, 43, 25, 60, 44, 9, 59, 34},
  {28, 7, 47, 48, 51, 9, 35, 41},
  {17, 6, 18, 25, 43, 42, 40, 15},
  {58, 7, 32, 45, 19, 18, 2, 56},
  {47, 49, 27, 58, 37, 48, 53, 56},
};
// D – раунд, j – індекс в таблиці циклічного зсуву
void mix (int j, int d) {
  usigned long long rotl;
  y [0] = x [0] + x [1];
  rotl = R16 [d% 8] [j];
  y [1] = (x [1] << rotl) | (x [1] >> (64 – rotl));
  y [1] ^ = y [0];
```

```
}
</Source>
```

Процедура розшифрування обернена процедурі зашифрування і містить зворотну функцію DEMIX.

Кожен з 72 раундів Threefish-256 і Threefish-512 має чотири MIX перетворення, Threefish-1024 – вісім звернень до MIX функції.

Безпека

За заявою авторів, алгоритм має більш високий рівень безпеки, ніж AES. Існує атака на 25 з 72 раундів Threefish, в той час як для AES – на 6 з 10. Threefish має показник фактора безпеки 2.9, в свою чергу, AES всього 1.7 [3]

Для досягнення повної дифузії, шифру Threefish-256 досить 9 раундів, Threefish-512 – 10 раундів і Threefish-1024 – 11 раундів. Виходячи з цього, 72 і 80 раундів відповідно в середньому, забезпечать кращі результати, ніж існуючі шифри. [4]

У той же час, алгоритм має набагато простішу структуру і функцію перетворення, проте виконання 72-80 раундів, на думку дослідників, забезпечує необхідну стійкість. Вживаний розмір ключа від 256 до 1024 біт зводить нанівець можливість повного перебору паролів при так званій атаці грубою силою (brute force attack) на сучасному обладнанні.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ яке зображено на рисунку 5.1. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні розділи:

1. Розділ меню яке складається з наступних елементів:

– Файл.

– Кластер.

– Хост.

– Параметри.

– Довідка.

2. Розділ функцій яке складається з наступних елементів:

– Топологічне проектування мережі.

– Розподіл навантаження трафіку.

3. Розділ даних про кластер яке складається з наступних елементів: Назва кластера; IP-адреса кластера; Маска під мережі кластера; Тип кластера.

4. Розділ журналювання.

5. Розділ налаштування.

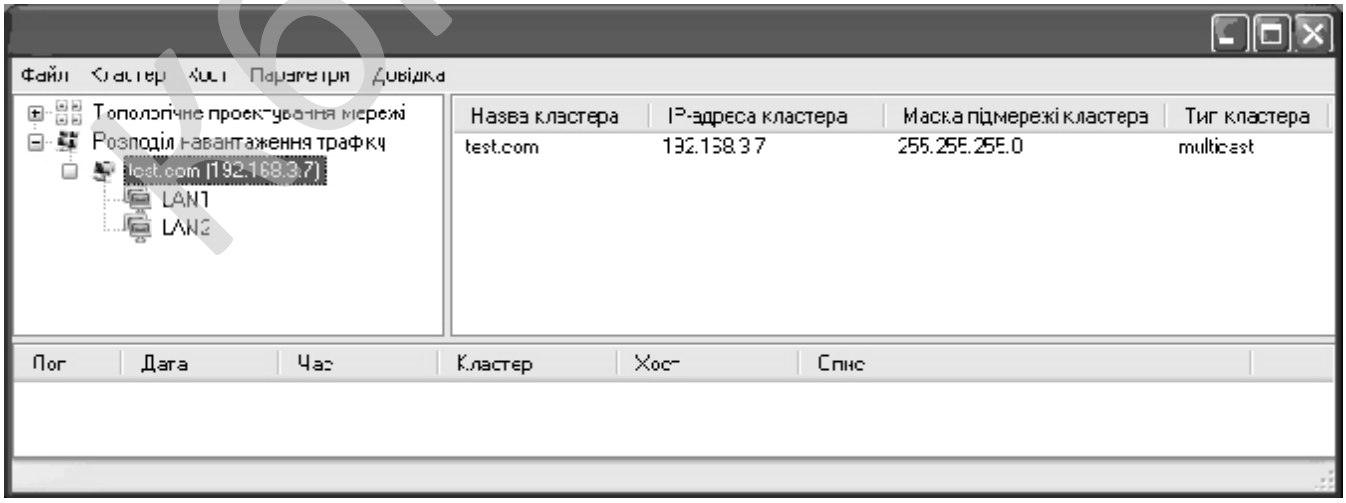

Рисунок 5.1 – Головне вікно програми

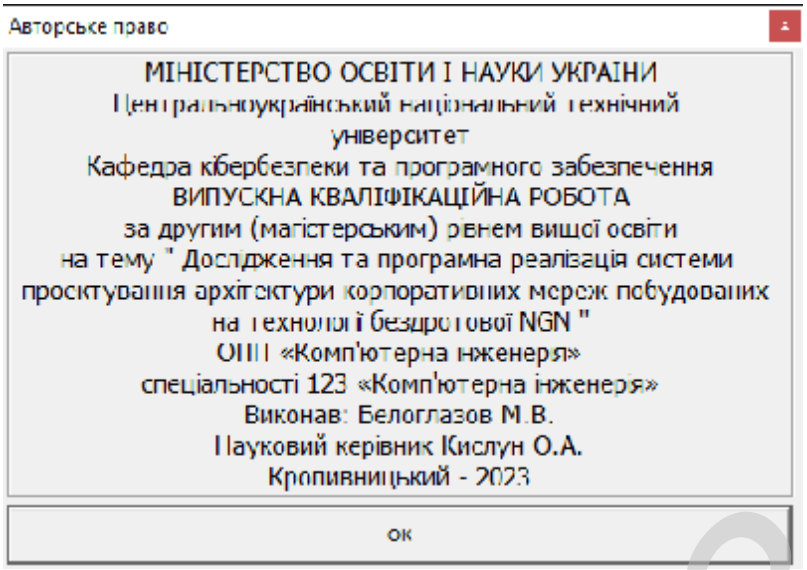

Рисунок 5.2 – Авторське право

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення. Обрано умови розповсюдження – Shareware. Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно. Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Метою розробки є дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Об'єктом дослідження є процес проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Предметом дослідження є методи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Методи дослідження базуються на методах теорії телекому, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– Розроблено вітчизняний продукт проектування архітектури корпоративних мереж побудованих на технології бездротової NGN, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведене дослідження та виконана програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	280
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	28000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 44 = 90 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	90	Ф 7.1-7.4
Впровадження	13	Д13
Всього	131	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{131 \cdot 1}{60 - 5} = 2,4 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	9	810	13,5
Монітор	60	9	540	9
Клавіатура	30	9	270	4,5
Маніпулятор «мишка»	30	9	270	4,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор–маршрутизатор	30	2	60	1
Кабельні господарства ЛВС на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	48,49

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{mic}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{49 \cdot 3}{1,2} = 122,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 122,5/(60·8) = 0,2 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Розміщення графіки і контенту на Інтернет сторінках	0,5	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	0,25	14000	10500
Продакт-менеджер	0,25	13128	9846
Інженер-програміст	2,4	13000	93600
Інженер-електронщик	0,2	12000	7200
Інженер-системотехнік	0,25	12000	9000
Адміністратор мережі	0,5	12000	18000
Системний програміст	0,25	12000	9000
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	12000	9000
Бухгалтер-економіст	0,5	14000	21000
Всього за період розробки	$R_{сп}$ = 5,1	-	$Ф_{роб}$ = 196146

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{196146}{5,1 \cdot 60} = 641 \, грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R_{сп}^{1} S_{у} Ц_{пл}, \quad (7.9)$$

де: $R_{сп}^{1}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_{у}$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно $8 м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R_{сп}^{1} \cdot Ц_{м}, \quad (7.10)$$

де: $Ц_{м}$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Компбест за 06.10.23 – джерело https://compbest.com.ua/.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	Intel Core i5-4590 (4 ядра по 3.3-3.7GHz), MB cache	1750
Системна плата	MSI H81M-P33 6 x USB 2.0, 2 x USB 3.(VGA, DVI, 2 x PS/2, LAN (RJ-45), 5 x Audic FireWire	1200
Відеокарта	nVidia GeForce GTX 660, 2 GB GDDR5, 192 bit	750
Жорсткий диск	240 GB SSD	1200
Оперативна пам’ять	Samsung DDR3-1333 8Gb PC3-10600R EC(Registered (M393B1K70CH0-CH9Q5)	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	15	10947	16420,5	180625,5
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	199177

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	199177	-	-
Всього по групі	199177	50	99588,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	0	20	0,0
6. Господарський інвентар	28000	25	7000
Всього по групі	33190	-	8297,5
7. Нематеріальні активи	120000	10	12000
Разом	$К_р$ = 1760367		$А_р$ = 190286

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 641 \cdot 131/280 = 300 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 300 \cdot 10 \cdot 0{,}01 = 30 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(300+30) = 73 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot N_м \, n_{міс}. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 100):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23 грн./шт., DVD-R box – 33,6 грн./шт.

$$З_{М2} = 100 \cdot 33{,}6 = 3360 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_{з.}, \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 3360 + 1702)/280 = 18 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 300 \cdot 15 \cdot 0{,}01 = 45 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 280 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 190286 \cdot 3/(280 \cdot 12) = 170 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 300+30+73+45+18+45+170 = 681 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1. Основна зарплата виконавців	$З_о$	300
2. Додаткова зарплата виконавців	$З_д$	30
3. Відрахування на соціальні потреби	$C_{оц}$	73
4. Загальногосподарські витрати	$Г_{осп}$	45
5. Витрати на матеріали	$З_М$	18
6. Освоєння нових операційних систем, мов програмування	$О_п$	45
7. Амортизація основних фондів	$A_м$	170
8. Повна собівартість програмного забезпечення	$C_п$	681
9. Плановий прибуток	$П_р$	341
10.Ціна підприємства $Ц_п = C_п + П_р$	$Ц_п$	1022
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	204,4
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	1226,4

Визначимо плановий прибуток за рівнем рентабельності (P_n) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_n \cdot C_n, \quad (7.22)$$

де: P_n – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 681 = 341 \text{ грн.}$$

7.6 Визначення об'єму капітальних вкладень у споживача програмної продукції

Об'єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.9.

Таблиця 7.9 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	1226
Всього капітальних витрат	–	1226

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування системи	$З_р$	27500	7500
2. Витрати на електроенергію	$З_{ел}$	1488	1030
3. Витрати на амортизацію	$З_{ам}$	0	307
Всього витрат за рік	$І$	28988	8837

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення витрати на обслуговування системи зменшились з 27500 грн до 7500 грн на рік.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	1226	–	306,5
Всього відрахувань	-	–	1226	–	306,5

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($T_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot T_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 0{,}545 \cdot 1300 \cdot 2{,}1 = 1487{,}85 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}545 \cdot 900 \cdot 2{,}1 = 1030{,}05 \text{ грн.}$$

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (1022 - 681) \cdot 280 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 199177 +$$
$$+0{,}25 \cdot 33190 + 0{,}1 \cdot 28000) \cdot 3/12 = 50208 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1760367}{(1022 - 681) \cdot 280 \cdot 12 / 3} = 4{,}6 \text{ роки}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (28988\text{-}8837)\text{-}0{,}25 \cdot 1226 = 19845 \text{ грн.}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	280
2. Повна собівартість розробленої програми	Грн.	681
3. Ціна розробленої програми	Грн.	1022
4. Плановий прибуток від реалізації розробленої програми	Грн.	341
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1760367
7. Загальний прибуток від реалізації програмної продукції	Грн.	95480
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	50208
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	4,6
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	1226
11.Величина економічного ефекту у користувача програмної продукції	Грн.	19845
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,1

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{1226}{28988 - 8837} = 0{,}1 \; року .$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

В охорону праці включають санітарно-гігієнічні, лікувально-профілактичні та організаційно-технічні системи правових і соціально-економічних заходів.

В кожній ІТ компанії є трудові відносини з працівниками. Згідно закону України “Про охорону праці” [3] кожна компанія впроваджує заходи з охорони праці. Реалізується трудові відносини з вживанням необхідних засобів з охорони праці та розробки відповідних документів:

– Інструкцій з охорони праці по кожній професії і загальні;

– Положення про охорону праці;

– Накази з охорони праці;

– Журнали реєстрації та інструктажу.

Роботодавець створює відділ який працює відповідно до типового положення, яку затверджується центральним органом виконавчої влади і забезпечує виконання вимог державної політики у сфері охорони праці.

За недотриманням вимог, керівники ІТ компаній можуть бути притягнуті до відповідальності, яка виглядає у виді накладання штрафу. Якщо в результаті порушення умов охорони праці є постраждалі працівники то керівні особи ІТ компаній притягуються до кримінальної відповідальності.

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи

з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп’ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2. Аналіз умов праці на робочому місці програміста

Згідно НПАОП 0.00 – 1.28 – 10 «Правила охорони праці під час електронно-обчислювальних машин» площа повинна задовольняти умові – не менш 6 $м^2$ на одне робоче місце. Кратність повітрообміну в приміщенні вузла також регламентується ДСанПіН 3.3.2.007 – 98, вона повинна становити 20 $м^3$/годину на одне місце. Виконання даних вимог забезпечить підтримку в приміщенні вузла оптимального значення вологості й складу повітря.

Відповідно ДБН В.2.5 – 28 – 2006 роботу програміста можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення вузла можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при сполученому висвітленні), повинен становити 0,5%, освітленість при штучному висвітленні повинна становити 300 лк.

За результатами виміру освітленості відділом охорони праці величина освітленості від системи загального штучного висвітлення лежить у межах 200-250 лк, що не відповідає вимогам, які пред'являються до приміщення.

Відповідно ДСанПіН 3.3.2.007 – 98 рівні звукового тиску в робочому приміщенні не повинні перевищувати в октавних смугах із середньо геометричними частотами наступних значень, наведених у таблиці 8.1.

У приміщенні перебувають наступні джерела шуму: електродвигуни внутрішнього вентилятора ЕОМ; працюючі принтери; працюючі дисководи. Шум, вироблений вентилятором можна класифікувати як постійний, всі інші джерела шуму, як імпульсні. Відповідно паспорта на приміщення рівень звуку, Дб(А), обмірюваний за шкалою (А) шумоміра досяг величини 28,3 Дб(А) при роботі всього встаткування вузла, включаючи й ксерокс. Це дозволяє зробити висновок про відповідність рівня звуку в приміщенні вимогам нормативних актів.

Таблиця 8.1 – Допустимі спектри рівнів звукового тиску

Робоче місце	Рівень звукового тиску, дБ, в октавних смугах із середньогеометричними частотами, Гц								Рівень звуку і еквівалентний рівень звуку, дБА
	63	125	250	500	1000	2000	4000	8000	
Приміщення конструкторських бюро, програмістів обчислювальних машин, лабораторій для теоретичних робіт і опрацювання експериментальних даних, прийому хворих в медпунктах	71	61	54	49	45	42	40	38	50

Ергономічні вимоги до робочого місця працюючого з ВДТ ЕОМ і ПЕОМ нормуються НПАОП 0.00 – 1.28 – 10**.** Оптимальне положення тіла того, що працює забезпечується відповідною конструкцією робочого місця, а також регуляцією висоти робочої поверхні, сидіння, простори й підставки для ніг. Даного місця програміста не мають регульованих параметрів. Відмінності

реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту дані в таблиці 8.2.

Таблиця 8.2 – Відмінності реальних параметрів робочого місця від параметрів відповідні вимоги нормативного акту

Ріст людини, см	Висота робочої поверхні мм,	Висота простору для ніг, мм	Висота робочого сидіння, мм
175	765 (740)	655 (600)	450 (440)

У дужках зазначені реальні значення параметрів робочого місця; всі вони не відповідають параметрам, зазначеним у стандарті.

Параметри мікроклімату можуть мінятися в широких межах, тоді як необхідною умовою життєдіяльності людини є підтримка сталості температури тіла завдяки властивості терморегуляції, тобто здатності організму регулювати віддачу тепла в навколишнє середовище.

У приміщеннях, де встановлені комп'ютери, повинні дотримуватися певні параметри мікроклімату. У санітарних нормах ДСН 3.3.6.042 – 99 встановлені величини параметрів мікроклімату, що створюють комфортні умови. Ці норми встановлюються в залежності від пори року, характеру трудового процесу і характеру виробничого приміщення (див. табл. 8.3).

Таблиця 8.3 – Параметри мікроклімату для приміщень, де встановлені комп'ютери

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні	22 – 24°С
	Відносна вологість	40 – 60%
	Швидкість руху повітря	до 0,1 м/с
Теплий	Температура повітря в приміщенні	23 – 25°С
	Відносна вологість	40 ... 60%
	Швидкість руху повітря	0,1 ... 0,2 м / с

8.3 Розробка заходів з умов поліпшення охорони праці

Перерахуємо проведені заходи щодо забезпечення умов праці на робочому місці програміста.

З точки зору забезпечення електробезпеки до цих заходів можна віднести: устаткування розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв; періодична перевірка всіх приладів і пристроїв; щорічна здача іспитів з охорони праці.

З точки зору забезпечення оптимальних умов мікроклімату, рівня звуку і освітленості до цих заходів можна віднести: організацію природної вентиляції, за допомогою дефлектора, для забезпечення необхідного повітрообміну в приміщенні вузла; організацію системи центрального опалювання, для підтримки оптимальної температури в холодний період року; організацію штучного загального освітлення, для забезпечення необхідних умов зорової роботи, що відповідають, оформлення паспорта на приміщення вузла, з занесенням в нього вимірювань освітленості і рівня звуку, проведених відділом охорони праці.

З точки зору забезпечення пожежної безпеки до цих заходів можна віднести наявність схеми евакуації з приміщення вузла, у випадку пожежі, повішену на вхідні двері.

Аналіз умов праці на робочому місці інженера-програміста показав, що на робочому місці не виконуються вимоги ергономіки. Для виконання їх можна запропонувати заміну не регульованого сидіння на крісло з регульованими ергономічними параметрами, а також заміну використовуваного столу на робоче місце оператора ЕОМ.

Різними фірмами в сукупності розроблено понад 11 схем регулювань параметрів робочого крісла, які забезпечують: плавне переміщення сидіння по висоті за допомогою газової пружини; плавна зміна нахилу спинки і сидіння; регулювання пружинного противотиску спинки крісла на спину оператора; перестановку спинки по висоті; зміна глибини сидіння шляхом зміни вигину

краю сидіння; синхронне повторення рухів оператора сидінням і спинкою в правильному кутовому співвідношенні; синхронне повторення спинкою крісла рухів верхньої частини тулуба того, що сидить; амортизацію сидіння.

Найбільш популярними моделями комп'ютерних крісел, які володіють чотирма основними регулюваннями (висоти сидіння, висоти, глибини і нахилу спинки), є італійські, фінські моделі «Senior», «Volos», «Ergo», «Toronto», «Bini», «Metro», «NewStar», «Capris», «Fenix», «Xenus», «Quintus» і т.д. Подібні крісла, відрегульовані відповідно до зростання і ваги оператора, а також характеру виконуваної роботи, дозволяють понизити навантаження на опорно-руховий апарат людини, що працює за комп'ютером.

8.4 Розрахункова частина

Для захисного штучного заземлення застосовуються вертикальні електроди: металевий куток *50·50·5 мм*., довжиною *L=3 м.,* та горизонтальний електрод – металева полоса з перетином *50·5 мм*. Напруга – *220/380 В*. Розрахункова схема розташування заземлюючих електродів – — по контуру (прямокутником).

Розрахунок проведем за допустимим опором розтіканню струму заземлювача.

Початкові дані для розрахунку захисного заземлення: тип верхнього шару грунта – чорнозем, нижнього шару грунта – глина (питомий опір $\rho_2 = 40$ *Ом·м)*. Умовна товщина верхнього шару грунта: *H=0,6 м*. Відстань між вертикальними заземлювачами (електродами) *A=3 м*. Глибина закладення горизонтального контура заземлення *t=0,75 м*. Опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*. Необхідно визначити необхідну кількість вертикальних заземлювачів та длину полоси (горизонтального заземлювача).

Розрахунок

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,75+3/2=2,25\ м.$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi\rho = 1,36 \cdot 40 = 54,5\ Ом \cdot м.$$

де $\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони у багатошаровому грунті [11];

$\rho_2 = 40\ Ом \cdot м.$ – табличне значення питомого опору нижнього шару грунта (глина) [11].

Еквівалентний діаметр вертикальнного електрода (кутка) [11]:

$$Dв=0,95 \cdot K= 0,95 \cdot 50 = 47,5\ мм. = 0,0475 м.$$

де $K= 50\ мм.$ – розмір метелевого кутка (задан).

Відношення $A/L=3/3=1$.

Опір розтіканню електричного струму одного електрода вертикального заземлювача з урахуванням заглиблення заземлювача [11]:

$$R_O=0,366(\rho/L)[lg(2L/Dв)+(1/2)lg((4T+L)/(4T-L))]=$$

$$=0,366(54,5/3)[(lg(2 \cdot 3/0,0475)+(1/2)lg((4 \cdot 2,25+3)/(4 \cdot 2,25-3))]=$$

$$=14,9\ Ом.$$

Визначаємо коефіцієнт екранування вертикальнних електродів $Кев=0,53$ при орієтовній кількості вертикальнних електродів, яке дорівнює 5 [11].

Визначаємо необхідну кількість вертикальних електродів заземлювача (без вхарування горизонтального заземлювача), при $R_{ЗН} = 4\ Ом$:

$$N=R_O/(Кев\ R_{ЗН})= 14,9 / (0,53 \cdot 4)= 7,05 \approx 7\ шт.$$

Визначаємо длину з'єднуючої полоси:

$$L_П = 1,05 \cdot A \cdot N = 1,05 \cdot 3 \cdot 7 = 18,5 \approx 18\ м.$$

Опір розтіканню електричного струму з'єднуючої полоси з урахуванням кліматичного коефіцієнта питомого опору грунта $K_П$ [11]:

$$R_П = 0,366(\rho \cdot K_П/L_П)lg(2 \cdot L_П^2/(B \cdot t))=$$

$$=0,366(40\cdot 5/18)\cdot lg((2\cdot 18^2)/(0,05\cdot 0,75))=20,1\ Ом.$$

де $K_П=5$ – табличне значення кліматичного коефіцієнта питомого опору грунта для відповідної климатичной зони для з'єднуючої полоси [11]:

$B = 50\ мм. = 0,05\ м.$- ширина з'єднуючої полоси (задана).

Загальний опір розтіканню електричного струму заземлювача [11]:

$$R =(R_O\cdot R_П)/(R_O\cdot \eta_П + N\cdot R_П\cdot Кев)=$$

$$=(14,9\cdot 20,1)/(14,9\cdot 0,55+ 7\cdot 20,1\cdot 0,53)=3,56\ Ом.$$

де $\eta_П = 0,55$ – табличне значення коефіцієнта екранування з'єднуючої полоси [11].

Умова $R \le R_{ЗН}$ виконується ($3,56 \le 4$).

*О*статочно кількість вертикальних електродів дорівнює *7*.

За потреби можна зменшити кількість електродів заземлювача, зменшивши загальний опір розтіканню електричного струму заземлювача методом зменшення питомого опору ґрунта, домішуючи у грунт безпосередньо навколи електродів заземлювача розчини солей NaCl, CaCl, сажу, соду, шлак або спеціальні суміші.

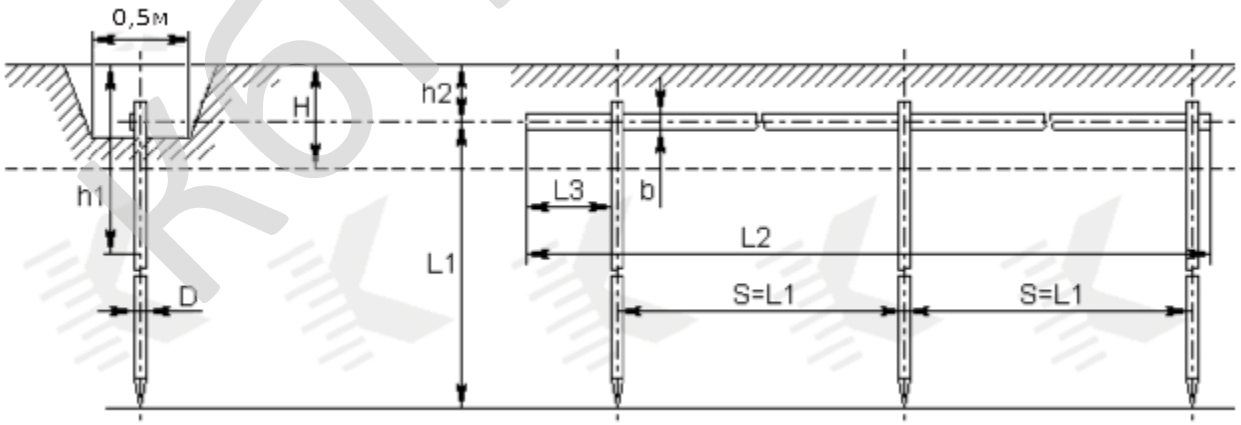

Рисунок 8.1 – Схема штучного заземлення .

8.5 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва в цілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи.

Тільки повна усвідомленість працівника про можливі небезпеки, що можуть підстерігати його на робочому місці та дотримання вимог нормативних актів з питань охорони праці та відповідних рекомендацій фахівців, дозволять значною мірою знизити негативний вплив шкідливих та небезпечних факторів при роботі з комп'ютером на організм людини.

Виконано розрахунок захисного штучного заземлення, як одного з ключових факторів безпеки програміста.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– Досліджена система проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

– На основі отриманих результатів досліджень створена програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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. 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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Threefish.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 19845 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,1 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Белоглазов М.В. Дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

2. Е. Таненбаум, Д. Уезеролл «Комп'ютерні мережі». – [5-е вид.]. – 2016. – 960 с.

3. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 1». Cisco Press. 2020. – 848 p.

4. Wendell Odom. «CCNA 200-301 Official Cert Guide, Volume 2 Premium Edition eBook and Practice Test». Cisco Press. 2020. – 624 p.

5. Scott Jernigan «CompTIA Network+ Certification All-in-One Exam Guide, Eighth Edition». 2022. – 976 p.

6. Doug Lowe «Networking For Dummies 12th Edition». 2020. – 480 p.

7. Ramon Nastase «Computer Networking: The Beginner's guide for Mastering Computer Networking, the Internet and the OSI Model». 2018. – 186 p.

8. Russ White & Ethan Banks «Computer Networking Problems and Solutions: An Innovative Approach to Building Resilient, Modern Networks». 2017. – 832 p.

9. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

10. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

11. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

12. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». *CEUR Workshop Proceedings*. Volume 2740, 2020, Pages 102-114.

13. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

14. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

15. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

16. Smirnov, O., Markovets, O. Vovk, N., Turchyn, Y., «Model of informational support for social network administrators' content creation». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 125-136.

17. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». *CEUR Workshop Proceedings* Volume 2616, 2020, Pages 366-379.

18. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated

with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

19. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

20. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

21. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

22. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

23. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». *CEUR Workshop Proceedings*, Vol 2588, P. 90-106, 2019.

24. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

25. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems*

(2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

26. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

27. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

28. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

29. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

30. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

31. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

32. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного

захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

33. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

34. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

35. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

36. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

37. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

38. О.А. Смірнов, П.С. Усік, «дослідження перспектив використання технологічних рішень в мережах 5g» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

39. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

41. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

42. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

43. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

44. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

45. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

46. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

47. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

48. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

49. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". – Випуск 3 (140). – Х.: ХУПС – 2016. – С. 36-39.

50. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Спосіб контролю ліній зв'язку телекомунікаційної системи антивірусу. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). – Харків: ХУПС. – 2016. – С. 121-127.

51. Смірнов О.А., Смірнов С.А., Дідик А.К. Метод безпечної маршрутизації метаданих у хмарні антивірусні системи. Системи озброєння та військова техніка. – Випуск 2 (46) – Х.: ХУПС – 2016. – С. 146-149.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 35-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN;

– 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 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2023 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці програміста.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 102 аркуші.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 13.12.2023 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

_____ Кислун О.А.

Дослідження та програмна реалізація системи проектування архітектури корпоративних мереж побудованих на технології бездротової NGN

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2023 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

```
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 сервісе             }
    ( 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      : '  + ' Ні'  );
      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( ' Ехо запит        : ' + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо повтор         : ' + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : ' + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : ' + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адреси : ' + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор маски адреси   : ' + IntToStr( dwAddrReps ) );
    end;
     //


    with ICMPStats.OutStats do


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


//---------------------------------------------------------------------------
```

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл NetConst.pas – ініціалізація констант

```
unit NetConst;



interface

resourcestring

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

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

 SFileError002 = '  - файл не знайдено' ;
  SFileError003 = '  - шлях не знайдено' ;
  SFileError004 = '  - не можу відкрити файл' ;
  SFileError005 = '  - немає доступу' ;
  SFileError014 = '  - не достатньо пам"яті' ;
  SFileError015 = '  - не можу знайти драйвер' ;
  SFileError017 = '  - не можу перемістити файл' ;
  SFileError019 = '  - носій захищений від запису' ;
  SFileError020 = '  - не можу знайти пристрій' ;
  SFileError021 = '  - пристрій не відкривається для читання' ;
  SFileError022 = '  - пристрій не може розпізнати команду' ;
  SFileError025 = '  - вказана область не знайдена' ;
  SFileError026 = '  - пристрій недоступний' ;
  SFileError027 = '  - сектор не знайдено' ;
  SFileError029 = '  - помилка запису на пристрій' ;
  SFileError030 = '  - помилка читання з пристрою' ;
  SFileError032 = '  - файл використовується іншою програмою' ;
  SFileError036 = '  - занадто багато відкритих файлів' ;
  SFileError038 = '  - досягнутий кінець файлу' ;
  SFileError039 = '  - диск переповнений' ;
  SFileError050 = '  - запит не підтримується' ;
  SFileError051 = '  - віддалений комп' ютер недоступний' ;
  SFileError052 = '  - у мережі 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

```
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.
```