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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп’ютерних систем”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Глобенко В.В.
«____» _____ 2022 р.

Керівник проекту
доктор технічних наук, доцент
_____ Коваленко О.В.
«____» _____ 2022 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 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

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

Глобенку Віталію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем*

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.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 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	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Глобенко В.В.
(прізвище та ініціали)

АНОТАЦІЯ

Глобенко В.В. Дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Метою розробки є дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Об'єктом дослідження є процес моніторингу LAN мереж інформаційних та комп'ютерних систем.

Предметом дослідження є методи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Методи дослідження базуються на методах побудови мереж інформаційних та комп'ютерних систем, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

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 Embarcadero RAD Studio Delphi 10.3.2.

Ключові слова: комп'ютерні науки, моніторинг, LAN, мережі інформаційних та комп'ютерних систем

ABSTRACT

Hlobenko V.V. Research and software implementation of the LAN monitoring system of information and computer networks. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the monitoring system of LAN networks of information and computer systems.

The purpose of the development is the research and software implementation of the LAN monitoring system of information and computer networks.

The object of research is the process of monitoring LAN networks of information and computer systems.

The subject of research is methods of monitoring LAN networks of information and computer systems.

Research methods are based on methods of building networks of information and computer systems, methods of mathematical statistics, methods of software development.

The result of the work is the software implementation of the LAN monitoring system of information and computer networks.

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 Embarcadero RAD Studio Delphi 10.3.2 environment.

Keywords: computer science, monitoring, LAN, networks of information and computer systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 10
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 10
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 23
2.3 Розгорнута постановка завдання 27
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 29
3.1 Опис функціонування системи 29
3.2 Розробка структурної схеми..... 52
3.3 Розробка функціональної схеми 54
3.4 Розробка діаграми процесів..... 56
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 58
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 58
4.2 Захист розробленого програмного забезпечення..... 88
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 90
6 НАУКОВА НОВИЗНА 94

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ..... 95
7.1 Техніко економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 95
7.2 Розрахунок трудомісткості розробки програмної продукції..... 97
7.3 Визначення чисельності виконавців і планового фонду зарплати..... 99
7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника..... 104
7.5 Визначення собівартості розробки та ціни програмної продукції..... 108
7.6 Визначення об’єму капітальних вкладень та експлуатаційних витрат у споживача програмної продукції..... 111
7.7 Визначення експлуатаційних витрат..... 111
7.8 Визначення економічної ефективності програмної продукції..... 113
7.9 Висновок..... 115
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ..... 116
8.1 Вступ..... 116
8.2 Шкідливі і небезпечні фактори при роботі з комп’ютером..... 117
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста..... 118
8.4 Розробка заходів з умов поліпшення охорони праці..... 121
8.5 Розрахункова частина..... 122
8.6 Висновки до розділу..... 124
9 ОСНОВНІ ВИСНОВКИ..... 125
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 127

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ЛОМ	–	локальна обчислювальна мережа
ASP		Active Server Pages – активні серверні сторінки
DHCP	–	Dynamic Host Configuration Protocol – протокол динамичної конфігурації вузла
HTTP	–	HyperText Transfer Protocol – протокол передачі гіпер тексту
IMAP	–	Internet Message Access Protocol – протокол доступу до електроної пошти Інтернету
ICMP	–	Internet Control Message Protocol – міжмережний протокол керуючих повідомлень
MMC	–	Microsoft Management Console
POP3	–	Post Office Protocol Version 3 – протокол поштового відділення, версія 3
SQL	–	Structured Query Language – мова структурованих запросів
SMTP	–	Simple Mail Transfer Protocol – простий протокол передачі пошти
SNMP	–	Simple Network Management Protocol – простий протокол керування мережею
Syslog	–	стандарт відправки повідомлень про зміни які відбуваються в мережі
UDP	–	User Datagram Protocol – протокол користувальницьких дейтаграм

ВСТУП

Актуальність теми. Розвиток засобів обчислювальної техніки відбувається в багатьох напрямках, що розширюють сферу застосування ЕОМ і підвищують ефективність їхнього використання. Найбільше застосування ЕОМ відбувається при побудові різного виду комп'ютерних мереж, як локальних або корпоративних, так і глобальних. Сучасний світ неможливо представити без використання комп'ютерних мереж, які у тому або іншому вигляді застосовуються усюди, починаючи від елементарних хатніх комп'ютерних мереж і закінчуючи промисловими мережами, глобальними мережами, банківськими мережами, мережами для проведення наукових досліджень.

Актуальність теми магістерського дослідження обумовлена необхідністю підвищення ймовірності достовірного надання даних, які передаються по комп'ютерним мережам. Ці задачі є одними із ключових при розробці сучасних систем моніторингу локальних мереж.

Терміном **моніторинг мережі** називають роботу системи, що виконує постійне спостереження за комп'ютерною мережею в пошуках повільних або несправних систем і яка при виявленні збоїв повідомляє про їх мережному адміністраторові за допомогою пошти, телефону або інших засобів оповіщення. Ці задачі є підмножиною задач керування мережею.

У той час, як система виявлення вторгнень стежить за появою погроз ззовні, система моніторингу мережі виконує спостереження за мережею в пошуках проблем, викликаних перевантаженими й/або серверами, що відмовили, іншими пристроями або мережними з'єднаннями.

Наприклад, для того, щоб визначити стан веб-серверу, програма, що виконує моніторинг, може періодично відправляти запит HTTP на одержання сторінки; для поштових серверів можна відправити тестове повідомлення по SMTP і одержати по IMAP або POP3.

Невдалі запити (наприклад, у тому випадку, коли з'єднання не може бути встановлено, воно завершується по таймауту, або коли повідомлення не було доставлено) звичайно викликають реакцію з боку системи моніторингу.

Як реакція може бути:

– відправлено сигнал тривоги системному адміністраторові;

– автоматично активована система захисту від збоїв, що тимчасово виведе проблемний сервер з експлуатації, доти, поки проблема не буде вирішена, і так далі.

Проведений критичний аналіз існуючих методів обробки даних, дозволив структурувати область застосування таких технологій і виявити ряд обмежень. У зв'язку із цим виникла необхідність у розробці перспективного, однак ще недостатньо дослідженого теоретично й апробованого на практиці, динамічного багатопоточного методу обробки даних і програмного забезпечення для реалізації цього підходу.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп’ютерних систем.

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

– Огляд існуючих систем моніторингу LAN мереж інформаційних та комп’ютерних систем.

– Дослідження системи моніторингу LAN мереж інформаційних та комп’ютерних систем.

– Програмна реалізація системи моніторингу LAN мереж інформаційних та комп’ютерних систем.

Об’єктом дослідження є процес моніторингу LAN мереж інформаційних та комп’ютерних систем.

Предметом дослідження є методи моніторингу LAN мереж інформаційних та комп’ютерних систем.

Методи дослідження базуються на методах побудови мереж інформаційних та комп'ютерних систем, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу LAN мереж інформаційних та комп'ютерних систем.

– Розроблено вітчизняний продукт моніторингу LAN мереж інформаційних та комп'ютерних систем, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі моніторингу LAN мереж інформаційних та комп'ютерних систем.

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 LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем, 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

Постійний контроль функціонування локальної мережі необхідний для підтримки її в працездатному стані, а також запобігання несанкціонованих змін у складі кабельної системи й конфігурації робочих станцій.

Очевидно, куди простіше попередити неполадки в мережі, ніж виправляти вже виниклі проблеми. Моніторинг серверів, виконуваних на них додатків і мережних пристроїв допоможе завчасно довідатися про потенційні несправності й запобігти їх наслідкам. Відслідковуючи функціонування мережі й зберігаючи передісторію її роботи, адміністратор до того ж може надати точну інформацію користувачам, у яких іноді складається невірне подання про частоту появи різних несправностей. Не менш важливо, що мережний моніторинг дозволяє одержувати точні відомості про події в мережі, а також часу й джерелах звернень до мережі. Отже, існує два типи моніторингу. Перший з них ми будемо називати оперативним моніторингом (operations monitoring), а другий – моніторингом безпеки (security monitoring).

Великі підприємства іноді ділять ці два типи моніторингу на два окремих процеси, виконувані співробітниками виробничих підрозділів і IT-безпеки, але малі й середні компанії з ряду причин частіше організують загальний процес моніторингу. Незалежно від розміру бюджету й числа співробітників, мережі малих і середніх компаній звичайно не мають потреби в такому ж рівні поточного оперативного моніторингу, як у великих корпораціях. Мережі малих підприємств завантажені не настільки інтенсивно, як корпоративні, і обслуговувати їх не так складно. Крім того, технічні проекти малих компаній простіше, і вони не мають потреби в детальному аналізі тенденцій і звітах, необхідних в установах з більш повільними процедурами прийняття рішень.

Контроль – це необхідний перший етап, що повинен виконуватися при керуванні мережею. Через важливість цієї функції її часто відокремлюють від інших функцій систем керування й реалізують спеціальними засобами. Такий поділ функцій контролю й властиво керування корисно для великих і середніх мереж, для яких установка інтегрованої системи керування економічно недоцільна. Використання автономних засобів контролю допомагає адміністраторові мережі виявити проблемні ділянки й пристрої мережі, а їхнє відключення або реконфігурацію він може виконувати в цьому випадку вручну.

Процес контролю роботи мережі звичайно ділять на два етапи – опитування параметрів і аналіз.

На першому етапі виконується більше проста процедура – процедура збору первинних даних про роботу мережі й пристрої, підключених до неї: програмних і апаратних характеристик робочих станцій, працездатність мережних пристроїв, кінцевого устаткування, мережний трафік й т.і.

Далі виконується етап аналізу, під яким розуміється більш складний і інтелектуальний процес осмислення зібраної на етапі опитування параметрів інформації, зіставлення її з даними, отриманими раніше, і вироблення припущень про можливі причини вповільненої або ненадійної роботи мережі.

Задачі опитування параметрів вирішуються програмними й апаратними вимірниками, тестерами, мережними аналізаторами, вбудованими засобами опитування комунікаційних пристроїв, а також агентами систем керування. Задача аналізу вимагає більше активної участі людини й використання таких складних засобів, як експертні системи, що акумулюють практичний досвід багатьох мережних фахівців.

1.2 Область застосування

Однією з основних проблем, що стоять зараз перед розроблювачами систем керування комп'ютерними мережами є проблема достовірного надання

даних про їхній стан. Потреба в надійно працюючих великих комп'ютерних мережах усе вище з кожним днем. Тому при проектуванні сучасних систем керування дуже важливу роль відводять розробкам оптимізованих за часом алгоритмам збору й обробки даних.

Об'єкти моніторингу

Дані, що надходять від контрольованих пристроїв, називаються телеметричними. Які протоколи й формати даних звичайно використовуються для передачі телеметрії? Як відслідковувати різні джерела даних і генерувати попередження й звіти, щоб перетворити дані в інформацію? Одна з найважливіших задач – вирішити, на підставі яких даних необхідно генерувати попередження в реальному часі, які дані варто вносити в щоденні й щотижневі звіти і які дані потрібно просто архівувати.

З метою безпеки корисно контролювати всі мережні пристрої (наприклад, брандмауери, шлюзи, VPN-пристрої, бездротові вузли доступу – AP) на границі мережі – периметрі безпеки, а також будь-які сервери, на яких розміщаються інформація й процеси, що вимагають конфіденційності або цілісності. Для виробничих цілей варто контролювати всі пристрої й сервери, відказостійкість яких важлива для нормальної роботи підприємства. Для Windows необхідний моніторинг не тільки операційної системи, але й важливих додатків, таких як Microsoft Exchange Server, Microsoft ISA Server, Microsoft IIS і Microsoft SQL Server. Корисно контролювати високорівневі додатки (наприклад, Microsoft SharePoint Portal Server), якщо в такий спосіб вдається виявити важливі події, що відносяться до сфери безпеки або виробництва, які можуть залишитися непоміченими засобами нижчележачих баз даних, на яких вони працюють.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп’ютерних систем, 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

Існує чимало способів контролю роботи мережі – починаючи від використання діагностичних засобів операційних систем, наприклад убудованого в операційні системи сімейства Windows набору утиліт моніторингу Monitoring performance, що дозволяє досить детально оцінити роботу окремого комп'ютера, у тому числі його мережного інтерфейсу, і закінчуючи спеціалізованими апаратними комплексами. Але на практиці ні убудовані засоби ОС, ні апаратні засоби моніторингу найчастіше не задовольняють потреби й можливості системного адміністратора по організації повноцінного моніторингу мережі. У першому випадку можна проаналізувати лише роботу окремого вузла, а не мережі в цілому, внаслідок чого одержати цілісну картину стану мережі без наявності певного досвіду й точного знання структури конкретної ЛОМ і точного розуміння принципів його роботи стає практично неможливо. Другий же підхід вимагає значних фінансових витрат, і не кожна організація готова піти на це, тим більше що в розумінні багатьох керівників витрачати гроші на нормально працюючу мережу – занадто більша розкіш. У подібній ситуації свого роду панацеєю стають утиліти мережного моніторингу, що дозволяють проводити аналіз мережного трафіку як окремих вузлів, так і мережі в цілому, а найбільш просунуті з них аналізують і топологію мережі, щоб виявити конфлікти, що виникають у результаті помилок, допущених при побудові ЛОМ. У цей час є величезну кількість утиліт, що розрізняються як по своїх можливостях, так і по доступності й зручності інтерфейсу. Із цієї причини при відборі утиліт моніторингу мережі для даного огляду віддана перевага програмним продуктам,

які дозволяють найбільше повно аналізувати структуру й завантаження мережі й володіють, максимально дружнім, інтуїтивно зрозумілим інтерфейсом.

Essential NetTools

Утиліта Essential NetTools компанії TamoSoft, Inc. – це зручний і простій у використанні інструмент, що дозволяє робити не тільки всебічний аналіз роботи окремого комп'ютера, на якому встановлена дана програма, але й досить детальний аналіз стану й роботи мережних інтерфейсів всіх хостів ЛОМ. Програма має традиційний Windows-інтерфейс. Запуск основних інструментів моніторингу може здійснюватися як за допомогою пунктів основного меню, так і за допомогою кнопок швидкого запуску, розташованих ліворуч від основної робочої області вікна програми, у якій і відображається обраний у цей час інструмент.

Essential NetTools надає в розпорядження користувача наступні інструменти (можливості) моніторингу, запуск яких, як уже говорилося, здійснюється по натисканні відповідної кнопки:

– NetStat – інструмент, аналогічний однойменній утиліті Windows, що запускається з командного рядка, відображає список з'єднань, ставлячи їм у відповідність відкриті порти TCP і UDP і додатка, що ініціюють ці з'єднання;

– NBScan – швидкий і потужний інструмент для дослідження мережі в заданому діапазоні IP-адрес із використанням можливостей протоколу NetBIOS. На відміну від Windows-команди nbtstat, цей інструмент має наочний графічний інтерфейс і здійснює значно більше швидке паралельне сканування;

– PortScan – інструмент для сканування портів TCP, що дозволяє сканувати порти віддаленого хосту як у звичайному, так і в невидимому (Stealth) або напіввідчиненому (Half-Open) режимі, при якому TCP-з'єднання ініціюється, але не завершується, у результаті чого таке сканування залишається непомітним для абонента (якщо, звичайно, на його комп'ютері на встановлені спеціальні засоби захисту). За допомогою цього інструмента можна оцінити захист портів

комп'ютера й виявити можливість коректної роботи тих або інших сервісів, що використовують з'єднання по певних портах;

– Shares – надає можливість контролювати підключення до поділюваного (загальним) ресурсам на комп'ютері, де встановлена дана утиліта, переглядати список поділюваних ресурсів віддаленого хосту й одержувати доступ до поділюваних ресурсів інших вузлів мережі по логіну й паролю;

– SysFiles – призначений для редагування п'яти системних файлів: services, protocol, networks, hosts і lmhosts. Відзначимо, що при використанні цього інструмента потрібно добре уявляти собі можливі наслідки внесених змін;

– NetAudit – служить для проведення аудита вузлів мережі в заданому IP-діапазоні на предмет доступу до їхніх ресурсів. Для цього використовується список можливих логінів і паролів (використання свого списку можливо тільки в ліцензійній версії продукту), що дозволяє уникнути можливого несанкціонованого доступу до мережних каталогів хостів, що перевіряються;

– RowSocket – дозволяє відправляти й приймати дані на вузли із заданим IP-адресою (або ім'ям) і від них, прослуховуючи при цьому обраний TCP– і UDP-порт локального (тобто свого) хосту;

– TraceRoute – даний інструмент по своїй суті є інтерпретацією добре знайомій команди Tracert, що дозволяє переглянути шлях маршрутизації до шуканого хосту. Основною відмінністю TraceRoute від цієї команди є наявність зручного графічного інтерфейсу;

– Ping – аналогічний однойменній команді Ping, але, як і в попередньому випадку, графічний інтерфейс робить процедуру пінгу віддаленого вузла (і навіть групи вузлів) гранично зручною;

– NSLookup – по своїх можливостях також дуже нагадує однойменну Windows-утиліту. Однак NSLookup дозволяє виконувати як звичайні запити (Standard Resolve Function) до відомих операційній системі DNS-серверам, так і специфічні (тип яких можна вибрати зі списку, що розкривається); при цьому можливо задавати як сервер будь-який доступний DNS-сервер;

– ProcMon – дозволяє одержувати інформацію про те, які процеси запущені на комп'ютері.

Утиліта Essential NetTools за допомогою пункту меню «Загальна інформація про систему» (System Summary), що дублюється однойменною кнопкою на панелі інструментів, дає можливість одержувати вичерпну інформацію про комп'ютер – починаючи від його імені й закінчуючи завантаженням процесора й оперативної пам'яті. Крім того, ця утиліта від компанії TamoSoft надає швидкий доступ до основних інструментів адміністрування й настроювання системи, дозволяючи також задавати гарячі клавіші для швидкого запуску додатків.

LANScan

Сімейство утиліт мережного моніторингу LANScan, створене компанією LANScan Software, Inc., дозволяє вирішувати наступні задачі:

– контроль маршрутизації;

– оцінка використання смуги пропущення мережі;

– контроль доступу в Інтернет;

– моніторинг додатків, що використовують мережні ресурси;

– локалізація й виявлення апаратних несправностей компонентів мережі;

– виявлення несправностей, викликаних неправильним конфігуруванням програмних засобів – у тому числі невірного настроювання драйверів мережних пристроїв, некоректного призначення мережних адрес і т.п.;

– вимір мережного трафіку, створюваного додатками;

– ведення запису звіту про роботу мережі;

– віддалений контроль роботи мережі.

Сімейство LANScan включає три утиліти: LANScan Traffic View, LANScan Network Monitor і LANScan Professional, які, у загальному-те, являють собою версії того самого програмного продукту й розрізняються лише набором функціональних можливостей. Так, утиліта LANScan Professional надає користувачеві самий широкий вибір інструментів і можливостей, у той час як

LANScan Network Monitor має трохи менші можливості (відсутня можливість ведення бази даних по мережному трафіку й зборі даних про мережні пакети), а LANScan Traffic View і зовсім є урізаною версією, позбавленої багатьох можливостей. Дана утиліта не може, по-перше, працювати із пристроями (вузлами) мережі як з об'єктами моніторингу (тобто відображати статистику по трафіку з фільтрацією по адресах пристроїв, здійснювати моніторинг високого або низького рівня трафіку мережних вузлів, робити конфігурування вузлів мережі як об'єктів категорії «Device», використовувати фільтри для відображуваних пристроїв, вести звіт про активність пристроїв по окремих протоколах і сервісам: (MAC, IP, IPX або DNS); по-друге, виконувати Ping і Tracert за встановленим розкладом; по-третє, відправляти результати моніторингу відповідно до встановленої схеми (за розкладом) на печатку, на задану адресу електронної пошти або зберігати у файл; по-четверте, вести log-файл по мережному трафіку.

Всі три утиліти мають схожі інтерфейси, тому розглянемо базові можливості, надавані утилітами LANScan, на прикладі версії LANScan Network Monitor. Робоча область утиліти розділена на три вікна: у нижньому відображаються фіксуємі події, середнє являє собою основне робоче вікно, що відображає поточний інструмент моніторингу, а у верхнім вікні відбувається графічне подання даних, відображуваних в основному робочому вікні.

Основне робоче вікно має п'ять закладок-перемикачів, за допомогою яких здійснюється перехід між основними режимами моніторингу, а утиліти LANScan дозволяють відображати наступну інформацію:

– Summary (підсумок) – показує основні параметри мережі, якось: завантаження мережі (кількість переданих байтів і пакетів) у встановлені проміжки часу (за останні 10 секунд, за останні 5 мінут, за 1 годину, за 8 годин); вузли мережі, відсортовані за рівнем мережної активності; мережні сервіси й протоколи, відсортовані за рівнем утилізації мережі;

– Trend (тенденції) – повідомляє про рівень трафіку в масштабі реального часу, відображаючи у верхнім вікні графік завантаження мережі, обмірюваний у пакетах або байтах у секунду. Частота вибірки значень для графіка визначається користувачем у діапазоні від 1 секунди до декількох годин (за замовчуванням 10 секунд). В основному вікні можна оцінити рівень трафіку, створюваного в мережі окремими вузлами, а також проаналізувати, пакети якого розміру найбільше часто використовуються вузлами мережі для спілкування;

– Conversation (сеанс зв'язку) – відображає всі активні в цей момент мережні пристрої й рівень створюваного ними трафіку (у байтах або пакетах) у сучасний момент і за увесь час роботи, а також час останньої мережної активності. Вся інформація про мережний трафік вузлів мережі також представляється в графічному виді, але, на жаль, при великій кількості вузлів відбувається повторення квітів, які в принципі повинні бути ексклюзивними для кожного хосту (використовується всього 16 кольорів). Даний пункт дає можливість вибирати відображувана назва вузла мережі, у якості якого можна взяти MAC-адресу мережної карти, її IP-адресу, ім'я DNS або ім'я, асоційоване з даним пристроєм і задане користувачем утиліти. Крім того, за допомогою контекстного меню, викликуваного натисканням правої клавіші миші, може бути виконаний тест на активність (присутність у мережі) мережного пристрою (пункт меню ActiveTest) і перевірений шлях маршрутизації до цього вузла мережі (пункт меню Trace);

– Protocols (протоколи) – відображає розподіл мережного трафіку відповідно до мережних протоколів. Ця інформація має як чисельне (у байтах або пакетах), так і графічне подання. При цьому, як і в попередньому випадку, відображається розподіл трафіку й на сучасний момент, і за увесь час спостереження;

– Layout (схема) – дозволяє будувати структурну схему мережі. Значки зконфігурованих пристроїв автоматично з'являються на аркуші схеми. Після

завдання режиму редагування можна переміщати ці рисунки, з'єднувати вузли в потрібному порядку, вишиковуючи в такий спосіб топологію мережі;

У версіях LANScan Network Monitor і LANScan Professional є можливість оперувати як об'єкт моніторингу не тільки мережним інтерфейсом комп'ютера, на якому встановлена дана утиліта (об'єкт Network), але й будь-якими мережними пристроями мережі (об'єкти Device). Це означає, що користувачеві надається можливість віртуально перенестися на обраний вузол мережі й проводити моніторинг таким чином, начебто дана утиліта працює на віддаленій машині. При цьому стають доступні ті ж інструменти мережного моніторингу, що й описані вище для локального мережного інтерфейсу, з тією лише різницею, що аналізуються тільки вхідний і вихідний трафіки цього вузла. Об'єкти Device можуть бути створені за допомогою пункту контекстного меню Create LANScan Device на аркуші Conversation (конкретний обраний вузол) або Auto Discovery у контекстному меню об'єктів Device: у діалоговому вікні, що з'являється, задається діапазон IP-адрес для сканування, і кожне зі знайдених пристроїв може бути додане в список об'єктів Device так само, як і з вікна Conversation.

При додаванні об'єктів Device можна задавати вид відображення іконки з ряду представлених варіантів. Швидкий перехід до моніторингу обраного у вкладці Conversation вузла можна здійснити по пункті контекстного меню Quick Device View. Для всіх об'єктів утиліт LANScan Network Monitor і LANScan Professional може бути задана функція оповіщення про події, що стосуються рівня трафіку вузла (low, high), по досягненні якого користувач буде попереджуватися за допомогою повідомлення відправленого по e-mail, спливаючого вікна й звукового сигналу. Крім того, це буде відбито відповідним записом в log-файлі й зміною кольору індикатора, що перебуває напроти об'єкта (який, до речі кажучи, з'являється тільки після призначення моніторингу подій для об'єкта).

3Com Network Supervisor

Утиліта 3Com Network Supervisor – це, мабуть, самий потужний і зручний інструмент мережного моніторингу з усіх, що нам довелося переглянути при

підготовці огляду. Даний програмний продукт сполучить у собі простоту установки й настроювання, максимальну інформативність і зручний, інтуїтивно зрозумілий інтерфейс. Хоча компанія 3Com, що є одним з лідерів по виробництву мережного встаткування, випустила цю утиліту насамперед для мереж, побудованих на основі компонентів її власного виробництва, однак цей програмний продукт може успішно працювати й з мережним устаткуванням інших виробників. Головним достоїнством утиліти 3Com Network Supervisor, на наш погляд, є її здатність відображати структуру мережі за результатами її автоматичного сканування, що полегшує роботу мережних адміністраторів великих ЛОМ, а якщо мережа невелика, то може сприяти більше детальному аналізу її структури.

Слід зазначити, що при скануванні локальної мережі програма виявляє різного роду конфлікти й помилки в структурі мережі, про що повідомляє користувача. За допомогою даної утиліти можна здійснювати моніторинг як стану ЛОМ у цілому, так і кожного окремого вузла мережі, оперативно локалізуючи й виявляючи перевантажені ділянки мережі й причину виникнення цих стресових ситуацій. Програмний продукт 3Com Network Supervisor надає зручні кошти для оцінки роботи основних мережних сервісів і протоколів кожного вузла ЛОМ (хоча тут варто обмовитися, що аналізуються протоколи, властиві Windows-мережам), даючи адміністраторам можливість вибирати склад сервісів і протоколів, по яких буде проводитися моніторинг.

Про виникнення стресової ситуації утиліта попереджає користувача звуковим сигналом і зміною кольору індикатора відповідного мережного вузла. При цьому існує можливість гнучко настроїти граничне значення стану стресу, тобто вибрати період часу, протягом якого мережний вузол перебуває в тім або іншому критичному стані, щоб уникнути трактування короткочасних збоїв (або іншої події) як стресового стану вузла.

До стресових станів вузла ставляться перевищення певного рівня трафіку й різного роду некоректності в роботі з того або іншого протоколу, причому

утиліта 3Com Network Supervisor указує можливі причини виникнення подібної ситуації. Цікаво, що при наявності у вузла працюючого Web-сервера можна одержати до нього доступ по протоколу http, зробивши подвійний клік на іконці цього вузла. Також слід зазначити, що програма веде журнал статистики (вхід через основне меню або через кнопку All Events на панелі інструментів), де відображаються всі важливі (стресові) події, що відбулися за час спостереження за мережею (із вказівкою дати й часу), а також вказується, чи розв'язалася дана ситуація для вузла або він продовжує працювати в стресовому режимі.

Програма ping

ping – це службова комп'ютерна команда, призначена для перевірки з'єднань у мережах на основі TCP/IP.

Вона відправляє запити Echo-Request протоколу ICMP зазначеному вузлу мережі й фіксує вступників відповіді (ICMP Echo-Reply). Час між відправленням запиту й одержанням відповіді (RTT, від *Round Trip Time*) дозволяє визначати двосторонні затримки (RTT) по маршруті й частоту втрати пакетів, тобто побічно визначати завантаженість на каналах передачі даних і проміжних пристроїв.

Також пінгом називається час, витрачене на передачу пакета інформації в комп'ютерних мережах від клієнта до сервера й обернено від сервера до клієнта, воно виміряється в мілісекундах. Час пінга зв'язаний зі швидкістю з'єднання й завантаженістю каналів на всьому протязі від клієнта до сервера.

Повна відсутність ICMP-відповідей може також означати, що віддалений вузол (або який-небудь із проміжних маршрутизаторів) блокує ICMP Echo-Reply або ігнорує ICMP Echo-Request.

Програма ping є одним з основних діагностичних засобів у мережах TCP/IP і входить у поставку всіх сучасних мережних операційних систем. Функціональність ping також реалізована в деяких вбудованих ОС маршрутизаторів, доступ до результатів виконання ping для таких пристроїв по протоколі SNMP визначається RFC 2925 (Definitions of Managed Objects for Remote Ping, Traceroute, and Lookup Operations).

Так як для відправлення ICMP-пакетів потрібно створювати raw-сокети, для виконання програми ping в unix-системах необхідні права суперкористувача. Щоб звичайні користувачі могли використовувати ping у правах доступу файлу /bin/ping установлюють SUID-біт.

Програма ipconfig

ipconfig – утиліта командного рядка для керування мережними інтерфейсами.

В операційних системах Microsoft Windows і Windows NT, ipconfig – це утиліта командного рядка для висновку деталей поточного з'єднання й контролю над клієнтським сервісом DHCP. Також є подібні графічні утиліти з назвами winipcfg і wntipcfg (останній передував ipconfig).

Часто в операційних системах GNU/Linux і UNIX деталі з'єднання відслідковуються декількома утилітами, головної серед них є ifconfig. Проте, ipconfig поряд з ifconfig є присутнім в Mac OS X, там ipconfig команда сервісу як оболонка до IPConfiguration агента й може використовуватися для контролю Boot і DHCP клієнта з CLI.

Сервери SNMP

SNMP – це технологія, покликана забезпечити керування й контроль за пристроями й додатками в мережі зв'язку шляхом обміну керуючою інформацією між агентами, що розташовуються на мережних пристроях, і менеджерами, розташованими на станціях керування. SNMP визначає мережу як сукупність мережних керуючих станцій і елементів мережі (головні машини, шлюзи й маршрутизатори, термінальні сервери), які спільно забезпечують адміністративні зв'язки між мережними керуючими станціями й мережними агентами.

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

Керуюча система може одержати інформацію через операції протоколу GET, GETNEXT і GETBULK. Агент може самостійно без запиту відправити дані, використовуючи операцію протоколу TRAP або INFORM. Керуючі системи можуть також відправляти конфігураційні відновлення або контролюючі запити, використовуючи операцію SET для безпосереднього керування системою. Операції конфігурування й керування використовуються тільки тоді, коли потрібні зміни в мережній інфраструктурі. Операції моніторингу звичайно виконуються на регулярній основі. Змінні доступні через SNMP організовані в ієрархії. Ці ієрархії й інші метадані (такі як тип і опис змінної) описуються Базами Керуючої Інформації (Management Information Bases (MIBs)). SNMP не визначає, яку інформацію (які змінні) керована система повинна надавати. Навпаки, SNMP використовує розширювану модель, у якій доступна інформація визначається Базами Керуючої Інформації (MIB – Management Information Base). Бази Керуючої Інформації описують структуру керуючої інформації пристроїв. Вони використовують ієрархічний простір імен, що містить унікальний ідентифікатор об'єкта (objectidentifier(OID)).Грубо кажучи, кожний унікальний ідентифікатор об'єкта ідентифікує змінну, котра може бути прочитана або встановлена через SNMP. MIBи використовують нотацію, задану в ASN.1.

Ієрархія MIB може бути зображена як дерево з безіменним коренем, рівні якого привласнені різними організаціями. На найвищому рівні MIB OIDи належать різним організаціям, що займаються стандартизацією, у той час як на більше низькому рівні OIDи виділяються асоційованими організаціями. Ця модель забезпечує керування на всіх шарах мережної моделі OSI, так як MIBи можуть бути визначені для будь-яких типів даних і операцій.

Керований об'єкт – це одна з будь-якого числа характеристик, специфічних для керованого пристрою. Керований об'єкт містить у собі один або більше екземплярів об'єкта (ідентифікуємих по OID), які насправді змінні.

Існує два типи керованих об'єктів:

1. Скалярні об'єкти визначають єдиний екземпляр об'єкта.

2. Табличні об'єкти визначають множинні, зв'язані екземпляри об'єктів які групуються в таблицях MIB.

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

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

Крім перерахованих вище програмних продуктів, моніторинг мережі можна проводити й за допомогою наступних програм:

– Cisco Works NMS;

– Hyperic HQ (Open Source);

– Zabbix (Open Source);

– NetXMS (Open Source);

– TclMon (Open Source);

– Big Brother;

– Caligare Flow Inspector – Аналізатор NetFlow;

– MRTG;

– RRDtool;

– Intellipool Network Monitor;

– Ipswitch WhatsUp;

– ManageEngine OpManager;

– Netmon – Appliance based network monitoring suite with email and pager alert system;

– Nagios (раніше Netsaint);

– Cricket;

– PRTG;

– Packet Analyzer: Network Traffic Monitoring, Analysis and Troubleshooting;

– NetVizor;

– NetDecision;

– HP OpenView Network Node Manager (NNM);

– ProLAN-експерт;

– Total Network Monitor;

– Monit (Open Source);

Представлені в цьому розділі утиліти моніторингу (як і багато інших подібних програмних продуктів, що не ввійшли в наш огляд), звичайно ж, могли б у чималому ступені полегшити життя людям, яким по роду їхніх занять доводиться створювати й адмініструвати локальні обчислювальні мережі, але тут з'являється одна обставина, здатна внести значні корективи в плани використання подібних програм. Що ж це за обставина? Вся справа в принципі роботи таких програм, що заснований на тім, що мережна карта фіксує всі мережні пакети – як адресовані даному вузлу мережі, так і не призначені для нього. У такій ситуації можливості аналізу мережного трафіку попадають у залежність від компонентів, на основі яких побудована ваша мережа (тут говориться про Ethernet-мережі). Якщо як основу для побудови мережі використовуються хаби (що, на жаль, малоймовірно), то проблем не буде й ви одержите все різноманіття описаних можливостей. У випадку ж застосування комутаторів дані утиліти дозволяють оцінювати трафік, створюваний тільки в одному сегменті мережі «комутатор-вузол», на якому й установлена дана утиліта. При цьому з'являється можливість вимірювати лише широкомовний трафік і трафік даного вузла. Проте й у таких ситуаціях цінність використання програм, подібних до утиліти мережного моніторингу 3Com Network Supervisor, залишається безперечним фактом, оскільки даний програмний продукт дає можливість досконального аналізу структури мережі й контролю за коректністю роботи мережних вузлів.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64-розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для

кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності.

Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windows або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв'язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв'язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, чим у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережний вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

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 моніторингу LAN мереж інформаційних та комп'ютерних систем.

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 обгрунтувати 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

Розробка системи моніторингу мережі з використанням стандартних інструментів

Джерела телеметричних даних

Головне джерело телеметричної інформації про сервери Windows – журнал подій Security, а найважливіше джерело виробничої телеметрії – журнали подій System і Application. Досвідчені користувачі оснащення Event Viewer консолі Microsoft Management Console (MMC) знають, що у всіх журналах подій Windows застосовується один формат файлів (.evt). Запис про кожну подію містить стандартні поля (наприклад, дата, час, джерело події, категорія, ID події), за якими знаходиться поле опису з даними у вільній формі, унікальними для конкретної події. Будь-який додаток моніторингу, сумісний з журналами подій Windows, дозволяє генерувати попередження й звіти на основі джерела, категорії або ID події, але в ідеальному випадку потрібно мати можливість відфільтровувати записи за даними в описі події.

Мережні пристрої, такі як маршрутизатори, комутатори, бездротові AP і брандмауери, незмінно передають телеметричні дані через протокол SNMP або Syslog. SNMP був спроектований наприкінці 1980-х для керування множиною пристроїв у мережі Internet, що бурхливо розвивається,. Диспетчери SNMP збирають телеметричну інформацію від агентів через UDP-порт 162. Диспетчери можуть використовувати SNMP-команди Get для запиту конкретних телеметричних даних, називаних змінними (variable), або пасивно чекати звіту про важливі події від агентів через повідомлення Trap. Для моніторингу в сферах виробництва й безпеки досить збирати повідомлення Trap. Для поглибленого

аналізу тенденцій і планування ресурсів варто опитувати агентів за допомогою команд Get.

Syslog

Syslog – стандарт протоколювання подій для UNIX. Перевага Syslog перед механізмом протоколювання подій Windows полягає в тому, що весь процес консолідації потоків подій від численних систем – невід'ємна частина Syslog. У дійсності Syslog одночасно мережний протокол і формат журналу, і за замовчуванням він використовує UDP-порт 514. Кожне повідомлення Syslog містить поля дати, часу, пріоритету, ім'я хосту й тексту повідомлення. З технічної точки зору пріоритет – число від 0 до 191. Однак більшість додатків Syslog відображають пріоритет у вигляді двох складових: Facility і Level.

Facility. Спочатку Syslog проектувався для моніторингу BSD Unix, і величина Facility використовувалася для ідентифікації процесу Unix про який свідчить подія. Значення від 0 до 15 відповідають найважливішим процесам Unix, а значення від 16 до 23 (з іменами від Local0 до Local7) призначені для додатків і пристроїв. Більшість мережних пристроїв використовують значення від Local0 до Local7 (наприклад, пристрої Cisco задіють Local6 і Local7), але не всі. Маршрутизатор Xincom Twin Wan використовує майже всі низькі значення Facility.

Level. Інший елемент пріоритету повідомлень Syslog – Level, значення якого перебувають у межах від 0 до 7. Level характеризує ступінь важливості повідомлення.

Продуктивність і стан

Для повного функціонального моніторингу корисно контролювати об'єкти продуктивності (performance-object) і стан серверів з окремого комп'ютера або від провайдеру послуг. Адміністратори, не знайомі з об'єктами продуктивності, можуть досліджувати їх за допомогою оснащення Performance консолі MMC. Різниця між моніторингом журналу подій і об'єкта продуктивності наступна: з журналів подій можна витягти інформацію про будь-яку частину системи, у якій

відбулися неполадки, а об'єкт продуктивності дозволяє переконатися, що конкретні параметри перебувають у припустимих межах. Наприклад, за допомогою об'єктів продуктивності можна стежити за простором жорсткого диска, так як системний журнал видає попередження, тільки коли тім заповнюється настільки, що користувач уже починає випробовувати незручності.

Ще одна типова перевірка із застосуванням об'єкта продуктивності – моніторинг коефіцієнта використання центрального процесора з відстеженням певних рівнів протягом тривалих періодів часу (наприклад, понад 90% протягом 10 хвилин). Однак при перевірках коефіцієнта використання центрального процесора варто проявляти обережність; неважко переплутати корисне навантаження з некерованим процесом і згенерувати помилкове повідомлення про проблему. Чудова властивість об'єктів продуктивності полягає в тому, що інші додатки можуть створювати власні об'єкти продуктивності й публікувати телеметричні дані, специфічні для даного додатка. Наприклад, Active Directory (AD), SQL Server і Exchange Server мають у своєму розпорядженні власні об'єкти продуктивності.

Відсутність повідомлень про помилки в журналі й показники продуктивності в межах припустимих порогів – гарні індикатори коректного функціонування. Однак деякі проблеми не знаходять відбиттів в індикаторах. Перевірки стану серверів – найефективніший спосіб переконатися в тому, що сервери й додатки працюють у мережі й успішно обробляють запити. Перевірки стану серверів надійні, так як вони виконують тестову транзакцію. Багато провайдерів додатків і служб в Internet дозволяють регулярно проводити тестові транзакції із сервером через обрані споживачем інтервали часу. Для Web-серверу можна періодично запитувати дану Web-сторінку й перевіряти, чи успішно вона передана. Для системи SQL Server можна періодично виконувати запит і перевіряти результати.

Однак навіть перевірки стану не розкривають всіх проблем. Наприклад, простий сигнал ping, переданий через кожні 5 мінут, дозволяє переконатися, що

операційна система й набір протоколів активні, але не містять ніякої інформації про стан самого додатка. Мені доводилося зустрічати завислі сервери, які відповідали на сигнали ping. Аналогічно простий запит HTML-сторінки із сервера не доводить, що відповідний додаток електронної комерції на базі Active Server Pages (ASP) працює коректно.

Тому перевірки стану повинні бути як можна більш функціональними. Ще одне застереження: програму перевірки стану варто розміщати поза контрольованим виробничим середовищем. Якщо помилково розмістити програму перевірки стану на контрольованому сервері, то, наприклад, не вдасться визначити відмову сервера або серверного з'єднання, так як додаток не зможе передати адміністраторові відповідне повідомлення. Але якщо додаток перевірки працює на окремому сервері (і якщо цей сервер доступний з Internet), то єдиний випадок, коли важливий додаток буде не готовий до роботи без відома адміністратора, – одночасна відмова виробничого й контролюючого середовища.

Необхідний інструментарій

Отже, що потрібно для моніторингу всіх пристроїв, серверів, журналів, пасток SNMP і подій Syslog? Очевидно, необхідні один-два інструмента за доступною ціною, що охоплюють всі елементи, які потрібно відслідковувати. Продукти моніторингу високого рівня, такі як Argent Guardian і Microsoft Operations Manager (MOM), дозволяють контролювати всі об'єкти продуктивності, журнали подій Windows, пастки SNMP, потоки подій Syslog і навіть виконувати різні перевірки стану. Деякі не настільки великі, менш дорогі пакети, такі як Sentry II компанії Engagent, EventTracker компанії Prism Microsystems і комплекс Event Log Management компанії Dorian, охоплюють підмножину телеметричних джерел і обмежений набір об'єктів продуктивності.

Збираючись придбати інструмент, варто скласти список всіх характеристик, які необхідно знати, і підібрати інструмент, що контролює їх всі. Якщо інструмент не забезпечує моніторинг важливого параметра, наприклад SNMP, то заповнити пробіл можна за допомогою безкоштовної або недорогої

умовно безкоштовної утиліти. Далі буде розглянутий ряд таких інструментів, з яких можна скласти ефективний комплекс моніторингу.

Розглянемо три корисних інструменти, які можна додати до арсеналу мережного моніторингу: Log Parser, безкоштовний інструмент Microsoft; tail, відмінну утиліту з миру UNIX; утиліту Kiwi Syslog Daemon, що представлена безкоштовною й могутнішою, але проте недорогою версією. Інформація буде корисна навіть тим адміністраторам, які вже мають або мають намір придбати інструменти: більшість інструментів на ринку розташовують лише функціями попереджень і звітності, доповненими шаблоновими звітами й зразковими правилами розсилання попереджень. Методи проектування й аналізу, описані в розділі, будуть надзвичайно корисні навіть для власників розгорнутої на підприємстві програми моніторингу.

Моніторинг журналу Security Log

Адміністраторам, які мають потребу в інструменті для моніторингу журналів подій Windows і розсилання оповіщень (по електронній пошті або на телефон) при виявленні подій, що відповідають заданим критеріям, я рекомендую створити спеціалізований механізм попереджень із використанням VBScript і Windows Management Instrumentation (WMI). Раніше в журналі ми вже приводили приклади сценаріїв для моніторингу певних подій і пересилання поштового повідомлення на зазначену адресу при кожній такій події. Сценарій не витрачає ресурси центрального процесора під час очікування події, і адміністратори може легко змінити WMI-запит, що описує події, оброблювані сценарієм. В WMI-запитах використовується команда SQL Select, аналогічна командам, з яких побудовані запити Microsoft Access. Проблема полягає у відборі повідомлень, які повинні генерувати попередження. Якщо не дотримувати обережності, телефон або вхідна поштова скринька будуть переповнені не занадто важливими повідомленнями.

Приступаючи до підготовки критерію для відправлення попереджень, необхідно використовувати інструмент для збору інформації з журналів подій.

Кращий інструмент для опитування журналів подій – Log Parser компанії Microsoft. Log Parser дозволяє направляти запити до журналу подій з використанням оператора Select, схожого на запит WMI, але більш зручного для опитування існуючих журналів. Задача адміністратора – скласти один або кілька запитів для журналу подій кожного типу (наприклад, Application, Security, System), щоб відфільтрувати незначні події, але повідомляти про надзвичайні події. Я рекомендую витягати з журналу Security найбільш типові й важливі події. Зразковий вид запиту:

```
logparser "select
 TimeGenerated,EventID,Message
from \\mtg1\security
where EventID in
        (675;676;681;642;624;644;617;632;660;636)"
```

Моніторинг інших журналів

Для інших журналів подій (наприклад, Application, System) я рекомендую складати звіт, починаючи з пошуку попереджень і помилок, ігноруючи інформаційні повідомлення. І навіть після цього залишається багато незначних попереджень і повідомлень про помилки. Наприклад, я одержую багато помилок із джерела MRxSmb, який можна сміло ігнорувати. Тому наступний крок – ідентифікувати події, які являють собою «шум», і виключити їх за допомогою пропозиції Where у запиті Log Parser. У наступну команду додане вираження, що виключає подію ID 3019 для джерела MRxSmb і інформаційні повідомлення з будь-якого іншого джерела (EventType < 4):

```
logparser "select
 TimeGenerated,EventID,Message
from system where
 EventID<>3019 and SourceName
 <>'MRxSmb' and EventType < 4"
```

Варто пам'ятати, що журнали подій досягають дуже великих розмірів. Крім того, EVT-файли не мають індексації, що забезпечує швидкий пошук інформації без переглядів усього журналу. На підготовку звіту Log Parser може піти чимало часу, так як програмі щораз доводиться переглядати весь журнал

подій. На щастя, завдяки контрольним крапкам Log Parser пам'ятає, де закінчився минулий перегляд.

Задача моніторингу журналів подій Windows ускладнюється через те, що в кожного комп'ютера є власний набір журналів і не існує природного способу зібрати їх воєдино. Якщо немає інструмента, що поєднував би журнали в одній центральній базі даних для розсилання попереджень і моніторингу, існує два варіанти рішення проблеми. Якщо має бути управляти лише нечисленними серверами й пристроями, то можна просто призначити попередження й звіти для кожної машини. Більше централізований підхід – використовувати можливість Log Parser опитувати кілька комп'ютерів. Необхідно лише перелічити журнали подій кожного комп'ютера в пропозиції From. Наприклад, що випливає команда опитує журнал System на машинах server1, server2 і server3.

```
logparser "select
 TimeGenerated,EventID,Message
from \\server1\system,
 \\server2\system,
 \\server3\system"
```

Після того як будуть складені один або кілька звітів для журналу кожного типу, можна готовити їх щодня або щотижня за розкладом. Досить перенаправляти вихід Log Parser у текстовий файл, додавши до наведеного вище команду: «> C:\path\file.txt» а потім регулярно переглядати цей файл. Ще краще пересилати отриманий файл по електронній пошті адміністраторові. Для цього варто додати один рядок у командний файл, для виклику Blat. За допомогою загальнодоступної утиліти Blat зручно пересилати файли по електронній пошті через SMTP.

Настроївши звіти, можна приступати до проектування критеріїв попередження. Було б дуже добре, якби всі постачальники додатків і пристроїв документували події й повідомлення, генеруємі їхніми продуктами. У такому випадку адміністратори могли б зробити усвідомлений вибір, але в житті нічого не дається легко, тому доводиться діяти «на дотик». Вибір критерію нагадує процес складання звітів, описаний вище. Однак варто бути більше розбірливим

при виборі попереджень, що посилаються на телефон або у свою поштову скриньку. У більшості програм попередження можна будувати як фільтри Exclude, Include, а потім застосовувати їх послідовно, видаляючи дрібні події. Якщо використовується WMI, то можна задіяти речення Where для фільтрації більшості непотрібних подій. Якщо можливостей речення Where виявляється недостатньо, можна побудувати додатковий фільтр із застосуванням сценарію VBScript.

Я рекомендую використовувати Log Parser для підготовки критеріїв попередження, замість того щоб призначати попередження, а потім набудовувати їх доти, поки на телефон не буде надходити тільки потрібна інформація. Використовуючи інформацію про події, уже зібрану в журналах, варто почати з тих же критеріїв, які застосовуються для звітів, і уточнювати їх, відкидаючи менш значимі події, що не заслуговують попередження. Якщо зібрані дані приблизно за місяць, то такий підхід, по суті, дозволяє моделювати обробку попереджень за тривалий період часу, щоб оцінити кількість вступників попереджень. Після того як буде знайдений прийнятний критерій у формі звітів Log Parser (або іншого інструмента підготовки звітів), можна перетворити критерій у запити WMI усередині сценаріїв VBScript. Щоб сценарії попереджень були постійно активні, можна настроїти їх на запуск у процесі початкового завантаження комп'ютера. Сценарії початкового завантаження можна розмістити в будь-якому об'єкті групової політики Group Policy Object (GPO) у розділі настроювання політик Computer Configuration\WindowsSettings\Startup and Shutdown Scripts. Сценарії, активізуємі при початковому запуску, працюють у контексті локального облікового запису System, тому в них є всі необхідні повноваження.

Ключ до підготовки ефективних звітів і критеріїв попереджень – відтинати непотрібну інформацію, який не повинне бути у звітах або попередженнях, а не вивуджувати корисні дані. Через численність джерел подій і поганої документованості більшості додатків і мережних пристроїв не існує способу

перелічити всі важливі події й повідомлення. Якщо обмежити критерій пошуком відомих найважливіших подій, то виникає реальна погроза, що в результаті неполадки пристрою або комп'ютера не буде отримане повідомлення про подію, не враховану в критеріях. Для балансу варто підготувати кілька попереджень або звітів, орієнтованих на конкретні критерії. Корисно підготувати такі попередження й звіти для вже знайомих проблем. Але в цілому при проектуванні критеріїв рекомендується додержуватися принципу виключення, а не включення.

Моніторинг текстових журналів

Більшість серверних продуктів Microsoft і компонентів Windows протоколюють будь-які важливі події в журналах System або Application, але кожна служба або основний компонент операційної системи (наприклад, IIS, DHCP, SMTP, Internet Authentication Service, IAS) записують більш докладну інформацію у власний текстовий журнал у спеціальному форматі. Для моніторингу інформації, наявної тільки в цих текстових файлах, зручно використовувати Log Parser. Log Parser розпізнає будь-який формат текстового файлу з розмежувачами, наприклад із символами табуляції або комами (CSV), і дозволяє задіяти ту ж команду SQL Select для опитування текстових файлів журналів.

Таким чином, за допомогою Log Parser можна вирішити задачу підготовки звітів на основі текстових журналів. Але що робити із попередженнями про критичні події, що поступають у реальному часі, інформація про які зберігається в текстових файлах? Я рекомендую скористатися інструментом tail, запозиченим з UNIX. Tail відслідковує додавання нових рядків у зазначені користувачем текстові файли. Як тільки виявляються нові дані, tail посилає їх у стандартний вихідний потік (stdout). Вихідні дані tail можна направити в сценарій, що аналізує нові записи в міру їхнього протоколювання й при необхідності генерує попередження. Наприклад:

```
tail /f logfile.txt |LoopOnNewMessages.cmd
```

виявляє нові повідомлення, додані у файл logfile.txt, і направляє їх в LoopOnNewMessages.cmd. Loop OnNewMessages.cmd передає кожне

повідомлення за адресою rsmith@ultimatewindowssecurity.com, але замість нього можна вказати будь-яку іншу поштову адресу. Щоб не пересилати не занадто істотні повідомлення, можна доповнити сценарій фільтруючою логікою.

Tail

Витягти інформацію з журнальних файлів або інших текстових файлів можна за допомогою широко розповсюдженої програми grep. Але є й інший корисний інструмент із миру UNIX, гідний зайняти місце в інструментальному наборі адміністратора Windows, – програма tail. По суті, tail показує останні кілька рядків текстового файлу – це особливо корисно при аналізі файлів журналів. Наприклад, якщо адміністратор становить нові правила для брандмауера, tail покаже, як правила відбиваються на файлі журналу.

Tail є в більшості систем UNIX, а версію Win32 можна завантажити в рамках прийняття умов ліцензії GNU з Web-вузла Sourceforge (*http://unxutils.sourceforge.net*). Спочатку потрібно завантажити файл UnxUpdates.zip, а потім витягти tail.exe у своєму комп'ютері.

Автономне використання Tail

При використанні поза комбінацією з іншими програмами, tail показує кілька останніх рядків текстового файлу. За допомогою декількох параметрів можна змінити подання інформації на екрані. Особливо корисний параметр follow (-f), що дозволяє безупинно відслідковувати й виводити на екран зміни в текстовому файлі. Наприклад, команда

```
tail -f ex050410.log
```

показує останні 10 рядків журнального файлу з ім'ям ex050410.log і буде відслідковувати й відображати нові записи в міру їхньої появи. Якщо файл являє собою журнал Web-служби Microsoft IIS і хто-небудь звертається до Web-вузла, IIS зробить у журналі новий запис. Нові додавання негайно відображаються на консолі, у якій працює tail. Цей параметр спрощує діагностику, дозволяючи негайно побачити нові записи.

Спільне застосування Tail і Grep

Як відомо, grep – програма, що веде пошук зазначених послідовностей символів у цільовому текстовому файлі. Наприклад, при діагностиці комп'ютера, що працює з Windows Firewall, потрібно відшукати в журналі брандмауера дії, зроблені в певний день. Журнал не розділений по датах і досить великий.

За допомогою grep можна витягти рядки даних від 7 березня 2009 року й записати їх у новий текстовий файл:

```
grep « 2009-03-07» p-firewall.log
> 030705 p-firewall.log
```

Як щодо tail? Команду можна використовувати для обробки журналів брандмауера в процесі діагностики або відстеження атак у реальному часі. Але можна застосувати tail разом з grep, щоб виводити на екран тільки певні дані.

Для початку варто настроїти брандмауер на запис журналів у текстовий файл. Всі системи UNIX використовують syslog для протоколювання подій; більшість комерційних брандмауерів також підтримують syslog. Якщо на системі UNIX використовуються grep і tail, то варто настроїти брандмауер на пересилання даних syslog у хост-машину syslog. Користувачі Windows можуть установити й працювати із сервером syslog на базі Windows. Я рекомендую Kiwi Syslog Daemon фірми Kiwi Enterprises, відмінний інструмент для збереження даних syslog у текстовому файлі.

Потім потрібно побудувати шаблон на основі синтаксису постачальника брандмауера. Наприклад, адміністратор використовує брандмауера Cisco PIX і хоче одержувати оповіщення щораз, коли хтось звертається до Web-служб через брандмауера. За допомогою tail і grep можна в реальному часі виявляти в журналах символи «/80» (представляють Web-трафік у журналі PIX), наприклад:

```
tail -f pix.log | grep «/80»
```

Більш вдалий підхід – використовувати метасимволи регулярних виражень, які забезпечують більше складну фільтрацію, чим звичайні текстові рядки:

```
tail -f pix.log | grep /80[[:space:]]
```

Освоєння регулярних виражень вимагає часу, але в нагороду ви одержуєте бібліотеку корисних і ефективних шаблонів, які можна використовувати для пошуку майже будь-яких даних, – безсумнівно, це виправдує витрачені зусилля.

Ускладнений Tail

Grep і tail – прості у використанні й дуже гнучкі програми. При роботі з консольними додатками обидва інструменти значно спрощують аналіз журнальних файлів і повсякденне адміністрування. Версія командного рядка tail – швидка й проста в експлуатації, і, імовірно, прихильники строгих правил нададуть їй перевагу завдяки простоті й можливості пересилати вихідні дані в інші програми, такі як grep. Але існують версії tail із графічним інтерфейсом Windows, причому деякі з них наділені більш складними функціями, наприклад кольоровим виділенням співпадаючих послідовностей. Таке форматування допомагає відзначати важливі файли.

Зразок безкоштовної графічної програми tail для Windows – BareTail компанії Bare Metal Software (її можна завантажити за адресою *http://www.baremetalsoft.com/baretail*). Як і tail, BareTail відображає текстовий файл і відслідковує доповнення до файлу, але оскільки BareTail працює із графічним інтерфейсом, вона має у своєму розпорядженні функції виділення.

Завдяки таким функціям простіше виявити певний текст (наприклад, конкретну IP-адресу або порт) «на ходу», спостерігаючи за журналом брандмауера. Можна також змінити шрифт, без праці скопіювати рядок тексту й відкрити недавно переглянуті файли журналів за допомогою списку недавно використаних файлів Windows.

Моніторинг SNMP і журналів Syslog

Контролювати телеметричні джерела SNMP і Syslog легко завдяки безкоштовній версії програми Kiwi Syslog Daemon компанії Kiwi Enterprises. Цей диспетчер серверів Windows Syslog і SNMP дозволяє зібрати всі телеметричні дані про мережні пристрої в одній програмі. Із графічного інтерфейсу програми можна настроїти фільтри для збору повідомлень, що відповідають певним

критеріям, а потім указати одне або кілька дій, що вживаються у відповідь на повідомлення. Можна побудувати фільтри для видалення непотрібних повідомлень і вказати, що повідомлення, що залишилися, повинні генерувати попередження або зберігатися в базі даних для наступних звітів. За допомогою Kiwi Syslog Daemon можна фільтрувати повідомлення за часом дня, днем тижня, пристроям, рівню, IP-адресі звітного агента або рядкам у повідомленні. Крім того, інструмент може виконувати різноманітні дії – оповіщення по електронній пошті, збереження в базі даних по ODBC, запуск програми й інші – у відповідь на зазначені події.

Безкоштовна версія Kiwi Syslog Daemon інтерактивно працює в настільному комп'ютері, тому адміністратор повинен зареєструватися, щоб контролювати пристрої. Але розширена версія продукту функціонує як служба, а її вартість – усього 100 дол. для одного сервера. Якщо активізувати моніторинг пасток SNMP, необхідно також указати поля Facility і Level, які використовуються інструментом при перетворенні пастки в повідомлення Syslog. Наприклад, можна вказати пастки SNMP як Facility Local4 і Level 3 Error. Потім можна скласти правила розсилання попереджень спеціально для пасток SNMP шляхом фільтрації повідомлень Local4.

Отже, існують ресурси для моніторингу різноманітних джерел телеметричних даних. Перш ніж почати проектувати власне рішення для моніторингу, корисно познайомитися з інструментами, які є на ринку. Вони доступні й повнофункціональні. Однак не можна одержати повне рішення, просто здобуваючи інструмент. Потрібно визначити критерії для звітів і попереджень, щоб не одержувати занадто багато повідомлень про незначні події, але не слід упадати в іншу крайність і задавати настільки строгі критерії, що рішення моніторингу може перешкодити виконанню тої самої задачі, для якої воно призначалося. Для досягнення балансу варто становити критерії, відтинаючи незначні, а не вибираючи важливі події. Єдине виключення із цього правила – журнал Security, що набагато коротше, а крім того, краще

документовано. Повна база даних подій журналу Security і їхніх значень опублікована в Security Log Encyclopedia на сайті Ultimate Windows Security (*www.ultimatewindowssecurity.com*).

Для підготовки ефективної й вичерпної процедури моніторингу потрібно прикласти певні зусилля, але вони не пропадуть впусту. Немає нічого гірше, ніж довідатися про проблему від користувачів і після перегляду журналів виявити, що попередження надходили трьома днями раніше. Вимоги бізнесу й законодавства щодо інформаційної безпеки й звітності дуже високі. Порушення безпеки можливі, але адміністратор і його компанія набагато успішніше переборють труднощі, якщо добре підготуються до критичної ситуації. Ефективному моніторингу немає повноцінної заміни.

Розробка удосконаленого методу моніторингу мережі

Як було відмічено вище, однією з основних проблем, що стоять зараз перед розроблювачами систем керування комп'ютерними мережами є проблема достовірного надання даних про їхній стан. Потреба в надійно працюючих великих комп'ютерних мережах усе вища з кожним днем. Тому при проектуванні сучасних систем керування дуже важливу роль відводять розробкам оптимізованих за часом алгоритмам збору й обробки даних.

Імовірність збереження актуальності інформації на момент її використання чисельно дорівнює:

$$P = \frac{c^2}{(c+b)(c+q)}, \qquad (3.1)$$

де c – середній час значимої зміни реальної інформації щодо інформації, збереженої в БД;

b – середній час підготовки, передачі й уведення інформації для відновлення БД;

q – середній час між двома послідовними опитуваннями того самого пристрою.

Ідеальним випадком є ситуація, коли інформація про будь-яку зміну стану досягає БД у момент зміни. З використанням механізму багатопоточності можна «скоротити» час очікування інформації від джерела, використавши його для обробки інформації від іншого джерела.

На представленому нижче графіку (рисунок 3.1) видна залежність імовірності збереження актуальності даних від відношення часу очікування відповіді до часу обробки даних.

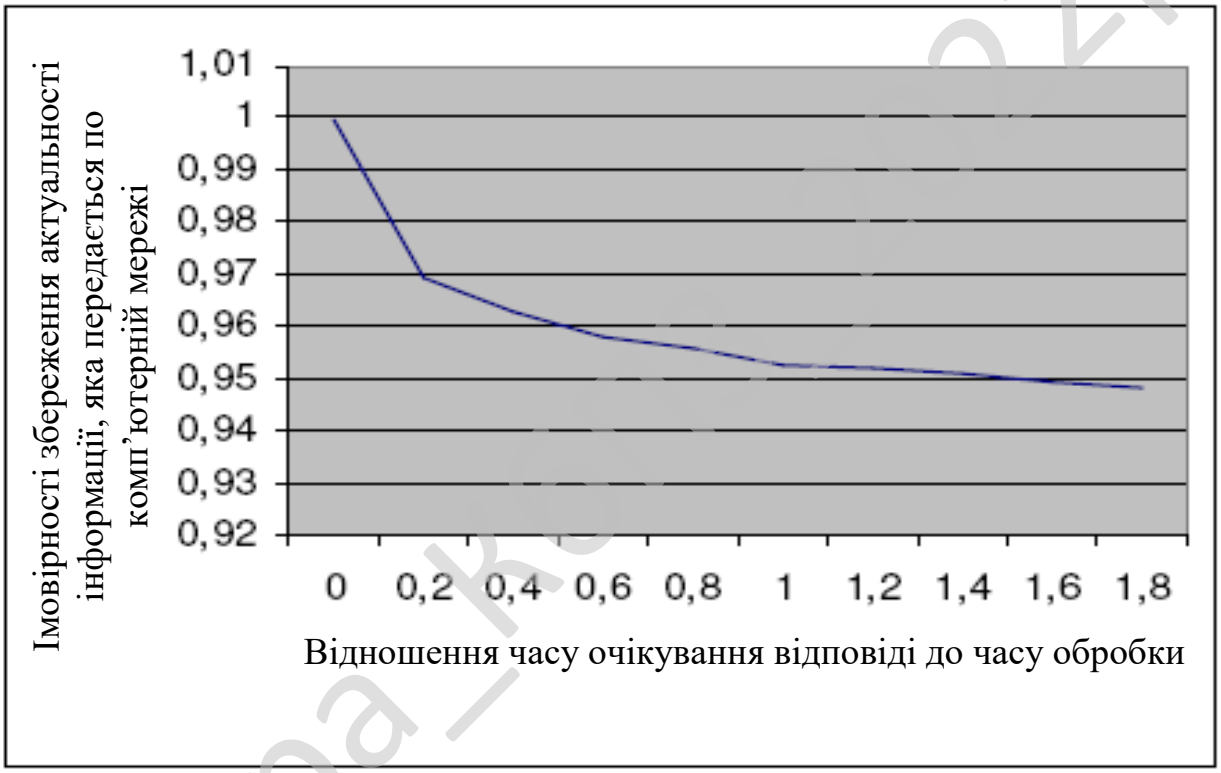

Рисунок 3.1 – Залежність імовірності збереження актуальності від відношення часу очікування відповіді до часу обробки

Із графіка видно, що навіть незначне скорочення часу очікування даних може привести до істотного збільшення ймовірності збереження актуальності даних, якщо час очікування приблизно дорівнює часу обробки. Можна зробити висновок, що для мереж, у яких при роботі системи моніторингу час обробки інформації приблизно дорівнює часу очікування після запиту, їсти можливість

істотно підвищити актуальність інформації при використанні алгоритмів, що дозволяють використовувати час простою для обробки інформації.

Сучасні операційні системи дозволяють використовувати багатопоточну схему роботи додатків. Це досягається за рахунок розподілу робочого часу процесора між різними додатками, що дозволяє декільком програмам працювати «одночасно».

Весь процесорний час персонального комп'ютера (сервера) можна розділити на періоди. В один такий період всі додатки одержують по «шматочку» процесорного часу для своїх потреб. Їхні розміри залежать від пріоритетів додатків в операційній системі. Чим вище пріоритет – тим довший часовий інтервал, у якому додаток використовує процесор. У рамках даного інтервалу додаток також може й не використовувати процесор (перебуває в стані очікування). Чим більше потоків «всередині» у додатка, тим більше додаток в цілому одержує цього процесорного часу.

Для складних математичних задач введення механізму паралельного розрахунку даремно, тому що час роботи додатка приблизно дорівнює часу процесора для розрахунку задачі. Для задач моніторингу дана схема навпроти є досить вигідною. Вона дозволяє одержати виграш за часом за рахунок того, що в процесі опитування робочої станції є часові інтервали, у яких не використовуються ресурси процесора, пам'яті, мережі. Це інтервали очікування відповіді від віддаленого пристрою. Тим самим, якщо використовувати ці інтервали для роботи інших потоків, то можна буде зменшити час «простою» процесора. А за рахунок цього відбувається оптимізація за часом. Очевидно, що якщо змусити процесор працювати без «простою» у рамках відведеного для додатка процесорного часу й при цьому не допускати переповнення пам'яті або надлишкового мережного трафіку, така система буде максимально ефективна.

Як уже було визначено раніше, збільшення кількості паралельно опитуваних робочих станцій спочатку збільшує продуктивність системи, а після

створення n+1 потоку, навпаки, сповільнюють її. Це пов'язане з закінченям обчислювальних ресурсів.

На продуктивність системи впливають три основних фактори: завантаженість центрального мікропроцесора, оперативний пам'яті, а також мережний трафік.

Очевидно, що поки всі три ресурси будуть використані не повністю, додавання нового потоку в систему буде збільшувати її продуктивність. Але як тільки один з ресурсів буде повністю вичерпаний, продуктивність системи або впаде, або втратить динаміку росту. Приміром, якщо при неповному завантаженні центрального процесора буде повністю зайнята доступна оперативна пам'ять, при додаванні нового потоку в систему, частину процесорного часу буде витрачатися на керування задачами по перевантаженню даних з оперативної пам'яті на жорсткий диск. Таким чином, одна з основних задач, яку необхідно вирішити при створенні систем моніторингу формулюється в такий спосіб: необхідно визначити максимальне число потоків, при якому система опитування робочих станцій буде працювати з максимальною ефективністю.

Щоб формально описати дану задачу необхідно визначити, як залежить продуктивність системи від значення вищеописаних факторів.

При запуску опитування мережі відбувається формування запиту і його відправлення віддаленому пристрою. Після цього потік переходить у стан очікування й до відповіді робітник станції майже не займає ресурсів процесора, пам'яті й не створює мережного трафіку. Природно, що в момент очікування відповіді одним потоком, використовувати ресурси процесора може інший потік.

Виходячи із усього вищесказаного й за умови необмежених ресурсів оперативної пам'яті й мережного трафіку, були визначені наступне співвідношення для визначення оптимальної кількості потоків, що додаються в систему моніторингу:

$$N = \frac{t_t}{t_p}, \tag{3.2}$$

де t_t – тривалість потоку (від запиту даних до закінчення їхньої обробки);

t_p – тривалість використання потоком ресурсів центрального процесора.

Дане співвідношення справедливо для ідеального випадку. У реальній ситуації процесор не може надати всі свої ресурси для системи моніторингу. Частина його ресурсів іде на керування операційною системою й іншими, а також використовується іншими додатками.

У зв'язку із цим протягом часу, за яке працює потік, до ресурсів процесора звертаються не тільки потоки системи моніторингу, але й інші додатки.

У такий спосіб вищенаведене співвідношення можна обмежити цією умовою:

$$N = \frac{t_t(1-P)}{t_p}, \tag{3.3}$$

де t_t – тривалість потоку (від запиту даних до закінчення їхньої обробки);

t_p – тривалість використання потоком ресурсів центрального процесора;

P – коефіцієнт завантаженості процесора іншими додатками ($0 < P < 1$).

Таке уточнення співвідношення дозволяє в будь-який момент часу визначити має сенс чи ні в цей момент часу додати додатковий потік у систему.

Але існують і інші фактори, що впливають на продуктивність системи моніторингу. Другим по значимості є завантаженість оперативної пам'яті. Усе раніше наведені міркування дійсні за умови, що оперативна пам'ять не повна, тобто система не використовує файл підкачування. У випадку ж його використання існують додаткові часові витрати за часом на перевантаження даних з файлу до пам’яті й обернено.

При цьому варто пам'ятати, що перевантаження даних відбувається тільки в тому випадку, коли потік готовий до виконання. А це відбувається не на кожному циклі ітерації в керуючому потоці.

Тому для випадку, коли всі потоки не будуть міститися в оперативній пам'яті, то попереднє співвідношення не може бути використана. У випадку недостачі оперативної пам'яті необхідно використовувати наступну формулу:

$$N = \frac{t_t(1-P)}{t_p + M \cdot t_s}, \tag{3.4}$$

де t_t – тривалість потоку (від запиту даних до закінчення їхньої обробки);

t_p – тривалість використання потоком ресурсів центрального процесора;

P – поточна завантаженість процесора ($0 < P < 1$);

t_s – час перезавантаження даних з оперативної пам'яті у файл;

M – число таких перезавантажень.

Визначення значення коефіцієнта M не представляє особливої праці. Перевантаження відбудеться тільки тоді, коли потік перебуває в стані роботи, а не очікування. У зв'язку із цим значення M можна визначити за формулою:

$$M = \frac{t_p}{t_i}, \tag{3.5}$$

де t_p – тривалість використання потоком ресурсів центрального процесора;

t_i – час, на який надається доступ до ресурсу процесора потоку, при передачі йому керування в одній ітерації

Немаловажним фактором є завантаженість мережі. Очевидно, що при повному завантаженні мережі про паралельність також безглуздо говорити. Потоки будуть формуватися в послідовні черги, і при цьому мережа буде практично непрацездатна для інших додатків і користувачів. У двох попередніх випадках, перевантаження параметра вело лише до істотного вповільнення роботи одного комп'ютера, а для мережного трафіку може паралізувати роботу всієї мережі. Через це для кожної конкретної мережі встановлюється гранично припустимий мережний трафик, що може створювати система моніторингу. Його розрахунок ведеться з розрахунку розмірів мережі, її швидкості, кількості мережних додатків, часу доби й т.д.

Поєднуючи все вищесказане в єдину задачу одержуємо, що:

– якщо оперативна пам'ять не переповнена:

$$N = \frac{t_t(1-P)}{t_p}, \tag{3.6}$$

– якщо оперативна пам'ять переповнена:

$$N = \frac{t_t(1-P)}{t_p + \frac{t_p}{t_i} \cdot t_s}, \tag{3.7}$$

Обидва співвідношення обмежені умовою, що не перевищена установлена межа мережного трафіку, створюваного системою моніторингу.

Виконання цієї умови, а також визначення переповнення оперативної пам'яті відбувається відповідно до раніше певних співвідношень.

Всі параметри, необхідні для розрахунку оптимальної кількості потоків надаються операційною системою.

Визначивши оптимальну кількість потоків у будь-який момент часу, була вирішена тільки половина поставленої задачі. У **класичній багатопоточній схемі** опитування мережі, нам потрібно розподілити весь діапазон IP-адрес на *N* груп, і провести опитування. Але класична схема не враховує того, що в сучасному житті під один додаток у мережі не виділяється сервер. Завантаженість сервера, на якому встановлена система моніторингу, постійно міняється через використання інших додатків, розташованих на ньому. А зі зміною завантаженості сервера міняється оптимальна кількість потоків, у рамках яких проходить опитування кінцевого або мережного устаткування. **Друга проблема класичної схеми** полягає в тому, що час опитування одного пристрою залежить від його типу, об'єму збирається информации, що, часу, необхідного на її обробку, а також швидкості каналу зв'язку. При «класичному» розподілі IP-адрес на групи всі ці моменти не враховуються, а це приводить до того, що в деякий момент часу одна група повністю оброблена, а інша ні.

З обліком цих двох причин для рішення поставленої задачі **дана модель була дороблена в такий спосіб**:

1. Визначаємо оптимальне число потоків у цей момент

2. Запускаємо *n* потоків у яких відбувається опитування перших *n* адрес діапазону.

3. Як тільки в якому-небудь потоці опитування закінчиться (або буде встановлена його неможливість) відбувається визначення оптимального числа потоків у цей момент.

4. Якщо оптимальне число потоків менше поточного, то потік (у якому закінчилася робота) знищується. Якщо більше, то створюється ще *к* потоків (*к* = оптимальне число потоків – існуюче число потоків).

Така схема дозволяє відслідковувати зміну стану системи в часі й дозволяє рівномірно розподіляти пристрою між потоками. На рисунку 3.2 представлений описаний вище алгоритм у графічній формі.

Для того щоб оцінити запропонований вище алгоритм системи моніторингу, було проведено його моделювання, а також моделювання класичних алгоритмів, у системі GPSS WORLD. Всі необхідні для такого моделювання параметри були визначені на реальній мережі. На їхній підставі були виведені наступні базові значення:

– Середнє число потоків для опитування мережі дорівнює 4.

– Середній час опитування 1 пристрою становить 80 секунд.

– Кількість перевантажень процесора або пам'яті серверів, на яких установлені засоби керування мережею рівнялося в середньому 6 разів за годину.

На підставі цих даних було зроблене моделювання опитування великої локальної мережі (більше 250 комп'ютерів) для трьох реалізацій моніторингу (однопотоковий, багатопоточний і динамічний багатопоточний). За отриманим даними були побудовані графіки продуктивності різних реалізацій моніторингу (рисунок 3.3).

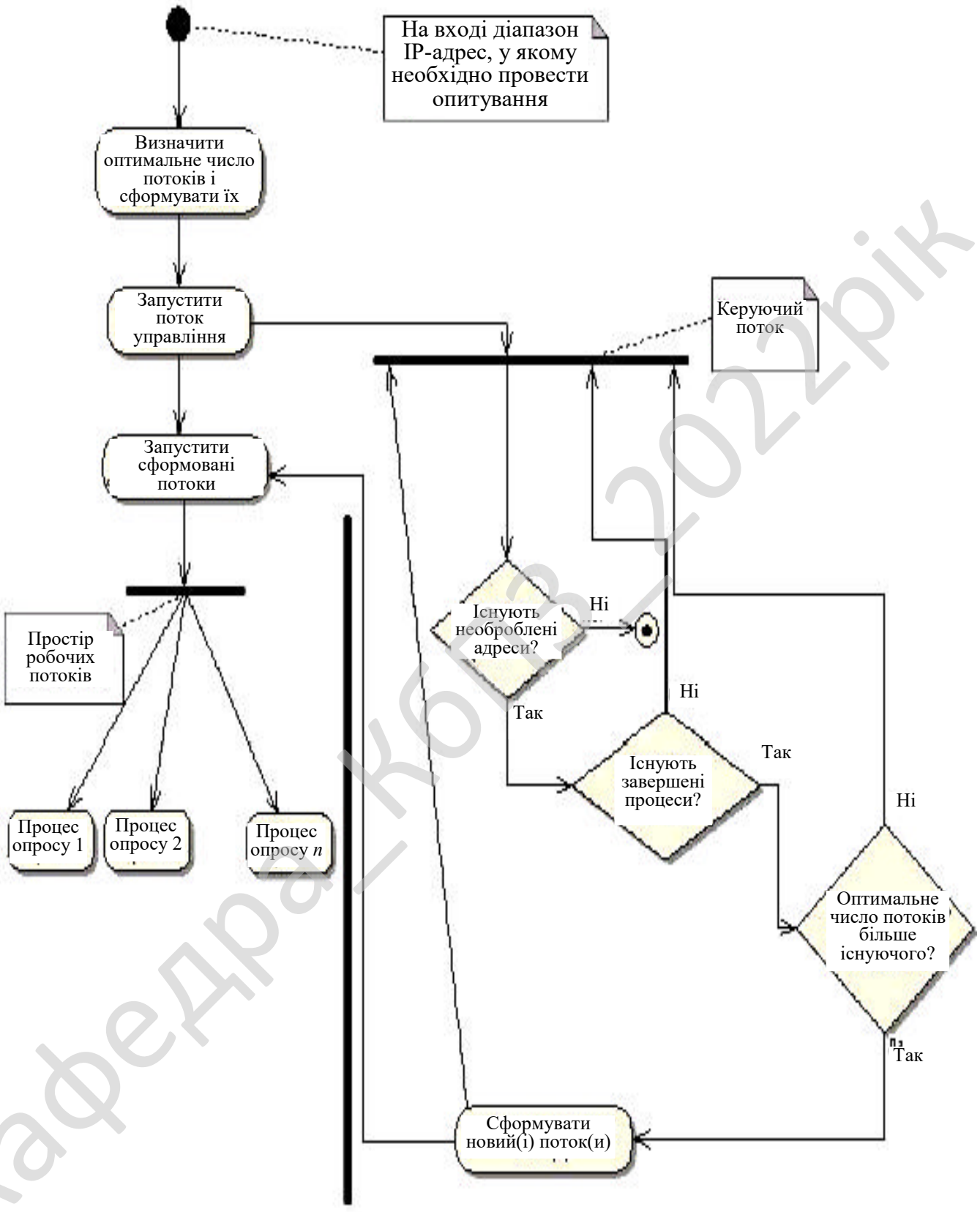

Рисунок 3.2 – Схематичне представлення удосконаленого алгоритму моніторингу мережі

Результати моделювання підтвердили раніше висунуті припущення. Модернізований багатопоточний алгоритм був реалізований у системі моніторингу. Після проведення тестових випробувань системи на мережі отримані дані підтвердили результати моделювання. Порівняння результатів отриманих при моделюванні й випробуванні системи моніторингу, побудованої на модернізованому багатопоточному алгоритмі, представлені на рисунку 3.4.

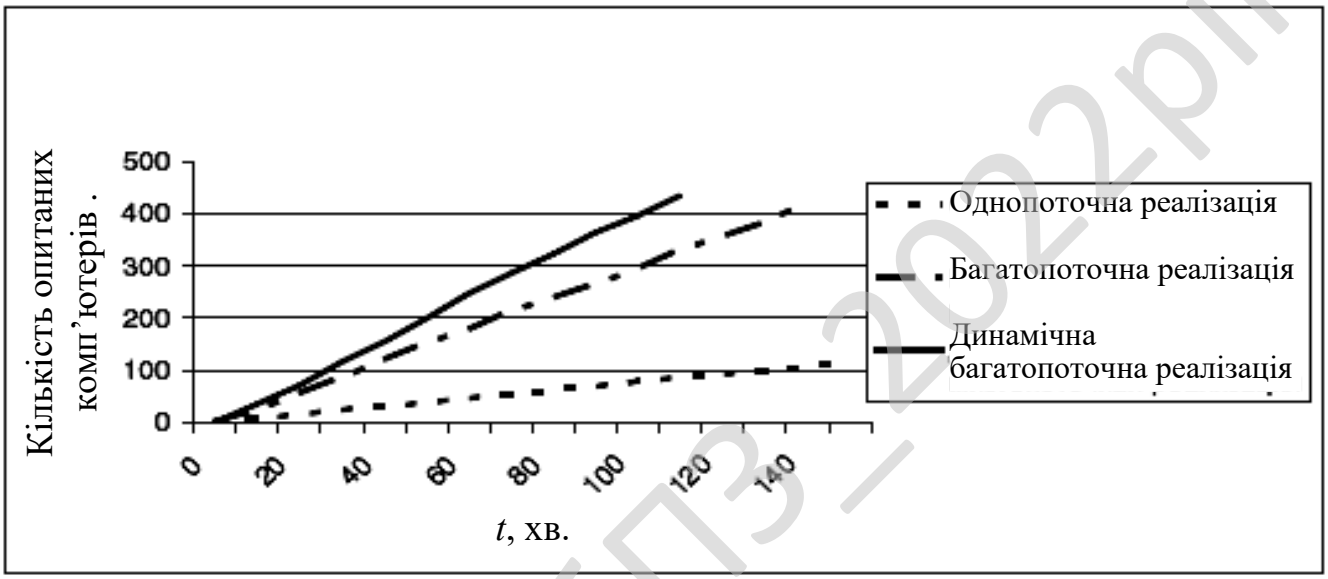

Рисунок 3.3 – Продуктивність різних реалізацій моніторингу при моделюванні

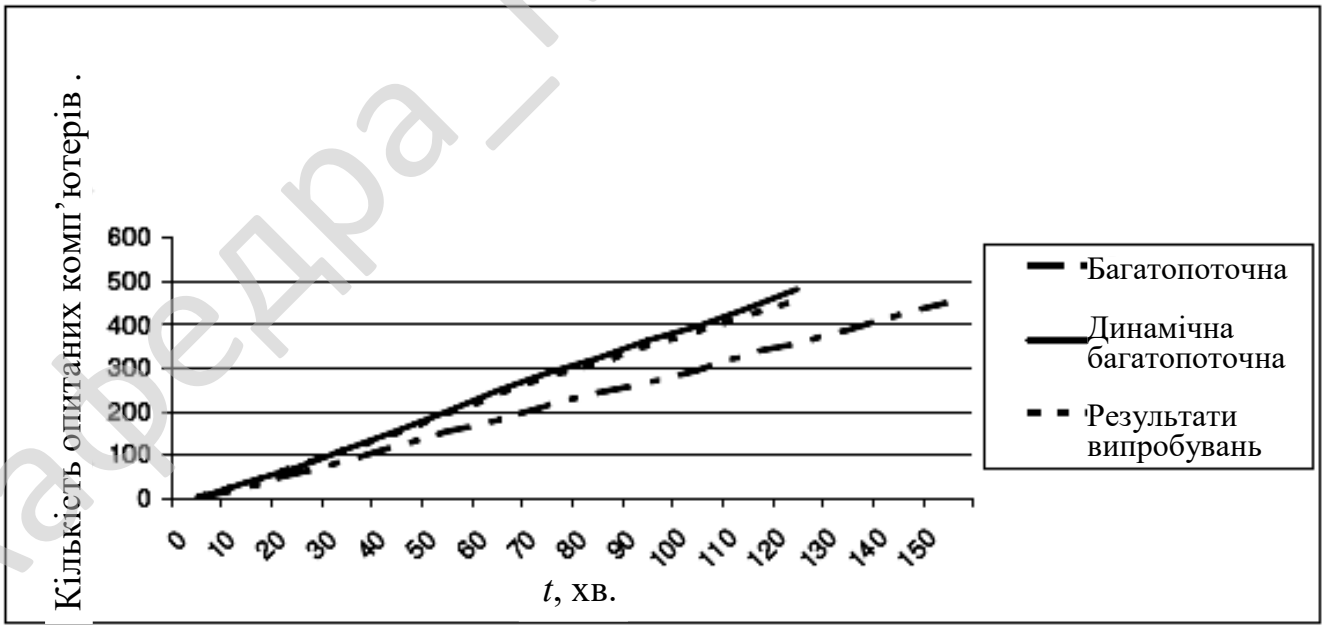

Рисунок 3.4 – Продуктивність різних реалізацій моніторингу при реальному випробуванні

На підставі аналізу наведених вище даних установлено, що час збору й обробки інформації для мережних або кінцевих пристроїв було скорочено більш ніж на 20%. Це означає, що час між повторними опитуваннями однієї й тієї ж робочої станції при круговому безперервному опитуванні змінилося, і стало становити 80% від того, котре забезпечує система моніторингу, побудована по класичній моделі. Таким чином, імовірність збереження актуальності даних для системи моніторингу, побудованої на алгоритмах розроблених у даній роботі для мереж збільшився з 92% до 95,5%.

Даний результат перевершує 95%, що у цей момент вважається мінімальним рівнем для вимог по актуальності даних.

3.2 Розробка структурної схеми

Структурна схема системи зображена на рисунку 3.5.

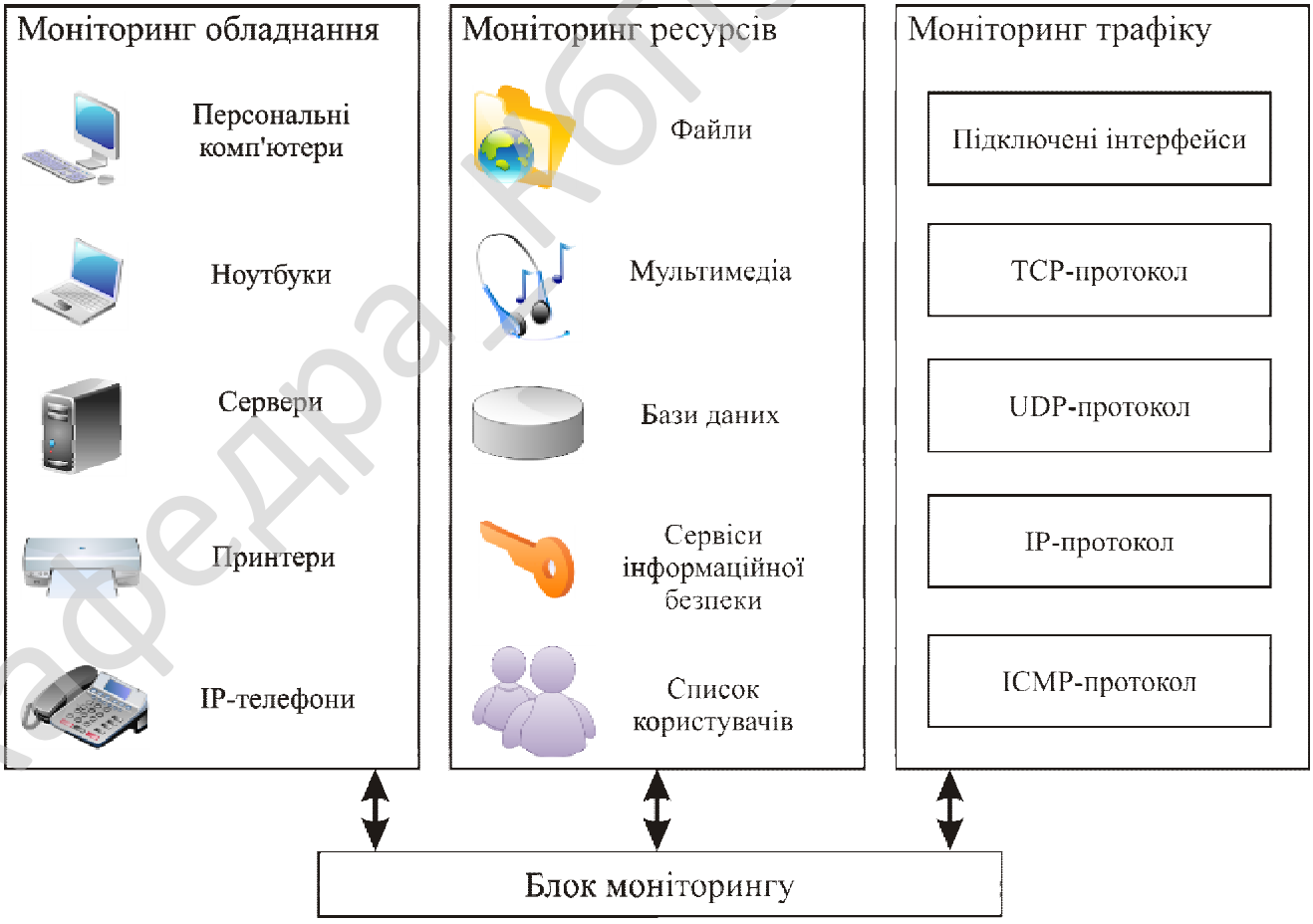

Рисунок 3.5 – Структурна схема системи

З рисунку видно, що моніторинг локальної мережі здійснюється по трьох напрямках:

– Моніторинг обладнання.
– Моніторинг ресурсів.
– Моніторинг трафіку.

Моніторинг обладнання включає в себе побудову списку наявного обладнання та здійснення його контролю. До мережного обладнання, що підлягає моніторингу, відносяться: персональні комп'ютери, ноутбуки, сервери, принтери, ip-телефони.

Моніторинг ресурсів дозволяє переглядати та завантажувати наявні в мережі ресурси, а також розміщювати чи приховувати для загального доступу свої ресурси. До ресурсів локальної мережі відносяться: файли, мультимедіа, бази даних, сервіси інформаційної безпеки, список користувачів.

Моніторинг трафіку використовується для контролю вхідного та вихідноготрафіку. Він включає у себе контроль підключених інтерфейсів, статистику подій по основним мережним протоколам: TCP, UDP, IP та ICMP.

TCP – один з основних мережних протоколів Інтернету, призначений для управління передачею даних в мережах і підмережах TCP/IP.

UDP – один із протоколів в стеку TCP/IP. Від протоколу TCP він відрізняється тим, що працює без встановлення з'єднання. UDP – це один з найпростіших протоколів транспортного рівня моделі OSI, котрий виконує обмін дейтаграмами без підтвердження та гарантії доставки.

IP – найбільш широко розповсюджена реалізація ієрархічної схеми мережної адресації. Використовуваний в мережі Інтернет, протокол відповідає за адресацію пакетів, але не відповідає за встановлення з'єднань, не є надійним і дозволяє реалізувати тільки негарантовану доставку даних.

ICMP – мережний протокол, що входить в стек протоколів TCP/IP. В основному ICMP використовується для передачі повідомлень про помилки й інші виняткові ситуації, що виникли при передачі даних. Також на ICMP покладають

деякі сервісні функції, зокрема на основі цього протоколу заснована дія таких загальновідомих утиліт як ping та traceroute.

3.3 Розробка функціональної схеми

Функціональна схема системи зображена на рисунку 3.6.

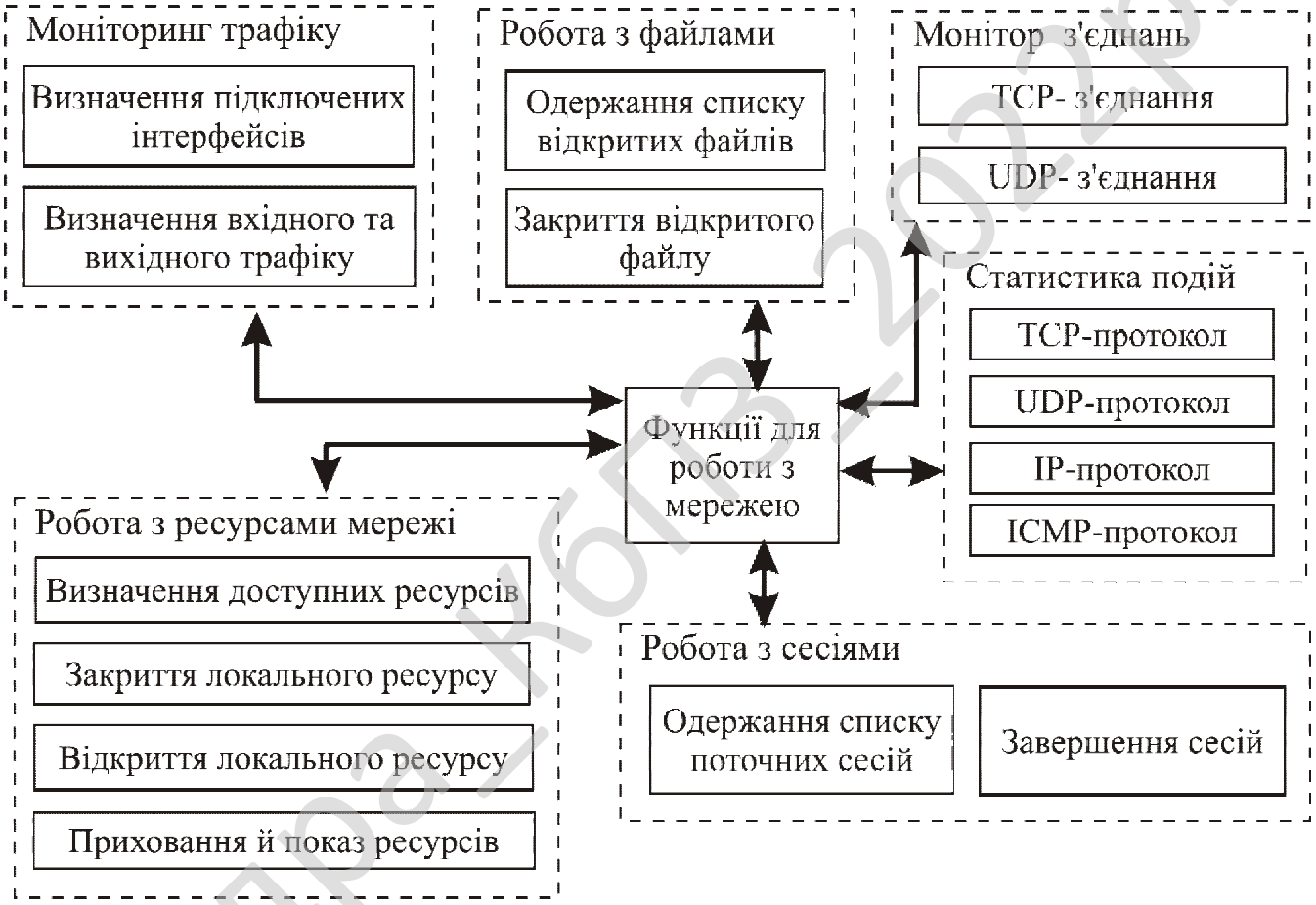

Рисунок 3.6 – Функціональна схема системи

З рисунку видно, що розроблена система складається з наступних блоків:

- Моніторинг трафіку.
- Робота з файлами.
- Монітор з'єднань.
- Статистика подій.

– Робота з ресурсами мережі.

– Робота з сесіями.

– Функції для роботи з мережею.

Розглянемо детальніше кожний з блоків.

Моніторинг трафіку включає в себе:

– Визначення підключених інтерфейсів.

– Визначення вхідного та вихідного трафіку.

Розроблена система відображає всі інтерфейси приєднані до комп'ютера, на якому запущена програма, їх MAC-адреси та вхідний і вихідний трафік на кожному з них.

Робота з файлами включає в себе:

– Одержання списку відкритих файлів.

– Закриття відкритого файлу.

Можна переглянути, які з Ваших файлів, що Ви відкрили для загального доступу, переглядають по мережі. Програма відобразить список файлів та користувачів, які їх переглядають. Також можна відкрити чи закрити файл.

Монітор з'єднань включає в себе:

– Відстеження TCP- з'єднань.

– Відстеження UDP- з'єднань.

Система фіксує всі підключення по TCP- та UDP-протоколу, та виводить їх на екран у форматі *IP-адреса:порт_призначення*.

Статистика подій включає в себе відстеження подій в наступних протоколах:

– TCP-протокол.

– UDP-протокол.

– IP-протокол.

– ICMP-протокол.

Статистика ведеться по цілому ряду параметрів. Наприклад для TCP-протоколу фіксується: Тип алгоритму повторної передачі, мінімальний тайм-аут,

максимальний тайм-аут, максимальна кількість помилок з'єднання, активні з'єднання, пасивні з'єднання, невдалі спроби відкриття, скидання встановлених з'єднань, отримані сегменти, надіслані сегменти, повторно передані сегменти, помилки тощо.

Робота з ресурсами мережі включає в себе:

- Визначення доступних ресурсів.
- Закриття локального ресурсу.
- Відкриття локального ресурсу.
- Приховання й показ ресурсів.

Система відображає наявні у мережі ресурси у вигляді дерева. Пошук ресурсів можна здійснювати по заданим умовам: локальні чи глобальні ресурси; всі ресурси, тільки файли, чи тільки принтери, тощо.

Можна додавати до загальних ресурсів мережі свої власні, а також закривати їх потім.

Робота з сесіями включає в себе:

- Одержання списку поточних сесій.
- Завершення сесій.

Програма дозволяє переглянути список відкритих сесій, що включає в себе: назву сесії, користувача, що її розпочав, номер сесії, час роботи та час очікування.

3.4 Розробка діаграми процесів

Діаграма процесів системи зображена на рисунку 3.7. Як видно з рисунку, після завантаження системи можна запустити наступні процеси:

- Пошук ресурсів локальної мережі.
- Моніторинг трафіку.
- Статистика.
- Вивід на екран відкритих по мережі файлів.
- Вивід на екран відкритих сесій.

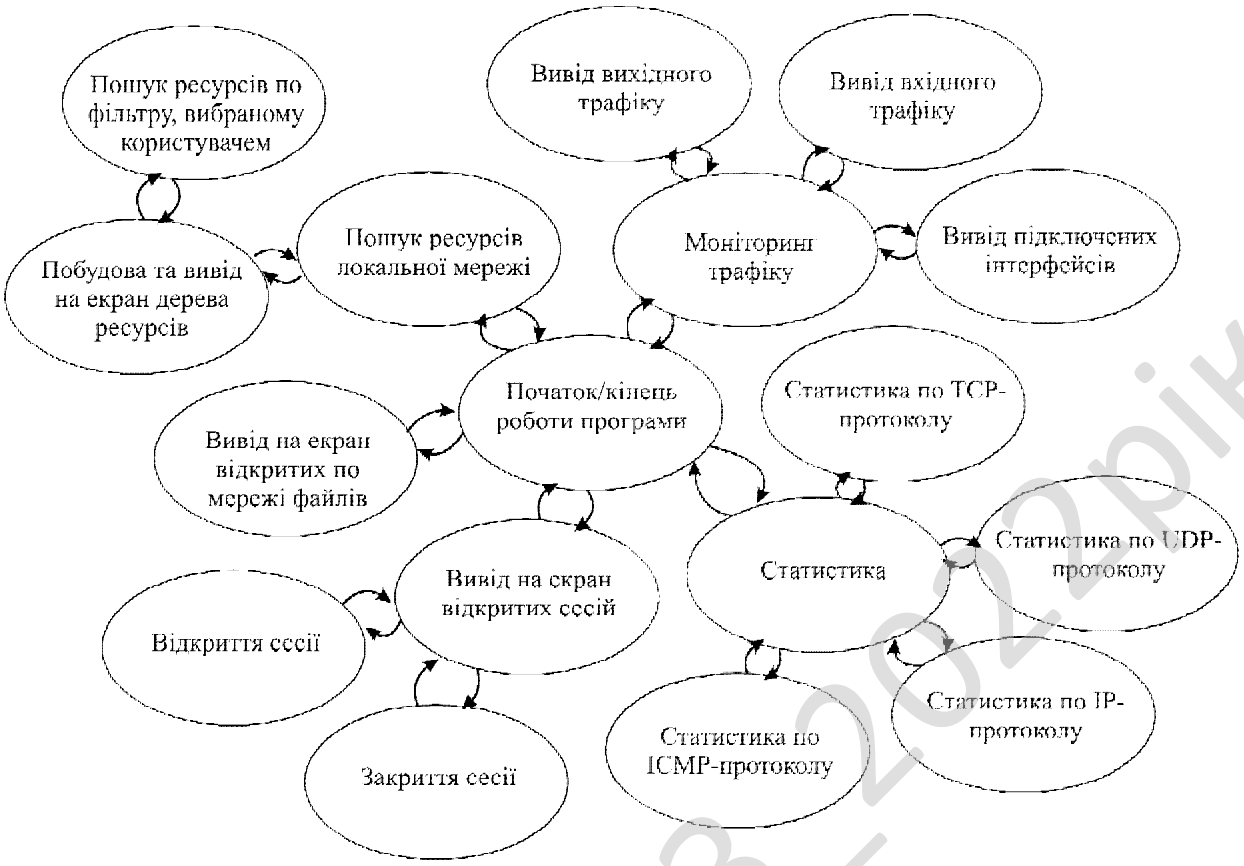

Рисунок 3.7 – Діаграма процесів системи

Після пошуку ресурсів локальної мережі запускається процес побудови та виводу на екран дерева ресурсів. Також можна здійснити пошук ресурсів по фільтру, вибраному користувачем.

Моніторинг трафіку включає в себе процеси виводу підключених інтерфейсів, виводу вихідного трафіку та виводу вхідного трафіку.

Статистика включає в себе статистику по TCP-протоколу, UDP-протоколу, IP-протоколу та ICMP-протоколу.

Після виводу на екран відкритих сесій можна запустити процеси відкриття сесії та закриття сесії.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо алгоритм роботи програми. Блок-схема роботи основної програми зображена на рисунку 4.1.

Після запуску розробленої системи відбувається вивід головного вікна програми. Потім здійснюється пошук ресурсів локальної мережі та побудова дерева ресурсів.

Структура мережі будується на основі автоматичного сканування пристроїв мережі. Як елементи структури виступають технічні пристрої, що мають IP-адресу та ресурси, що на них містяться. Крім того, у структуру можна включати елементи, що не є технічними пристроями, а є об'єктами, які логічно поєднують пристрої. Як логічні об'єкти можуть виступати такі об'єкти як домен, підмережа, робоча група і т.д. Ну й нарешті як такі логічні об'єкти можуть виступати об'єкти, що не мають прямого відношення до структури мережі, а являються скоріше організаційними об'єктами, такі як «Співробітник», «Група співробітників», «Відділ» і т.д. У даній програмі структура мережі відображається у вигляді дерева елементів. Так як елементи структури мережі підлеглі один одному, то деревоподібна система найкраще підходить для відображення такої структури. Вкладеність елемента дерева відображає підпорядкованість елементів один одному.

Далі на екран виводяться:

- Список відкритих по мережі файлів.
- Список відкритих сесій.
- Вхідний та вихідний трафік.
- Статистика системних подій у мережі.

Користувач може переглядати та змінювати дану інформацію. Зокрема, він може відкривати та закривати ресурси локальної мережі.

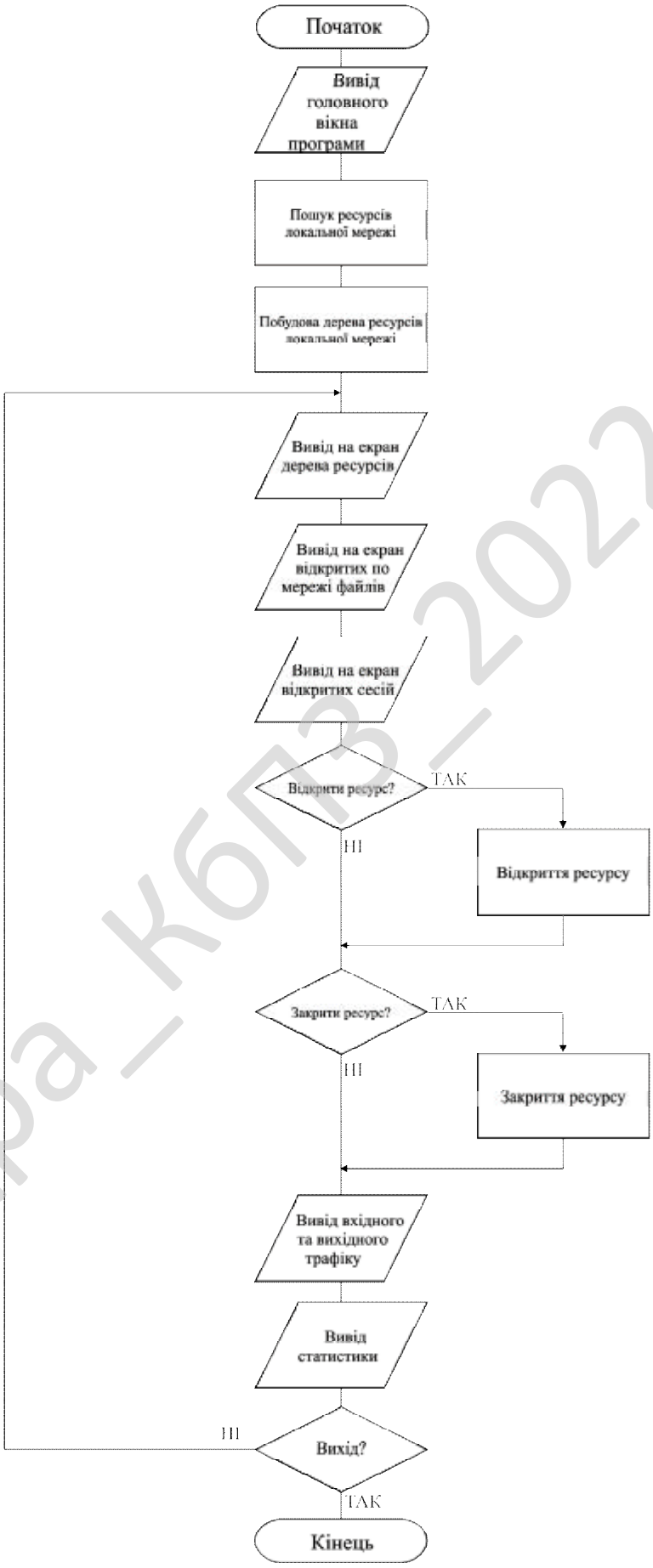

Рисунок 4.1 – Блок-схема основної програми

Визначення доступних ресурсів

Блок-схема алгоритму визначення доступних локальних ресурсів зображена на рисунку 4.2.

Розглянемо функції, що надають інформацію про локальні ресурси й можливість їхнього контролю.

NetShareEnum – за допомогою цієї функції ми одержимо дані про всі свої й чужі загальні ресурси.

Оголошення функції для Windows 10/11:

```
var
    NetShareEnum :function (pszServer        : PChar;
                            sLevel     : Cardinal;
                            pbBuffer         : PChar;
                            cbBuffer         : Cardinal;
                            pcEntriesRead,
                            pcTotalAvail     : Pointer):DWORD; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо виконуємо в себе то даному параметру можна привласнити NIL.

– sLevel – повинен містити ідентифікатор структури.

– pbBuffer – повинен містити *покажчик* на масив структур.

– cbBuffer – повинен містити розмір масиву структур.

– pcEntriesRead – повинен містити покажчик на змінну, в яку запишеться кількість спільних ресурсів доступних на даний момент.

– pcTotalAvail – не використовується.

Оголошення функції для Windows NT:

```
var
    NetShareEnum :function (ServerName :PWChar;
                            Level             :DWORD;
                            Bufptr            :Pointer;
                            Prefmaxlen        :DWORD;
                            EntriesRead,
                            TotalEntries,
                            resume_handle:LPDWORD): DWORD; stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо виконуємо в себе то даному параметру можна привласнити NIL.

– Level – повинен містити ідентифікатор структури.

– Bufptr – повинен містити *адресу покажчика* на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах, якщо не ставити обмеження, то даному параметру потрібно привласнити DWORD(-1).

– EntriesRead – повинен містити покажчик на змінну, в яку запишеться кількість спільних ресурсів доступних на даний момент.

– TotalEntries – не використовується

– Resume_handle – не використовується, повинен бути NIL.

У випадку успішного виконання результат обох функцій дорівнює нулю. Слід звернути увагу на те, що функція, яка використовується в Windows 10/11 одержує саме покажчик на масив структур, у той час як інша функція одержує адресу покажчика, це критично.

Результати виконання будуть збережені в масиві структур передані функції при її виклику.

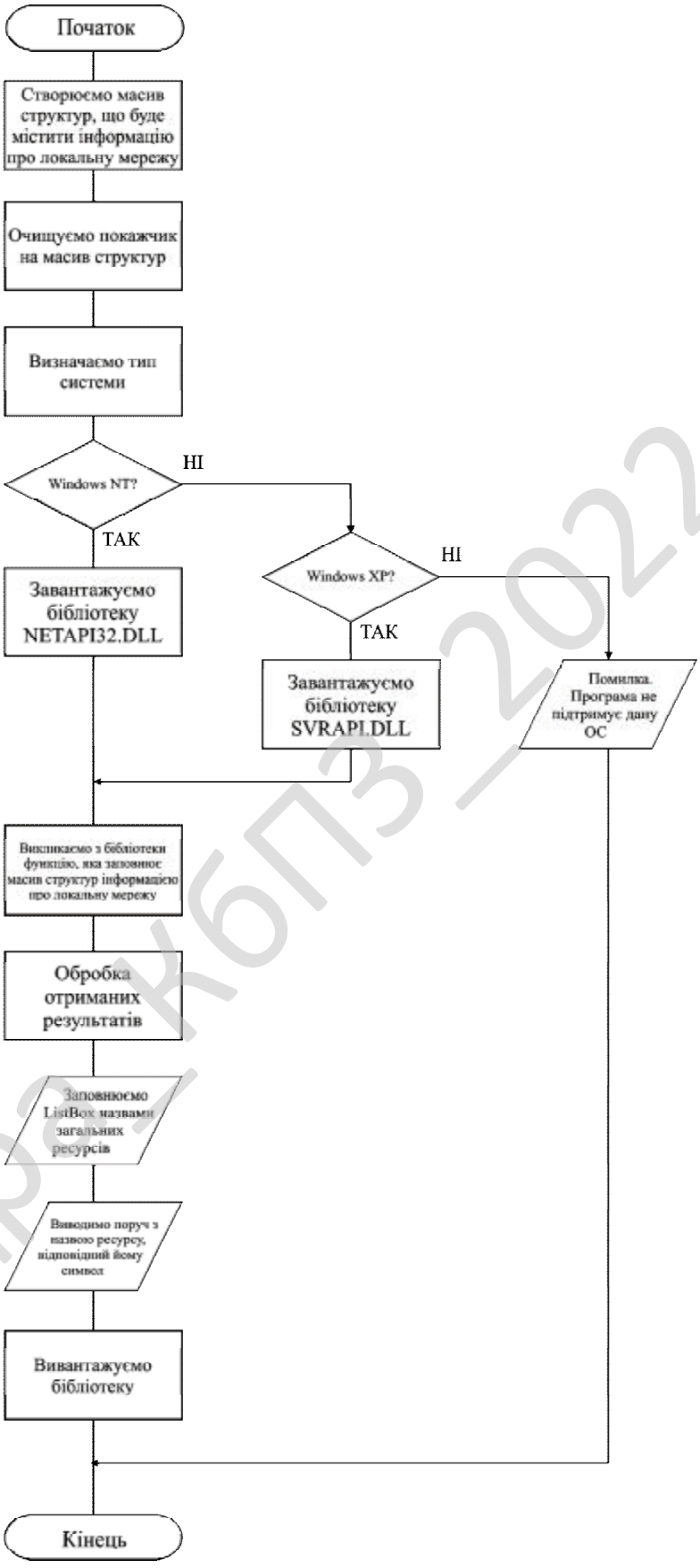

Рисунок 4.2 – Блок-схема підпрограми визначення доступних локальних ресурсів

Існує 6 типів структур передаваємих функції **NetShareEnum**:

- **SHARE_INFO_0** – тільки Windows NT.
- **SHARE_INFO_1** – тільки Windows NT.
- **share_info_1** – тільки Windows 10/11.
- **SHARE_INFO_2** – тільки Windows NT.
- **share_info_50** – тільки Windows 10/11.
- **SHARE_INFO_502** – тільки Windows NT.

Зупинимося на двох з них.

Структура share_info_50

Оголошення структури:

```
type
 TShareInfo50 = packed record
  shi50_netname         : array [0..12] of Char;
  shi50_type            : Byte;
  shi50_flags           : Word;
  shi50_remark          : PChar;
  shi50_path            : PChar;
  shi50_rw_password     : array [0..8] of Char;
  shi50_ro_password     : array [0..8] of Char;
 end;
```

Поля:

- shi50_netname – містить рядок, що утримує мережне ім'я ресурсу;
- shi50_type – визначає тип ресурсу (докладніше в MSDN);
- shi50_flags – містить інформацію про права доступу до ресурсу;
- shi50_remark – покажчик на рядок, що містить необов'язковий коментар до ресурсу;
- shi50_path – містить локальне розташування ресурсу;
- shi50_rw_password – містить пароль на запис/читання;
- shi50_ro_password – містить пароль на читання.

Реально одержати значення двох останніх полів можна тільки при одержанні інформації про свій комп'ютер, в інших випадках вони залишаються порожніми.

Структура SHARE_INFO_2

Оголошення структури:

```
type
 TShareInfo2 = packed record
  shi2_netname                 : PWChar;
  shi2_type                    : DWORD;
  shi2_remark                  : PWChar;
  shi2_permissions             : DWORD;
  shi2_max_uses                : DWORD;
  shi2_current_uses            : DWORD;
  shi2_path                    : PWChar;
  shi2_passwd                  : PWChar;
 end;
 PShareInfo2 = ^TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^TShareInfo2Array;
```

Поля:

– shi2_netname – містить покажчик на рядок, що утримує ім'я ресурсу;

– shi2_type – визначає тип ресурсу (докладніше в MSDN);

– shi2_remark – покажчик на рядок, що містить необов'язковий коментар до ресурсу;

– shi2_permissions – містить інформацію про права доступу до ресурсу;

– shi2_max_uses – визначає максимальну кулькість підключень до ресурсу;

– shi2_current_uses – визначає кулькість поточних підключень;

– shi2_path – містить покажчик на рядок утримуючий локальне розташування ресурсу;

– shi2_passwd – містить покажчик на рядок утримуючий пароль.

Закриття локального ресурсу

Тепер розглянемо функцію NetShareDel яка дозволяє закрити обраний загальний ресурс.

Оголошення функції для Windows 10/11:

```
var
 NetShareDel:function  (pszServer,
```

```
                pszNetName      :PChar;
                usReserved      :Word ): DWORD; stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо свої ресурси, то даному параметру потрібно привласнити **NIL**.

– NetName – покажчик на рядок, який містить ім'я ресурсу, що закривається.

– Reserved – не використовується, повинен бути рівним нулю.

Обидві функції не використовують ніяких структур. Нас цікавить тільки другий параметр, що містить ім'я ресурсу, що закривається. Як ім'я передається не шлях до ресурсу, а саме ім'я ресурсу яке ми визначили за допомогою коду даного вище. У випадку успішного виконання функцій, їхній результат буде рівний нулю.

Відкриття локального ресурсу

Під відкриттям локального ресурсу мається на увазі дозвіл загального доступу до нього. Блок-схема відкриття ресурсу зображена на рисунку 4.3.

Для вікриття локального ресурсу скористаємося функцією **NetShareAdd.**

Оголошення функції для Windows 10/11:

```
var
 NetShareAdd: function (      pszServer      :Pchar;
                        SLevel          :Cardinal;
                        PbBuffer        :PChar;
                        CbBuffer        :Word):DWORD; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо відкриваємо локальний ресурс, то даному параметру потрібно привласнити NIL.

– sLevel – повинен містити ідентифікатор структури.

– pbBuffer – повинен містити *покажчик* на структуру.

– cbBuffer – повинен містити розмір структури.

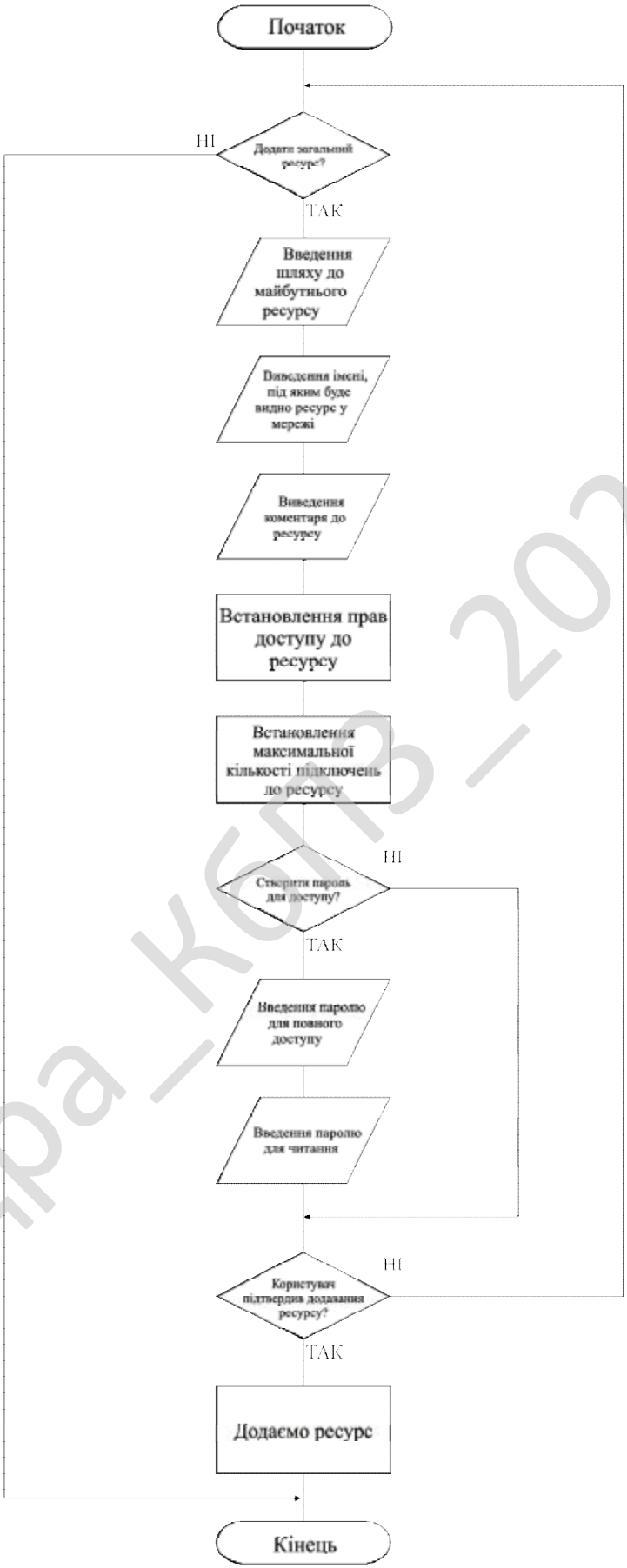

Рисунок 4.3 – Блок-схема відкриття локального ресурсу

Оголошення функції для Windows NT:

```
var
 NetShareAdd: function (       servername      : PWideChar;
                        level           : DWORD;
                        buf             : Pointer;
                        parm_err        : LPDWORD): DWORD; stdcall;
```

Параметри:

– servername – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо відкриваємо локальний ресурс то даному параметру потрібно привласнити NIL.

– level – повинен містити ідентифікатор структури.

– buf – повинен містити *покажчик* на структуру.

– parm_err – містить покажчик на член структури, що викликає помилку.

У функції для NT також використовується покажчик а не адреса покажчика. Обидві функції використовують структури описані вище.

Для демонстрації їхніх можливостей скористаємося структурами **share_info_50** і **SHARE_INFO_502**. Код, що дозволяє вибрати папку, загальний доступ до якої ми хочемо відкрити:

```
procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL = 258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до майбутнього ресурсу
  TmpName := InputBox('Share name','Enter name','Test'); //Визначаємо ім'я під яким його буде видно у мережі
  if TmpDir = '' then Exit;
```

```
if not IsNT(OS) then Close; //З'ясовуємо тип системи
if OS then begin //Код для NT
  FLibHandle := LoadLibrary('NETAPI32.DLL');
  if FLibHandle = 0 then Exit;
  @NetShareAddNT := GetProcAddress(FLibHandle,'NetShareAdd');
  if not Assigned(NetShareAddNT) then
  begin
    FreeLibrary(FLibHandle);
    Exit;
  end;
  TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір
  GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
  StringToWideChar(TmpName, TmpNameNT, TmpLength);
  ShareNT.shi2_netname := TmpNameNT; //Ім'я
  ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
  ShareNT.shi2_remark := ''; //Коментар
  ShareNT.shi2_permissions := ACCESS_READ; //Доступ
  ShareNT.shi2_max_uses := DWORD(-1); // Кількість макс. підключ.
  ShareNT.shi2_current_uses := 0; // Кількість тік подкл.
  GetMem(TmpDirNT, TmpLength);
  StringToWideChar(TmpDir, TmpDirNT, TmpLength);
  ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу
  ShareNT.shi2_passwd := nil; //Пароль
  NetShareAddNT(nil,2,@ShareNT,nil); //Додаємо ресурс
  FreeMem (TmpNameNT); //звільняємо пам'ять
  FreeMem (TmpDirNT);
end else begin //Код для 9x
  FLibHandle := LoadLibrary('SVRAPI.DLL');
  if FLibHandle = 0 then Exit;
  @NetShareAdd := GetProcAddress(FLibHandle,'NetShareAdd');
  if not Assigned(NetShareAdd) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;
  FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
  move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім'я
  Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
  Share9x.shi50_flags := SHI50F_RDONLY; //Доступ
  FillChar(Share9x.shi50_remark,
    SizeOf(Share9x.shi50_remark), #0); //Коментар
  FillChar(Share9x.shi50_path,
```

```
        SizeOf(Share9x.shi50_path), #0);
      Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
      FillChar(Share9x.shi50_rw_password,
        SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
      FillChar(Share9x.shi50_ro_password,
        SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
      NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
    end;
    FreeLibrary(FLibHandle);
  end;
```

Змінні **TmpNameNT** і **TmpDirNT** звільняються тільки після виконання функції. Це критично, у противному випадку структура передана функції буде із двома "незаповненими" полями. Також слід звернути увагу на те, що у версії коду для Windows 10/11 всі поля, що являють собою масив Char елементів, спочатку очищаються функцією **FillChar**, щоб уникнути перекручування даних (знак закінчення рядка дорівнює #0). Даний код відкриває новий ресурс на повний доступ.

Приховання й показ ресурсів

Сховати ресурс можна простим додаванням до його імені значка долара. Активувати, оберненою операцією. Це не зовсім правильно, але працює. Другий варіант сховати ресурс, це видалити його й створити його копію але із правами **SHI50F_SYSTEM** або вказівкою **shi502_security_descriptor** (для цього потрібно використовувати структуру SHARE_INFO_502).

Не можна проводити маніпуляції над наступними системними ресурсами:

IPC$, ADMIN$, PRINT$, WWWROOT$, BIOSINFO$, A$, B$, C$, D$, E$, F$, G$, H$, I$, J$, K$, L$, M$, N$, O$, P$, Q$, R$,S$, T$, U$, V$, W$, X$, Y$, Z$.

Відкриття доступу до цих ресурсів робить беззахисною Вашу операційну систему.

Одержання списку поточних сесій

Для визначення користувачів підключених до нашого комп'ютера скористаємося функцією **NetSessionEnum.**

Оголошення функції для Windows 10/11:

```
var
 NetSessionEnum:function(      pszServer      : PChar;
                        sLevel         : DWORD;
                        pbBuffer       : Pointer;
                        cbBuffer       : DWORD;
                        pcEntriesRead,
                        pcTotalAvial   : Pointer):integer; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо дивимося в себе, то даному параметру потрібно привласнити NIL.

– sLevel – повинен містити ідентифікатор структури.

– pbBuffer – повинен містити покажчик на масив структур.

– cbBuffer – повинен містити розмір масиву структур.

– pcEntriesRead – покажчик на змінну утримуючу загальну кількість структур.

– pcTotalAvial – не використовується.

Оголошення функції для Windows NT:

```
var
 NetSessionEnum:function(      ServerName,
                        UncClientName,
                        Username       : PWChar;
                        Level          : DWORD;
                        bufptr         : Pointer;
                        prefmaxlen     : DWORD;
                        entriesread,
                        totalentries,
                        resume_handle  : LPDWORD):DWORD; stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо дивимося в себе, то даному параметру потрібно привласнити NIL.

– UncClientName – містить покажчик на рядок утримуючий ім'я сесії, про

яку ми хочемо одержати інформацію, якщо потрібно переглянути всі сесії параметру потрібно привласнити NIL.

– UserName – містить покажчик на рядок утримуючий ім'я користувача, про яке ми хочемо одержати інформацію, якщо потрібно переглянути всі, параметру потрібно привласнити NIL.

– Level – повинен містити ідентифікатор структури.

– Bufptr – повинен містити *адресу покажчика* на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах, якщо не ставити обмеження, то даному параметру потрібно привласнити DWORD(-1).

– Entriesread – повинен містити покажчик на змінну, в яку запишеться кількість загальних ресурсів доступних на даний момент.

– Totalentries – не використовується.

– Resume_handle – не використовується, повинен бути NIL.

Існує також 6 типів структур, що передаються функції **NetSessionEnum**:

– **SESSION_INFO_0** – тільки Windows NT.

– **SESSION_INFO_1** – тільки Windows NT.

– **SESSION_INFO_2** – тільки Windows NT.

– **SESSION_INFO_10** – тільки Windows NT.

– **SESSION_INFO_502** – тільки Windows NT.

– **session_info_50** – тільки Windows 10/11

Розглянемо дві з них як найбільш повні по своїй суті.

Структура session_info_50

Опис структури:

```
type
 TSessionInfo50 = packed record
   sesi50_cname       : PChar;
   sesi50_username    : PChar;
   sesi50_key         : Cardinal;
   sesi50_num_conns   : Word;
   sesi50_num_opens   : Word;
```

```
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;
```

Поля:

– sesi50_cname – містить покажчик на рядок утримуючий ім'я комп'ютера, який встановив сесію.

– sesi50_username – містить покажчик на рядок утримуючий ім'я користувача, який встановив сесію.

– sesi50_key – містить значення за допомогою якого ми будемо завершувати сесію.

– sesi50_num_conns – містить число підключень зроблених під час сесії.

– sesi50_num_opens – містить кількість файлів відкритих під час сесії.

– sesi50_time – містить час в секундах, протягом якого сесія була активна.

– sesi50_idle_time – містить час у секундах протягом якого сесія була неактивна.

– sesi50_protocol – містить ім'я протоколу, за допомогою якого клієнт зв'язується із сервером.

– Pad1 – невеликий вирівнювач структури, не використовується.

Структура SESSION_INFO_502

Опис структури:

```
type
 TSessionInfo502 = packed record
    Sesi502_cname               : PWideChar;
    Sesi502_username     : PWideChar;
    Sesi502_num_opens    : DWORD;
    Sesi502_time         : DWORD;
    Sesi502_idle_time    : DWORD;
    Sesi502_user_flags   : DWORD;
    Sesi502_cltype_name  : PWideChar;
    Sesi502_transport    : PWideChar;
 End;
```

```
PSessionInfo502 = ^TSessionInfo502;
TSessionInfo502Array = array[0..512] of TSessionInfo502;
PSessionInfo502Array = ^TSessionInfo502Array;
```

Поля:

– sesi502_cname – містить покажчик на рядок утримуючий ім'я комп'ютера, який встановив сесію.

– sesi502_username – містить покажчик на рядок утримуючий ім'я користувача, який встановив сесію.

– sesi502_num_opens – містить кількість файлів відкритих під час сесії.

– sesi502_time – містить час у секундах протягом якого сесія була активна.

– sesi502_idle_time – містить час у секундах протягом якого сесія була неактивна.

– sesi50_user_flags – значення, що описує як користувач установив сесію.

– sesi502_cltype_name – тип клієнта, який встановив сесію, не використовується.

– sesi502_transport – містить ім'я протоколу, за допомогою якого клієнт зв'язується із сервером.

Розглянемо детальніше значення полів **time** і **idle_time**. Що означає коли сесія не активна? Наприклад Ви відкрили якийсь ресурс на віддаленій машині, просто подивитися, що в ньому знаходиться (наприклад імена файлів). У той час коли Ви нічого не робите, не копіюєте, не запускаєте, не відкриваєте файли, сесія не активна, вона чекає ваших дій. Як тільки Ви починаєте копіювати файл з віддаленої машини, сесія стає активною до закінчення копіювання.

Слід звернути увагу на поле **key** структури **TSessionInfo50**, воно містить унікальний ідентифікатор за допомогою якого можна завершити сесію.

Код одержання поточних сесій:

```
procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
```

```
        SessionInfo50: array [0..512] of TSessionInfo50;
        SessionInfo502 : PSessionInfo502Array;
        TotalEntries,EntriesReadNT: DWORD;
        EntriesRead,TotalAvial: Word;
        i:integer;
      begin
        lvSessions.Items.Clear;
        if not IsNT(OS) then Close; //З'ясовуємо тип системи
        if OS then begin  //Код для NT
          FLibHandle := LoadLibrary('NETAPI32.DLL');
          if FLibHandle = 0 then Exit;
          @NetSessionEnumNT := GetProcAddress(FLibHandle, 'NetSessionEnum');
          if not Assigned(NetSessionEnumNT) then
          begin
            FreeLibrary(FLibHandle);
            Exit;
          end;
          SessionInfo502 := nil;
          if              NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
          for i:=0 to EntriesReadNT-1 do
          begin
            with lvSessions.Items.Add do  //Заповнення даними зі структури
            begin
              Caption    :=    string(SessionInfo502^[i].sesi502_cname);    //Ім'я
комп'ютера
              SubItems.Add(SessionInfo502^[i].sesi502_username);           //Ім'я
користувача
       SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));   //Відкритих
ресурсів
      SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time));      //Час
активн.  SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
            end;
          end;
        end else begin //Код для Windows 10/11
         FLibHandle := LoadLibrary('SVRAPI.DLL');
         if FLibHandle <> 0 then Exit;
         @NetSessionEnum := GetProcAddress(FLibHandle, 'NetSessionEnum');
          if not Assigned(NetSessionEnum) then
          begin
            FreeLibrary(FLibHandle);
```

```
            Exit;
          end;
          if                                                    NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
          for i:=0 to EntriesRead-1 do
          begin
             with lvSessions.Items.Add do  //Заповнення даними зі структури
             begin
               Caption := string(SessionInfo50[i].Sesi50_cname); //Ім'я комп'ютера
               SubItems.Add(SessionInfo50[i].Sesi50_username);              //Ім'я
користувача
           SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens));   //Відкритих
ресурсів
       SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time));         //Час
активн.
       SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));    //Час
не активн.
               SessionCloseKey[i]:=    SessionInfo50[i].sesi50_key;    //Унікальний
ідентифікатор для закриття
             end;
           end;
         end;
         FreeLibrary(FLibHandle);
       end;
```

Ключі для закриття сесій слід заносити в масив у тому порядку, в якому були отримані самі сесії. Це зроблено для простоти. Якщо в списку, що відображає сесії, не застосовувати сортування, то порядковий номер виділеної для закриття сесії і її ідентифікатор у масиві збіжаться.

Завершення сесій

Для завершення відкритих сесій будемо використовувати функцію **NetSessionDel**.

Оголошення функції для Windows 10/11:

```
var
 NetSessionDel:function(       pszServer      : PChar;
                        PszClientName   : PChar;
                        SReserved       : SmallInt):DWORD; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо завершується сесія в себе, то даному параметру потрібно привласнити NIL.

– pszClientName – повинен містити ім'я клієнта, чия сесія завершується.

– sReserved – повинен містити унікальний ключ для завершення сесії (той який ми одержали попередньою функцією).

Оголошення функції для Windows NT:

```
var
 NetSessionDel:function(       ServerName,
                        UncClientName,
                        Username              :PWChar):DWORD; stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо завершується сесія в себе, то даному параметру потрібно привласнити NIL.

– uncClientName – повинен містити ім'я клієнта, чия сесія завершується, якщо параметр NIL, завершаться всі сесії зазначені в параметрі username.

– username – повинен містити ім'я користувача, чия сесія завершується, якщо параметр NIL, завершаться всі сесії зазначені в параметрі uncClientName.

Як можна помітити, ніяких структур дані функції не використовують. Напишемо процедуру яка буде завершувати обрану нами в lvSessions сесію по ім'ю клієнта. От як цей код виглядає:

```
procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З'ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
```

```
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії
  if OS then begin
    FLibHandle := LoadLibrary('NETAPI32.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, 'NetSessionDel');
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString('\\'+lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary('SVRAPI.DLL');
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, 'NetSessionDel');
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;
```

Одержання списку відкритих файлів

Розглянемо функцію **NetFileEnum**.

Оголошення функції для Windows 10/11:

```
var
 NetFileEnum:function(  pszServer,
                pszBasePath    : PChar;
                sLevel         : DWORD;
                pbBuffer       : Pointer;
                cbBuffer       : DWORD;
                pcEntriesRead,
                pcTotalAvail   : Pointer):Integer; stdcall;
```

Параметри:

– pszServer – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо дивимося в себе, то даному параметру потрібно привласнити NIL.

– pszBasePath – повинен містити каталог, відкриті файли з якого ми хочемо подивитися, якщо ми хочемо переглянути всі відкриті файли параметру потрібно привласнити NIL.

– sLevel – повинен містити ідентифікатор структури.

– pbBuffer – повинен містити покажчик на масив структур.

– cbBuffer – повинен містити розмір структури.

– pcEntriesRead – покажчик на змінну утримуючу загальну кількість структур.

– pcTotalAvial – не використовується.

Оголошення функції для Windows NT:

```
var
 NetFileEnum:function(  servername,
                basepath,
                username     : PWChar;
                level        : DWORD;
                bufptr       : Pointer;
                prefmaxlen   : DWORD;
                entriesread,
                totalentries,
                resume_handle   : LPDWORD):DWORD; stdcall;
```

Параметри:

– ServerName – повинен містити ім'я віддаленого комп'ютера, на якому повинна виконається функція, якщо дивимося в себе, то даному параметру потрібно привласнити NIL.

– BasePath – повинен містити каталог, відкриті файли з якого ми хочемо подивитися, якщо ми хочемо переглянути всі відкриті файли параметру потрібно привласнити NIL.

– UserName – містить покажчик на рядок утримуючий ім'я користувача,

про яке ми хочемо одержати інформацію, якщо потрібно переглянути всі, параметру потрібно привласнити NIL.

– Level – повинен містити ідентифікатор структури.

– Bufptr – повинен містити адресу покажчика на масив структур.

– Prefmaxlen – повинен містити максимальну довжину повернутих даних у байтах, якщо не ставити обмеження те даному параметру потрібно привласнити DWORD(-1).

– Entriesread – повинен містити покажчик на змінну, в яку запишеться кількість загальних ресурсів доступних на даний момент.

– Totalentries – не використовується.

– Resume_handle – не використовується, повинен бути NIL.

Існує три типи структур, що використовує дана функція:

– **FILE_INFO_2** – використовується під Windows NT.

– **FILE_INFO_3** – використовується під Windows NT.

– **file_info_50** – використовується під Windows 10/11.

Розглянемо дві з них.

Структура file_info_50

Опис структури:

```
type
 TFileInfo50 = packed record
   fi50_id          : Cardinal;
   fi50_permissions : WORD;
   fi50_num_locks   : WORD;
   fi50_pathname    : PChar;
   fi50_username    : PChar;
   fi50_sharename   : PChar;
 end;
```

Поля:

– Fi50_id – містить ідентифікаційний номер відкритого файлу.

– Fi50_permissions – містить рівень доступу, з яким відкритий файл.

– Fi50_num_locks – містить кількість блокувань файлу.

– Fi50_pathname – містить повний шлях до відкритого файлу.

– Fi50_username – містить ім'я користувача або комп'ютера открывшего файл.

– Fi50_sharename – містить ім'я загального ресурсу, в якому перебуває відкритий файл.

Структура FILE_INFO_3

Опис структури:

```
type
 TFileInfo3 = packed record
   fi3_id          : DWORD;
   fi3_permissions : DWORD;
   fi3_num_locks   : DWORD;
   fi3_pathname    : PWChar;
   fi3_username    : PWChar;
 end;
 PFileInfo3 = ^TFileInfo3;
 TFileInfo3Array = array[0..512] of TFileInfo3;
 PFileInfo3Array = ^TFileInfo3Array;
```

Поля:

– Fi3_id – містить ідентифікаційний номер відкритого файлу.

– Fi3_permissions – містить рівень доступу, з яким відкритий файл.

– Fi3_num_locks – містить кількість блокувань файлу.

– Fi3_pathname – містить повний шлях до відкритого файлу.

– Fi3_username – містить ім'я користувача або комп'ютера, який відкрив файл.

Тепер напишемо код який покаже нам список відкритих файлів на нашому комп'ютері:

```
procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
```

```
        begin
          lvfiles.Items.Clear;
          if not IsNT(OS) then Close; //З'ясовуємо тип системи

          if OS then begin  //Код для NT
            FLibHandle := LoadLibrary('NETAPI32.DLL');
            if FLibHandle = 0 then Exit;
            @NetFileEnumNT := GetProcAddress(FLibHandle, 'NetFileEnum');
            if not Assigned(NetFileEnumNT) then
            begin
              FreeLibrary(FLibHandle);
              Exit;
            end;
            FileInfoNT := nil;
            if     NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
            for i:=0 to EntriesReadNT-1 do
            begin
              with lvFiles.Items.Add do  //Заповнення даними зі структури
              begin
                Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
                SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
                SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім'я користувача
              end;
            end;
          end else begin //Код для Windows 10/11
           FLibHandle := LoadLibrary('SVRAPI.DLL');
           if FLibHandle = 0 then Exit;
           @NetFileEnum := GetProcAddress(FLibHandle, 'NetFileEnum');
            if not Assigned(NetFileEnum) then
            begin
              FreeLibrary(FLibHandle);
              Exit;
            end;
            if                              NetFileEnum                              (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial) = 0 then
            for i:=0 to EntriesRead-1 do
            begin
              with lvFiles.Items.Add do  //Заповнення даними зі структури
              begin
                Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
                SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
```

```
          SubItems.Add(FileInfo9x[i].fi50_username); //Ім'я користувача
        end;
      end;
    end;
    FreeLibrary(FLibHandle);
  end;
```

Закриття відкритого файлу

Тепер розглянемо функції **NetFileClose** і **NetFileClose2**, за допомогою яких будемо закривати відкриті по мережі файли.

Функція **NetFileClose2** – використовується тільки в Windows 10/11:

```
var
 NetFileClose2:function(       pszServer      :PChar;
                UlFileId       :LongWord):DWORD; stdcall;
```

Поля:

– pszServer – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо в себе, то даному параметру потрібно привласнити NIL.

– UlFileId – повинен містити унікальний ідентифікатор відкритого файлу.

Функція **NetFileClose** – використовується тільки в Windows NT:

```
var
 NetFileClose:function( ServerName     :PWideChar;
                FileId  :DWORD):DWORD; stdcall;
```

Поля:

– ServerName – повинен містити ім'я віддаленого комп'ютера, якщо закриваємо в себе, то даному параметру потрібно привласнити **NIL**.

– FileId – повинен містити унікальний ідентифікатор відкритого файлу.

Як перший параметр передамо NIL (закриваємо в себе), у якості другого, ідентифікатор файлу, що відображається у полі ID. От приклад коду:

```
procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З'ясовуємо тип системи
  if not Assigned(lvFiles.Selected) then Exit;
```

```
        i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу
        if OS then begin //Код для NT
          FLibHandle := LoadLibrary('NETAPI32.DLL');
          if FLibHandle = 0 then Exit;
          @NetFileClose := GetProcAddress(FLibHandle, 'NetFileClose');
          if not Assigned(NetFileClose) then
          begin
            FreeLibrary(FLibHandle);
            Exit;
          end;
          NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption)); //Закриваємо
        end else begin //Код для Windows 10/11
          FLibHandle := LoadLibrary('SVRAPI.DLL');
          if FLibHandle = 0 then Exit;
          @NetFileClose2 := GetProcAddress(FLibHandle, 'NetFileClose2');
          if not Assigned(NetFileClose2) then
          begin
            FreeLibrary(FLibHandle);
            Exit;
          end;
          NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
//Закриваємо
        end;
        FreeLibrary(FLibHandle);
      end;
```

Визначення вхідного та вихідного трафіку

Для цього досить використовувати всього лише одну функцію бібліотеки IPHLPAPI.DLL, що поставляється з усіма версіями Windows. Оголошення функції (всі версії Windows):

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

Параметри:

- pIfTable – повинен містити покажчик на структуру.
- pdwSize – повинен містити розмір структури.
- bOrder – указує, чи потрібне сортування в масиві, що повертається.

Як перший параметр функція використовує покажчик на структуру. Опис структури:

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

Поля:

– dwNumEntries – визначає розмірність масиву представленого другим параметром.

– Table – є масивом структур.

Структура сама по собі вкрай неінформативна, нас цікавить друге її поле, що також представляє собою структуру. Опис структури:

```
type
  TMibIfRow = packed record
    wszName             : array[0..255] of WideChar;
    dwIndex             : DWORD;
    dwType              : DWORD;
    dwMtu               : DWORD;
    dwSpeed             : DWORD;
    dwPhysAddrLen       : DWORD;
    bPhysAddr           : array[0..7] of Byte;
    dwAdminStatus       : DWORD;
    dwOperStatus        : DWORD;
    dwLastChange        : DWORD;
    dwInOctets          : DWORD;
    dwInUcastPkts       : DWORD;
    dwInNUCastPkts      : DWORD;
    dwInDiscards        : DWORD;
    dwInErrors          : DWORD;
    dwInUnknownProtos   : DWORD;
    dwOutOctets         : DWORD;
    dwOutUCastPkts      : DWORD;
    dwOutNUCastPkts     : DWORD;
    dwOutDiscards       : DWORD;
    dwOutErrors         : DWORD;
    dwOutQLen           : DWORD;
```

```
    dwDescrLen          : DWORD;
    bDescr              : array[0..255] of Char;
  end;
  TMibIfArray = array [0..512] of TMibIfRow;
  PMibIfRow = ^TMibIfRow;
  PmibIfArray = ^TmibIfArray;
```

Поля:

– wszName – покажчик на рядок утримуючий ім'я інтерфейсу.

– dwIndex – визначає індекс інтерфейсу.

– dwType – визначає тип інтерфейсу.

– dwMtu – визначає максимальну швидкість передачі.

– dwSpeed – визначає поточну швидкість передачі в бітах у секунду.

– dwPhysAddrLen – визначає довжину адреси, що втримується в bPhysAddr.

– bPhysAddr – містить фізичну адресу інтерфейсу (його трохи видозмінений MAC-адрес).

– dwAdminStatus – Визначає активність інтерфейсу.

– dwOperStatus – містить поточний статус інтерфейсу.

– dwLastChange – містить останній змінений статус.

– dwInOctets – містить кількість байтів прийнятих через інтерфейс.

– dwInUcastPkts – містить кількість спрямованих пакетів прийнятих інтерфейсом.

– dwInNUCastPkts – містить кількість ненаправлених пакетів прийнятих інтерфейсом.

– dwInDiscards – містить кількість забракованих вхідних пакетів (навіть якщо вони не містили помилки).

– dwInErrors – містить кількість вхідних пакетів утримуючих помилки.

– dwInUnknownProtos – містить кількість забракованих вхідних пакетів зі структурою невідомого протоколу.

– dwOutOctets – містить кількість байтів відправлених інтерфейсом.

– dwOutUCastPkts – містить кількість спрямованих пакетів відправлених інтерфейсом.

– dwOutNUCastPkts – містить кількість ненаправлених пакетів відправлених інтерфейсом.

– dwOutDiscards- містить кількість забракованих вихідних пакетів (навіть якщо вони не містили помилки).

– dwOutErrors – містить кількість вихідних пакетів утримуючих помилки.

– dwOutQLen – містить довжину черги даних.

– dwDescrLen – містить розмір масиву bDescr.

– bDescr – містить опис інтерфейсу.

В цій структурі утримується безліч інформації, що ми й будемо використовувати.

Код визначення поточного вхідного/вихідного трафіку виглядає наступним чином:

```
procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC-адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := ' 00-00-00' else
    begin
      Result := '';
      for i:= 0 to Length -2 do
        Result := Result + IntToHex(Value[i],2)+'-';
     Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle    : THandle;
  Table         : TMibIfTable;
```

```
    i             : Integer;
    Size          : Integer;
  begin
    tmrTraffic.Enabled := False; //Зупиняємо таймер
    lvTraffic.Items.BeginUpdate;
    lvTraffic.Items.Clear;  //Очищаємо список
    FLibHandle := LoadLibrary('IPHLPAPI.DLL'); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    @GetIfTable := GetProcAddress(FLibHandle, 'GetIfTable');
    if not Assigned(GetIfTable) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    Size := SizeOf(Table);
    if GetIfTable(@Table, @Size, False ) = 0 then //Виконуємо функцію
      for i:= 0 to Table.dwNumEntries-1 do begin
        with lvTraffic.Items.Add do begin //Виводимо результати
          Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
          SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
            Table.Table[i].dwPhysAddrLen)); //MAC адреса
          SubItems.Add(IntToStr(Table.Table[i].dwInOctets));          //Усього
прийнято байтів
          SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));         //Усього
відправлено байтів
        end;
      end;
    lvTraffic.Items.EndUpdate;
    FreeLibrary(FLibHandle);
    tmrTraffic.Enabled := True; //Активуємо таймер
  end;
```

Тип даних TMAC було створено для передачі масиву, у якому втримується сам MAC адресу у функцію для перетворення його в більш звичний вигляд.

Код TMAC(Table.Table[i].bPhysAddr) – це передача масиву, обов'язково потрібно вказати, що масив передається як тип TMAC, у протилежному випадку компілятор видасть помилку несумісності типів.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Blowfish, який є симетричним алгоритмом шифрування, тобто таким, у якому ключ шифрування дорівнює ключу дешифрування. Він є мережею Фейштеля, у якій кількість ітерацій дорівнює 16. Довжина блоку дорівнює 64 бітам, ключ може мати будь-яку довжину в межах 448 біт. Хоча перед початком будь-якого шифрування виконується складна фаза ініціалізації, саме шифрування даних виконується досить швидко.

Алгоритм призначений в основному для додатків, у яких ключ міняється нечасто, до того ж існує фаза початкового рукостискання, під час якої відбувається автентифікація сторін і узгодження загальних параметрів і секретів. При реалізації на 32-бітних мікропроцесорах з більшим кешем даних Blowfish значно швидше DES.

Алгоритм складається із двох частин: розширення ключа й шифрування даних. Розширення ключа перетворює ключ довжиною, принаймні, 448 біт у кілька масивів підключів загальною довжиною 4168 байт.

В основі алгоритму лежить мережа Фейштеля з 16 ітераціями. Кожна ітерація складається з перестановки, що залежить від ключа, і підстановки, що залежить від ключа й даних. Операціями є XOR і додавання 32-бітних слів.

Blowfish використовує велику кількість підключів. Ці ключі повинні бути обчислені заздалегідь, до початку будь-якого шифрування або дешифрування даних. Елементи алгоритму:

1. P – масив, що складається з вісімнадцяти 32-бітних підключів:

$$P_1, P_2, ..., P_{18}.$$

2. Чотири 32-бітних S-boxes с 256 входами кожний. Перший індекс означає номер S-box, другий індекс – номер входу.

$$S_{1,0}, S_{1,1}, \ldots S_{1,255};$$
$$S_{2,0}, S_{2,1}, \ldots S_{2,255};$$
$$S_{3,0}, S_{3,1}, \ldots S_{3,255};$$
$$S_{4,0}, S_{4,1}, \ldots S_{4,255};$$

Шифрування

Входом є 64-бітний елемент даних X, що ділиться на дві 32-бітні половини, X_l і X_r.

$$X_l = X_l \text{ XOR } P_i$$

$$X_r = F(X_l) \text{ XOR } X_r$$

$$\text{Swap } X_l \text{ and } X_r$$

Функція F

Розділити X_l на чотири 8-бітних елементи A, B, C, D.

$F(X_l) = ((S_{1,A} + S_{2,B} \bmod 2^{32}) \text{ XOR } S_{3,C}) + S_{4,D} \bmod 2^{32}$

Дешифрування відрізняється від шифрування тим, що P_i використовуються у зворотному порядку.

Генерація підключів

Підключи обчислюються з використанням самого алгоритму Blowfish.

1. Ініціалізувати перший P-масив і чотири S-boxes фіксовані рядки.

2. Виконати операцію XOR P_1 з першими 32 бітами ключа, операцію XOR P_2 із другими 32 бітами ключа й т.д. Повторювати цикл доти, поки весь P-масив не буде побітово складний з усіма бітами ключа. Для коротких ключів виконується конкатенація ключа із самим собою.

3. Зашифрувати нульовий рядок алгоритмом Blowfish, використовуючи підключи, описані в пунктах (1) і (2).

4. Замінити P_1 і P_2 виходом, отриманим на кроці (3).

5. Зашифрувати вихід кроку (3), використовуючи алгоритм Blowfish з модифікованими підключами.

6. Замінити P_3 і P_4 виходом, отриманим на кроці (5).

7. Продовжити процес, заміняючи всі елементи P-масиву, а потім всі чотири S-boxes, виходами відповідним чином модифікованого алгоритму Blowfish.

Для створення всіх підключів потрібна 521 ітерація.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Програма має простий та інтуїтивно зрозумілий інтерфейс, який зображений на рисунку 5.1

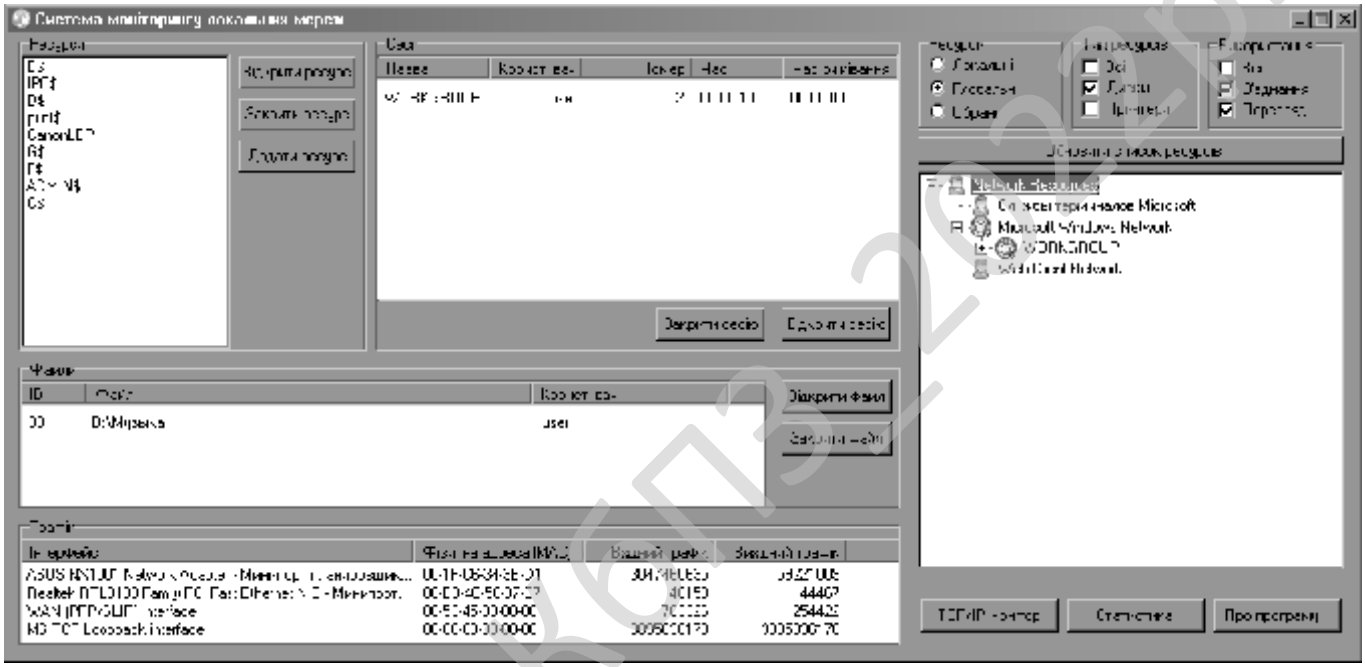

Рисунок 5.1 – Головне вікно програми

Як видно з рисунку, користувач може переглянути ресурси мережі, відкриті сесії, відкриті користувачами мережі файли, мережні інтерфейси, вхідний та вихідний трафік.

Також користувач розробленої програми може виконувати наступні дії:

- Відкрити ресурс.
- Закрити ресурс.
- Додати ресурс.
- Обновити список ресурсів.
- Закрити сесію.

– Відкрити сесію.

– Відкрити файл.

– Закрити файл.

– Ввімкнути TCP/IP монітор.

– Переглянути статистику з'єднань.

– Переглянути довідку.

На рисунку 5.2 показане вікно монітору TCP/IP з'єднань. Для його виклику слід натиснути кнопку " TCP/IP монітор".

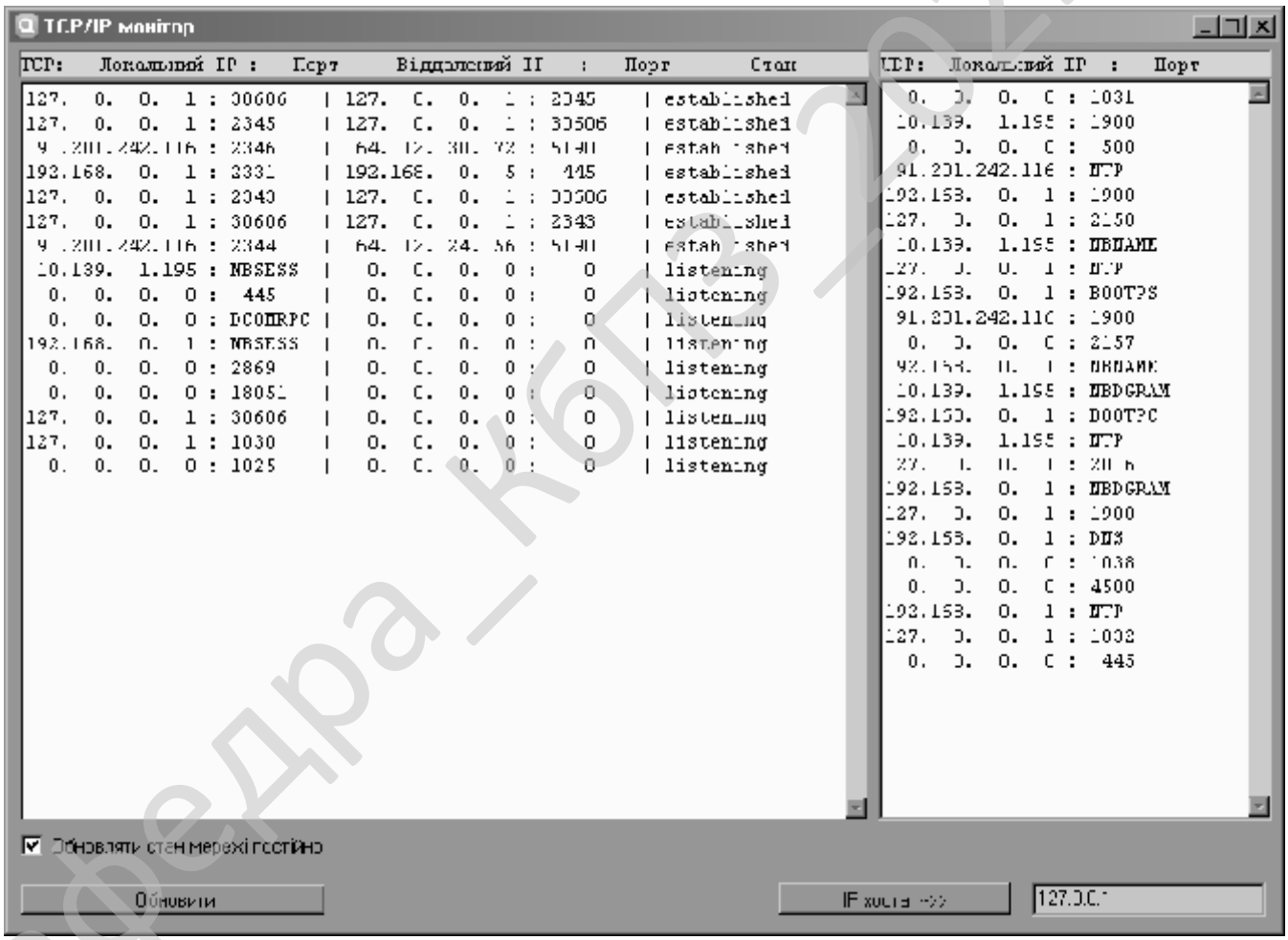

Рисунок 5.2 – Монітор TCP/IP з'єднань

Монітор TCP/IP з'єднань дозволяє переглядати всі з'єднання, що відбуваються у мережі.

Монітор складається з двох полів, в першому показані з'єднання по TCP-протоколу, в тругому – по UDP-протоколу.

На рисунку 5.3 зображене вікно статистики. Для його виклику слід натиснути кнопку "Статистика".

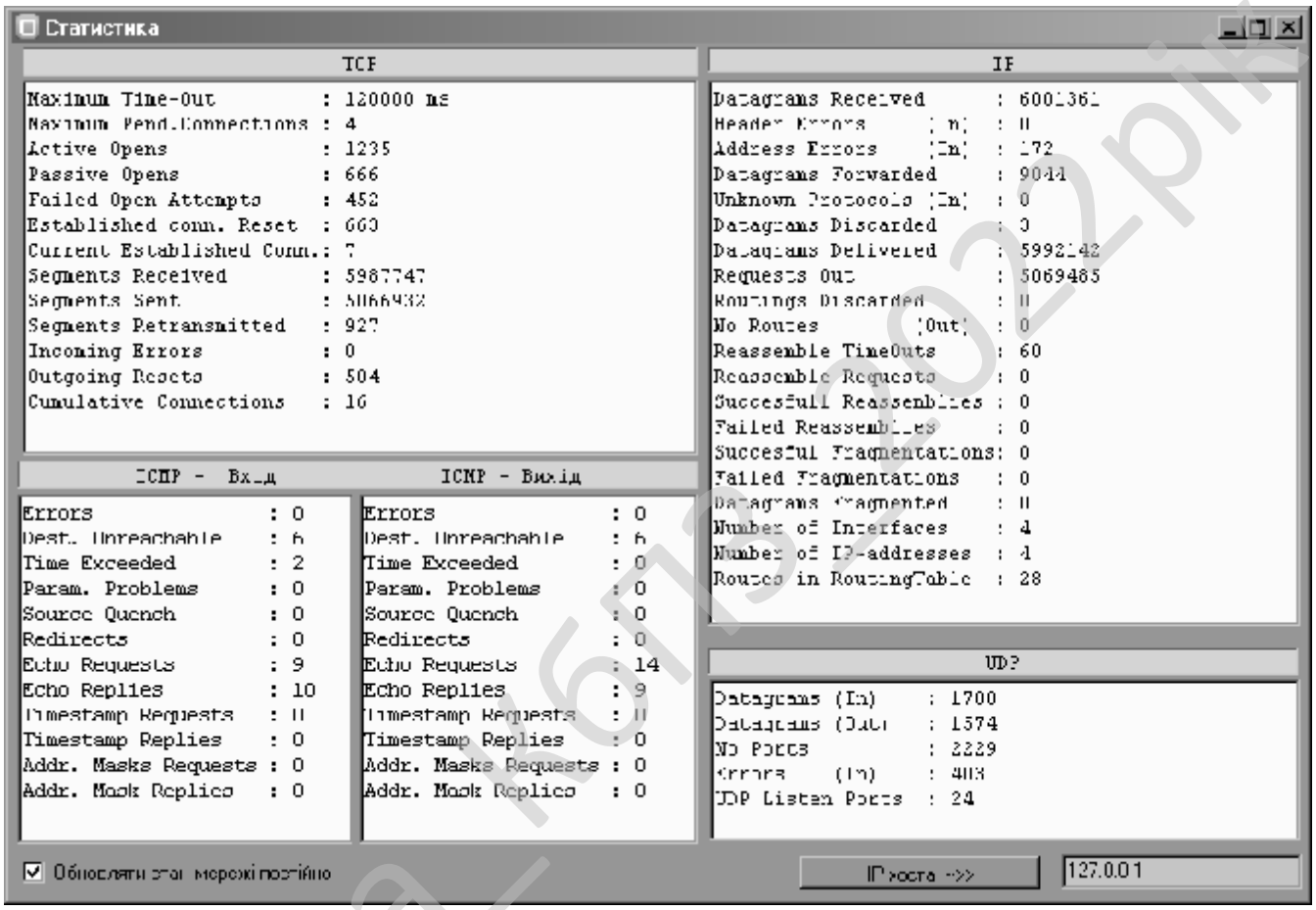

Рисунок 5.3 – Статистика

Вікно статистики складається з п'яти полів:

- Поле статистики подій по TCP-протоколу.
- Поле статистики подій по IP-протоколу.
- Поле статистики подій по UDP-протоколу.
- Поле статистики вхідних подій по ICMP-протоколу.
- Поле статистики вихідних подій по ICMP-протоколу.

Коротку довідку про розроблену систему можна переглянути натиснувши кнопку "Про програму…". Після натиснення з'явиться вікно зображене на рисунку 5.4.

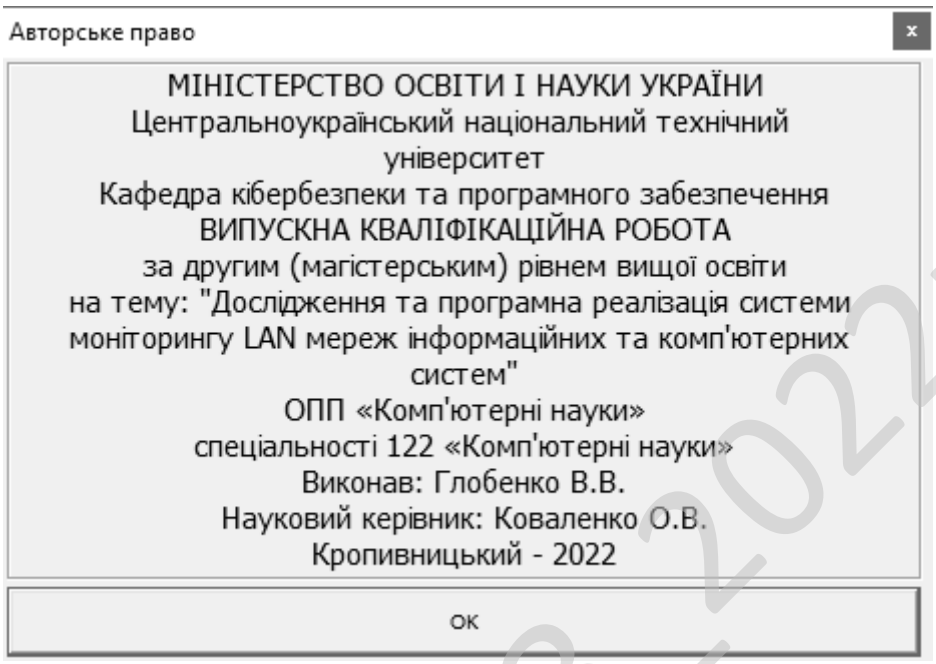

Рисунок 5.4 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Метою розробки є дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Об'єктом дослідження є процес моніторингу LAN мереж інформаційних та комп'ютерних систем.

Предметом дослідження є методи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Методи дослідження базуються на методах побудови мереж інформаційних та комп'ютерних систем, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод моніторингу LAN мереж інформаційних та комп'ютерних систем.

– Розроблено вітчизняний продукт моніторингу LAN мереж інформаційних та комп'ютерних систем, 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 моніторингу LAN мереж інформаційних та комп'ютерних систем. 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	80
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)	–	3
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.	–	80000
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.:

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

де: $ПV_j$ – 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)%.

$$T_{РП} = 0{,}3 \cdot 2{,}85 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 180 \text{ люд/день.}$$

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	180	Ф 7.1-7.4
Впровадження	13	Д13
Всього	221	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

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

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

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

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

$$Ч = \frac{221 \cdot 1}{60 - 5} = 4 \text{ ставки.}$$

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	12	1080	18
Монітор	60	12	720	12
Клавіатура	30	12	360	6
Маніпулятор «мишка»	10	12	120	2
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	2	40	0,67
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛВС на 1 м. п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	57,92

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{тіс}}{1{,}2}, \tag{7.6}$$

$$Ф_{др}^{с} = \frac{58 \cdot 3}{1{,}2} = 145 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \tag{7.7}$$

$$Ч_{ел} = 145/(60 \cdot 8) = 0{,}3 \text{ ставки.}$$

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

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.

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 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	1	0,25
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,25	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,25	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	0,5	
Всього		2	

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	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,4	
Всього		1,6	
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	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	7870	23610
Продакт-менеджер	0,25	7000	5250
Інженер-програміст	4	7000	84000
Інженер-електронщик	0,3	7000	6300
Інженер-системотехнік	0,25	7000	5250
Адміністратор мережі	0,25	7000	5250
Системний програміст	0,25	7000	5250
Дизайнер WEB	0,2	7000	4200
Інженер-верстальник	0,25	7000	5250
Бухгалтер-економіст	0,5	7000	10500
Всього за період розробки	$R_{сп}$ = 7,25	-	$Ф_{роб}$ = 154860

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.

$$З_{сд} = \frac{154860}{8,25 \cdot 60} = 356 \; грн.$$

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 за прайсом Інтернет магазину Компбест за 14.11.2022 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		Acer X2632G DT
Процесор	Intel Core i3-4130 (2(4) ядра по 3.40 GHz)	-
Системна плата	X4660G VX4660G-I5840H1 4x USB 2.0, 2x USB 3.0, 5x Audio Ports, Com Port, 2x PS/2, LAN (RJ-45); DVI, HDMI	-
Відеокарта	AMD Radeon RX 550 4GB GDDR5 Re Dragon PowerColor (AXRX 550 4GBD5 DH)	-
Жорсткий диск	HDD Seagate Barracuda 750 Gb 7200 32Mb SATAII ST3750528AS (ST3750528AS) + 120 GB SSD	-
Оперативна пам’ять	DIMM 4096Mb DDR3 PC3-10600 CL9 Transcend JetRam, non-Reg., no-ECC , CL 9 (2 модулі)	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bulk 22x, SecurDisc, black	-
Корпус	GRESSO GE-7525, 500W (120mm big fan), 2xIDE, full-ATX,БЖ 2xSATA, 1xFDD, Air Duct, 2xUSB 2.0, Mic+Audio, silver/black	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
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) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

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 7.7 – 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	1	968	96,8	1064,8
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	200241,8

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	200242	-	-
Всього по групі	200242	50	100121
Нематеріальні активи			
4. Нематеріальні активи	80000	10	8000
Група 5, 6			
5. Вимірювальні пристрої	9031	25	2257,75
6. Транспортні засоби	122500	20	24500
7. Господарський інвентар	28000	25	7000
Всього по групі	159531	-	20000
Разом	$К_р$ = 1847773		$А_р$ = 198521

Примітка: вартість автомобіля (Suzuki GSX-R 600) взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http://www.auto-trading.com.ua/sale/lot69649.html, складає 3224 USD, що враховуючи курс 38 складає 122500 грн.

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.

$$З_о = 356 \cdot 209/80 = 930 \text{ грн.}$$

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, %.

$$З_д = 930 \cdot 10 \cdot 0{,}01 = 93 \text{ грн.}$$

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, %.

$$С_{оц} = 0{,}01 \cdot 22(930+93) = 225 \text{ грн.}$$

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, %.

$$Г_{осп} = 930 \cdot 15 \cdot 0{,}01 = 140 \text{ грн.}$$

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:

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

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

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

$$З_{М2} = \sum Ц_д\ , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: – 24 грн./шт.

$$З_{М2} = 40 \cdot 24 = 960 \text{ грн.}$$

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.

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

$$З_М = (105 + 960 + 1702)/80 = 34 \text{ грн.}$$

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, %.

$$О_п = 930 \cdot 15 \cdot 0{,}01 = 140 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 80 прим.):

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

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

$$A_м = 198521 \cdot 3/(80 \cdot 12) = 620 \text{ грн.}$$

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	$З_о$	930
2. Додаткова зарплата виконавців	$З_д$	93
3. Відрахування на соціальні потреби	$С_{оц}$	225
4. Загальногосподарські витрати	$Г_{осп}$	140
5. Витрати на матеріали	$З_М$	34
6. Освоєння нових операційних систем, мов програмування	$О_п$	140
7. Амортизація основних фондів	$А_м$	620
8. Повна собівартість програмного забезпечення	$С_п$	2182
9. Плановий прибуток	$П_р$	1091
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	3273
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	654,6
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	3927,6

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

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

$$C_п = 930+93+225+140+34+140+630 = 2182 \text{ грн.}$$

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, %.

$$П_р = 0{,}01 \cdot 50 \cdot 2182 = 1091 \text{ грн.}$$

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 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	3928
Всього капітальних витрат	–	3928

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 відновлення працездатності системи після збоїв та профілактичні роботи	$З_р$	21901	5475
2. Витрати на амортизацію	$З_{ам}$	0	982
Всього витрат за рік	$І$	21901	6457

Витрати на відновлення працездатності системи після збоїв та профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування сегменту мережі за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 60 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 68 \cdot 1{,}1 \cdot 1{,}22 = 21901\ грн,$$

до:

$$З_{р\,нов} = 60 \cdot 68 \cdot 1{,}1 \cdot 1{,}22 = 5475\ грн.$$

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	–	3928	–	982
Всього відрахувань	-	–	3928	–	982

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:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.24)$$

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

$$Ев = (3273 - 2182) \cdot 80 - (0,05 \cdot 1408000 + 0,3 \cdot 200242 +$$
$$+0,25 \cdot 37031 + 0,2 \cdot 122500 + 0,1 \cdot 80000) \cdot 3/12 = 44222 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.25)$$

де: $K_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{439773}{(3273 - 2182) \cdot 80 \cdot 12 / 3} = 1,3 \text{ років}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1	2	3
1. Кількість екземплярів програми	Прим.	80
2. Повна собівартість розробленої програми	Грн.	2182
3. Ціна розробленої програми	Грн.	3273
4. Плановий прибуток від реалізації розробленої програми	Грн.	1091
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1847773
7. Загальний прибуток від реалізації програмної продукції	Грн.	87280
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	44222
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,3
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	3928
11.Величина економічного ефекту у користувача програмної продукції	Грн.	14462
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,25

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.26)$$

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.

$$E_{сп} = (21901\text{-}6457)\text{-}0{,}25 \cdot 3928 = 14462 \text{ грн.}$$

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \qquad (7.27)$$

$$T_{сп} = \frac{3928}{21901 - 6457} = 0,25 \text{ року}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

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

З давніх давен людство приділяє прискіпливу увагу безпеці життя і охорони праці як її складової частини. Умови праці розглядали Арістотель (384-322 до н.е.) та Гіппократ (460-377 до н.е.) [4].

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.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців 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 [5], та «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

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 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3,9
Довжина	5,2
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	7,76
Об'єм, V	$м^3$	не менше 20.0	20,28

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють троє людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5],

але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	46-54	0,10
Тепла	23-25	50-70	0,1	24-25	50-65	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], 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 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 поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

8.5 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3,9 м, довжина – 5,2 м, висота – 3 м.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, необхідно враховувати, що, з одного боку «середня освітленість робочих місць з постійним перебуванням людей повинна бути не менш як 200 люкс», а з іншого «штучне освітлення при системі комбінованого освітлення для зорової роботи найвищої точності повинна складати з00 люкс» та вимогу ДБН В.2.5-28:2018, що “створювати освітленість більше ніж 300 лк при світлодіодних світильникх дозволяється тільки за наявності обґрунтування».

Таким чином для розрахунку приймаэмо середню освітленість робочих місць 300 Люкс [1].

Визначимо світловий потік, що падає на робочу поверхню за формулою [6]:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк. [1];

S – площа поверхні, на яку падає світловий потік, м^2 [1];

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

K – коефіцієнт запасу, що враховує зменшення світлового потоку

світильники в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5) [1];

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50% [6].

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа поверхні, на яку падає світловий потік, м^2 [1];

h – розрахункова висота підвісу, h = 3 м;

A – ширина приміщення, A = 3,9 м;

B – довжина приміщення, B = 5,2 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=0,74.$$

Знаючи індекс приміщення, за знаходимо n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу [6]). Підставимо всі значення у формулу, визначемо світловий потік: *F=43646 Лм.*

Будемо використовувати світильники (світлодіодні панелі) EVROLIGHT ОПАЛ-40 6400К 3000 Лм., світловий потік яких $F_л$ = 3000 Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік одного світильника.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$N = 43646 / 3000 = 14{,}54 \text{ шт.}$$

Приймаємо необхідну кількість світильників 15 шт.

8.6 Висновки до розділу

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

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. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

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 моніторингу LAN мереж інформаційних та комп'ютерних систем.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем моніторингу LAN мереж інформаційних та комп'ютерних систем.

– Досліджена система моніторингу LAN мереж інформаційних та комп'ютерних систем.

– На основі отриманих результатів досліджень створена програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання моніторингу LAN мереж інформаційних та комп'ютерних систем.

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 Embarcadero RAD Studio Delphi 10.3.2. 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 Blowfish.

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 14462 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,25 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Глобенко В.В. Дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» *CEUR Workshop Proceedings,* Volume 3187, 2022, pp. 1-12. **(Scopus).**

3. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». *Sensors (Basel, Switzerland)* Volume 22, Issue 16, 6223, 2022. (**Scopus**).

4. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

5. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

6. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». *SN Computer Science,* Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w **(Scopus).**

7. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th*

International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

8. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

9. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

10. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». *International Journal of Computing*; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256. **(Scopus)**.

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

12. Smirnov O.A., 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. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

14. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 1-14. **(Scopus)**.

15. Smirnov O., Lutsenko M., Kuznetsov A., Kiian A., Kuznetsova T., «Biometric cryptosystems: overview, state-of-the-art and perspective directions».

Lecture Notes in Networks and Systems, vol 152. **Springer**, Cham. 2021, pp 66-84. **(Scopus)**.

16. 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. **(Scopus)**.

17. 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. **(Scopus**).

18. 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. **(Scopus)**.

19. Smirnov, O., Shekhanin, K., Kuznetsov, A., Krasnobayev, V. «Detecting Hidden Information in FAT». *International Journal of Computer Network and Information Security (IJCNIS)*. Vol. 12, No. 3, 2020. PP.33-43. **(Scopus)**.

20. 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. **(Scopus)**.

21. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 646-660., **(Scopus)**.

22. 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. **(Scopus)**.

23. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during

Information Campaign Based on the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, Vol 2588, P. 215-227, 2019. **(Scopus)**.

24. 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. **(Scopus)**.

25. 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. **(Scopus)**.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Kuznetsova, K., Ivko, T., Prokopovych-Tkachenko, D., «Soft Decoding Based on Ordered Subsets of Verification Equations of Turbo-Productive Codes», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 873-884. **(Scopus)**.

27. Smirnov, O., Kuznetsov, A., Prokopovych-Tkachenko, D. «Hiding Data in Images Using a Pseudo-Random Sequence». *ISCI'2020: Information Security in Critical Infrastructures. **Collective monograph***. Edited by Ivan D. Gorbenko, Victor A. Krasnobayev and Alexandr A. Kuznetsov. ASC Academic Publishing, USA, 2020. pp. 46-59. – ISBN: 978-1-7362833-0-1 (Hardback), ISBN: 978-1-7362833-1-8 (Ebook).

28. Smirnov, O., Kuznetsov, A., Shekhanin, K., Chepurko, I. Detecting Hidden Information in FAT. 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. 412-429. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

29. 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).

30. 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.

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

32. Смирнов А.А., Коваленко А.В. Комплекс математических моделей технологии тестирования WEB-приложений. Інформаційні технології: сучасний стан та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: ТОВ «ДІСА ПЛЮС», 2018. – 461 с.

33. Смирнов А.А., Коваленко А.В. Разработка метода управления рисками разработки программного обеспечения. Інформаційні технології: проблеми та перспективи: монографія / За загальною редакцією В.С. Пономаренка. – Х.: Видавець Рожко С.Г., 2017. – 447 с.

34. Смірнов О.А. Дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем / О.О. Кузнецов, О.А. Смірнов, Д.О. Даниленко // Інформаційна та економічна безпека: сучасний стан та тенденції розвитку : монографія за заг. ред. – Х.: ХІБС УБС НБУ – 2014 – С. 82-100.

35. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні системи та мережі / Д.О. Даниленко, О.А. Смірнов, Є.В. Мелешко // Системи озброєння і військова техніка. – Випуск 1(29) – Х.: ХУПС – 2012. – С. 92-100

36. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения. Часть 1. Корреляционный анализ сетевого трафика // А.А.Смирнов, Д.А. Даниленко, Е.В.Мелешко // Науково-технічний журнал «Інформаційно-керуючі системи на залізничному транспорті» – Випуск 4(95). – Х.: УкрДАЗТ – 2012. – С. 8-14.

37. Смирнов А.А. Методы обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник наукових праць "Системи обробки інформації". – Випуск 3(101) том 2. – Х.: ХУПС – 2012. – С. 152-155.

38. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, А.В. Коваленко // Системи управління, навігації та зв'язку. – Випуск 1 (21) том 2. – Київ: ДП «ЦНДІНУ». – 2012. – С. 183-186.

39. Смирнов А.А. Системы обнаружения и предотвращения вторжений для защиты компьютерных сетей от вредоносного программного обеспечения / Д.А. Даниленко, А.А. Смирнов, И.Г. Кирилов // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 21-22 березня 2012 р. – Харків. АВВ МВС. – 2012. – С. 70-71.

40. Смірнов О.А. Дослідження методів виявлення вторгнень в телекомунікаційні мережі для підвищення інформаційної безпеки // Д.О. Даниленко // Збірник тез науково-практичної конференції «Захист інформації в інформаційно-комунікаційних системах». м. Київ. 24-27 квітня 2012 р. – Київ: НАУ. – 2012. – С. 22-25.

41. Смирнов А.А. Исследование систем обнаружения и предотвращения вторжений для защиты телекоммуникационных сетей от вредоносного программного обеспечения / Д.А. Даниленко // Збірник тез доповідей VIII наукової конференції «Новітні технології – для захисту повітряного простору». Харків. 18-19 квітня 2012 р. – м. Харків. ХУПС. – 2012. – С. 45.

42. Смирнов А.А. Исследование методов сигнатурного обнаружения вредоносного программного обеспечения в телекоммуникационных системах и сетях // Д.А. Даниленко // Збірник тез XIII міжнародного науково-практичного семінару «Комбінаторні конфігурації та їх застосування». м. Кіровоград. 13-

14 квітня 2012 р. – Кіровоград: КНТУ. – 2012. – С. 43-45.

43. Смирнов А.А. Исследование методов проактивной защиты от вредоносного программного обеспечения в телекоммуникационных системах и сетях / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «інтегровані інтелектуальні робототехнічні комплекси» (ІПРТК-2012). м. Київ. 15-16 травня 2012 р. – Київ: НАУ. – 2012. – С. 314-315.

44. Смирнов А.А. Метод обнаружения вредоносного программного обеспечения на основе корреляционного анализа сетевого трафика / Д.А. Даниленко // Матеріали XII всеукраїнської наукової інтернет-конференції «Наукові дослідження: зв'язок теорії і практики». м. Тернопіль. 29-30 квітня 2012 р. – Тернопіль: ТНЕУ. – 2012. – С. 9-10.

45. Смирнов А.А. Метод детектирования вредоносного трафика в телекоммуникационных сетях на основе использования bds-тестирования / Д.А. Даниленко // Збірник тез V міжнародної науково-практичної конференції «Комп'ютерні системи та мережні технології» (CSNT-2012). м. Київ. 13-15 червня 2012 р. – Київ: НАУ. – 2012. – С. 121.

46. Смирнов А.А. Обнаружение и предотвращение вторжений в компьютерных сетях на основе статистического анализа сетевого трафика / А.А. Смирнов, Д.А. Даниленко // Збірник тез доповідей науково-практичної конференції «Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку». м. Харків. 12-13 березня 2014 р. – Харків. АВВ МВС. – 2014. – С. 13-14.

47. Смірнов О.А. дисперсійний аналіз мережного трафіку для забезпечення інформаційної безпеки телекомунікаційних систем та мереж / О.А. Смірнов, Д.О. Даниленко // Збірник тез V Всеукраїнської науково-практичної конференції "Інформатика та системні науки". м. Полтава. 13-15 березня 2014 р. – Полтава: ПУЕТ. – 2014. – С. 289-291.

48. Смирнов А.А. Метод дисперсионного анализа сетевого трафика для обнаружения и предотвращения вторжений в телекоммуникационных системах и

сетях/ А.А. Смирнов, Д.А. Даниленко // Збірник тез VI міжнародної науково-практичної конференції “Проблеми і перспективи розвитку ІТ-індустрії”. м. Харків. 17-18 квітня 2014 р. – Харків: ХНЕУ. – 2014. – С. 258.

49. Смірнов О.А. метод забезпечення інформаційної безпеки телекомунікаційних систем з використанням дисперсійного аналізу мережного трафіку / О.А. Смірнов, Д.О. Даниленко // Збірник тез міжнародної науково-практичної конференції «Інформаційна та економічна безпека» (INFECO-2014)». м. Харків. 15-16 травня 2014 р. – Харків: ХІБС УБС НБУ. – 2014. – С. 135-139.

50. Девянин П.Н. Модели безопасности компьютерных систем / П.Н. Девянин. – М.:Издательский центр «Академия», 2005. – 144 с.

51. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты / В.В. Домарев. – К.: ООО "ТИД "ДС", 2002 – 688 с.

52. ДСТУ ISO/IEC TR 13243-2003 Інформаційні технології. Посібник із методів та механізмів якості послуг / [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/informaciini-tehnologiyi.-posibnik-iz-metodiv-ta-mehanizmiv--nor2718.html

53. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.: УкрНДІССІ, 1995. – 23 с.

54. ДСТУ ISO 9000:2007 Системи управління якістю. Основні положення та словник термінів [Электронный ресурс]. – Режим доступа к ресурсу: http://document.ua/docs/tdoc14237.php

55. Ершов В.А.. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов. – М.: МГТУ им. Н.Э. Баумана, 2003. – 432 с.

56. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» [Электронный ресурс]. – Режим доступа к ресурсу: http://zakon4.rada.gov.ua/laws/show/2594-15

57. Информационная война и защита информации. Словарь основных терминов и определений [Электронный ресурс]. – Режим доступа к ресурсу:

http://www.csef.ru/files/csef/articles/2176/2176.pdf

58. Казарин О.В. Безопасность программного обеспечения компьютерных систем / О.В. Казарин. – М.:МГУЛ, 2003. – 212 с.

59. Касперский Е. Компьютерное зловредство / Е. Касперский. – СПб.: Питер, 2007. – 208 с.

60. Касперский К. Техника сетевых атак. [Електронний ресурс]. – Режим доступу до ресурсу: http://rghost.ru/download/43730077/

61. 8e48b6263ce45c7dc2a65a7453383dc33b22486d/Крис%20Касперски%20-%20Техника%20сетевых%20атак.pdf

62. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

63. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

64. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

65. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

66. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

67. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

68. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

69. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

70. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: *https://cpo.stu.cn.ua*

71. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування.....2
2 Підстава для розробки.....2
3 Мета та призначення розробки.....2
4 Джерела розробки.....2
5 Технічні вимоги.....2
5.1 Вміст проекту.....2
5.2 Показники призначення.....3
5.3 Вимоги до функціональних характеристик.....3
5.4 Вимоги до архітектури.....3
5.5 Вимоги до надійності.....3
5.6 Умови експлуатації.....4
5.7 Вимоги до складу та параметрів технічних засобів.....4
5.8 Вимоги до інформаційної і програмної сумісності.....4
5.8.1 Обладнання.....4
5.8.2 Мова програмування.....4
5.8.3 Вхідні дані.....5
5.8.4 Вихідні дані.....5
6 Вимоги до програмної документації.....5
7 Економічні вимоги.....5
8 Вимоги щодо охорони праці.....5
9 Перелік документів, що розробляються.....6
10 Етапи розробки.....6
11 Порядок контролю та приймання.....6

					ВКРМ-122.22.0025.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Глобенко В.В.			Дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп'ютерних систем	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КН-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи моніторингу LAN мереж інформаційних та комп’ютерних систем.

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. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи моніторингу LAN мереж інформаційних та комп’ютерних систем.

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 моніторингу LAN мереж інформаційних та комп'ютерних систем;

– 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 забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 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 Embarcadero RAD Studio Delphi 10.3.2.

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 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних 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 – 136 аркушів.

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 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 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

_____ Коваленко О.В.

Дослідження та програмна реалізація

системи моніторингу LAN мереж інформаційних та комп’ютерних систем

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2022 року

Основна програма

Файл Monitoring.dpr основної програми

```
program Monitoring;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName            : array[0..255] of WideChar;
     dwIndex            : DWORD;
     dwType             : DWORD;
     dwMtu              : DWORD;
     dwSpeed            : DWORD;
     dwPhysAddrLen      : DWORD;
     bPhysAddr          : array[0..7] of Byte;
     dwAdminStatus      : DWORD;
     dwOperStatus       : DWORD;
     dwLastChange       : DWORD;
     dwInOctets         : DWORD;
     dwInUcastPkts      : DWORD;
     dwInNUCastPkts     : DWORD;
     dwInDiscards       : DWORD;
     dwInErrors         : DWORD;
     dwInUnknownProtos  : DWORD;
     dwOutOctets        : DWORD;
     dwOutUCastPkts     : DWORD;
     dwOutNUCastPkts    : DWORD;
     dwOutDiscards      : DWORD;
     dwOutErrors        : DWORD;
     dwOutQLen          : DWORD;
     dwDescrLen         : DWORD;
     bDescr             : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
   PMibIfRow = ^TMibIfRow;
   PMibIfArray = ^TMibIfArray;

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

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

var
NetShareEnum:function ( pszServer   : PChar;
                        sLevel           : Cardinal;
                        pbBuffer    : Pchar;
                        cbBuffer    : Cardinal;
                        pcEntriesRead,
                        pcTotalAvail: Pointer):DWORD; stdcall;

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

var
NetShareAddNT: function(servername: PWideChar;
                        level: DWORD;
                        buf: Pointer;
                        parm_err: LPDWORD): DWORD; stdcall;

var
NetShareAdd: function ( pszServer:Pchar;
                        sLevel:Cardinal;
                        pbBuffer:PChar;
                        cbBuffer:Word):DWORD; stdcall;

Var
NetSessionEnumNT:function(servername,
                        UncClientName,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetSessionEnum:function(pszServer:PChar;
                        sLevel: DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvial:Pointer):integer; stdcall;

var
NetSessionDelNT:function(ServerName,
                        UncClientName,
                        username:PWChar):DWORD; stdcall;

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
```

```
                        sReserved: SmallInt):DWORD; stdcall;

var
NetFileEnumNT:function( servername,
                        basepath,
                        username:PWChar;
                        level:DWORD;
                        bufptr:Pointer;
                        prefmaxlen:DWORD;
                        entriesread,
                        totalentries,
                        resume_handle:LPDWORD):DWORD; stdcall;

var
NetFileEnum:function(     pszServer,
                        pszBasePath:PChar;
                        sLevel:DWORD;
                        pbBuffer:Pointer;
                        cbBuffer:DWORD;
                        pcEntriesRead,
                        pcTotalAvail:pointer):integer; stdcall;

var
NetFileClose:function(  ServerName:PWideChar;
                        FileId:DWORD):DWORD; stdcall;

var
NetFileClose2:function(pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

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

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT- підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
```

```
end;

////////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

////////////////////////////
//
//  Закриття загального ресурсу
//
```

```
procedure TMainForm.btnCloseSharesClick(Sender: TObject);
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);   //Видаляємо ресурс
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Показу діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
```

```
  end else Result := ' ' ;
  Result := String(TempPath);
end;

////////////////////////////
//
//  Додавання загального ресурсу
//

procedure TMainForm.btnAddSharesClick(Sender: TObject);
const
  STYPE_DISKTREE = 0;
  ACCESS_ALL =  258;
  SHI50F_FULL = 258;
var
  FLibHandle : THandle;
  Share9x : TShareInfo50;
  ShareNT : TShareInfo2;
  TmpDir, TmpName: String;
  TmpDirNT, TmpNameNT: PWChar;
  OS: Boolean;
  TmpLength: Integer;
begin
  TmpDir := SelectDirectory; //Визначаємо шлях до наступного ресурсу
  TmpName := InputBox(' Share name' ,' Enter name' ,' Test' ); //Визначаємо ім'
я під яким він буде видний у мережі
  if TmpDir = ' '  then Exit;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кіл-У максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-У тік підкл.

    GetMem(TmpDirNT, TmpLength);
    StringToWideChar(TmpDir, TmpDirNT, TmpLength);
    ShareNT.shi2_path := TmpDirNT; //Шлях до ресурсу

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

```
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати кіл-у секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
```

```
begin
  lvSessions.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
var
```

```
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
```

```
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
    begin
      FreeLibrary(FLibHandle);
```

```
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний- вихідний трафік
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := '  00-00-00'  else
    begin
      Result := ' ' ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+' -' ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо про всякий випадок таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(' IPHLPAPI.DLL' ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ' GetIfTable' );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;
```

```
 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

function EnumResources(const ParentNode: TTreeNode;
ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
 function ShowResource(const ParentNode: TTreeNode; Res: TNetResource):
TTreeNode;
  begin
   Result:=MainForm.NetTree.Items.AddChild(ParentNode,
string(Res.lpRemoteName));
  end;

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                              @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає о у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES данні на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES данні в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
    for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;

procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
```

```
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(‘ WNetCloseEnum Помилка’ );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=‘ Пошук мережних ресурсів. Чекайте...’ ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ‘ Network Resources’ ),
                              ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=‘ Обновити список ресурсів’ ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,’ open’ ,PChar(NetTree.Selected.Text),’ ‘ ,’ ‘ ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;
```

```
procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл IPHLPAPI.pas- обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

const
  ANY_SIZE      = 1;
  MAX_ADAPTER_DESCRIPTION_LENGTH = 128; // arb.
  MAX_ADAPTER_NAME_LENGTH = 256; // змінна
  MAX_ADAPTER_ADDRESS_LENGTH = 8; // змінна
  DEFAULT_MINIMUM_ENTITIES = 32; // змінна
  MAX_HOSTNAME_LEN = 128; // змінна
  MAX_DOMAIN_NAME_LEN = 128; // змінна
  MAX_SCOPE_ID_LEN = 256; // змінна

 // Вузлові типи ( NETBIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NETBIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , '
Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'
      );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

 found in ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , '
loopback' , ' SLIP'  );
   AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , '
tokenring' ,
            ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , ' ' ,
' ' , ' ' , ' PPP' ,
            ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

      //-------для інших MS заготовочних файлів------------------

        MAX_INTERFACE_NAME_LEN = 256; { mrapi.h }
        MAXLEN_PHYSADDR = 8; { iprtrmib.h }
        MAXLEN_IFDESCR = 256; {-"--     }

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

      type
        TMacAddress = array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;

      //---IP адресні структури-------------------------

        PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
        TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
        //
        PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
        TIP_ADDR_STRING = packed record // для використання у зв' язних списках
          Next: PTIP_ADDR_STRING;
          IpAddress: TIP_ADDRESS_STRING;
          IpMask: TIP_ADDRESS_STRING;
          Context: DWORD;
        end;

      //-----Fixed Info структура---------------------

        PTFixedInfo = ^TFixedInfo;
        TFixedInfo = packed record
          HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // данні
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // данні
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // данні
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати данні зразу. Стан >= DISCONNECTED
//
      // може передавати деякі данні. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                      або не з'єднується з картою.        //
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
```

```
      //                      не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                      з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                      в праці.                                    //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                        //
      //
//
      ////////////////////////

      const
      // данні додані до ipifcons.h
        IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        IF_OPER_STATUS_UNREACHABLE = 1 ;
        IF_OPER_STATUS_DISCONNECTED = 2 ;
        IF_OPER_STATUS_CONNECTING = 3 ;
        IF_OPER_STATUS_CONNECTED = 4 ;
        IF_OPER_STATUS_OPERATIONAL = 5 ;

        MIB_IF_TYPE_OTHER = 1 ;
        MIB_IF_TYPE_ETHERNET = 6 ;
        MIB_IF_TYPE_TOKENRING = 9 ;
        MIB_IF_TYPE_FDDI = 15 ;
        MIB_IF_TYPE_PPP = 23 ;
        MIB_IF_TYPE_LOOPBACK = 24 ;
        MIB_IF_TYPE_SLIP = 28 ;

        MIB_IF_ADMIN_STATUS_UP = 1 ;
        MIB_IF_ADMIN_STATUS_DOWN = 2 ;
        MIB_IF_ADMIN_STATUS_TESTING = 3 ;

        MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
        MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
        MIB_IF_OPER_STATUS_CONNECTING = 3 ;
        MIB_IF_OPER_STATUS_CONNECTED = 4 ;
        MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          dwLastChange: DWORD;
          dwInOctets: DWORD;
          dwInUcastPkts: DWORD;
          dwInNUCastPkts: DWORD;
          dwInDiscards: DWORD;
          dwInErrors: DWORD;
          dwInUnknownProtos: DWORD;
```

```
          dwOutOctets: DWORD;
          dwOutUCastPkts: DWORD;
          dwOutNUCastPkts: DWORD;
          dwOutDiscards: DWORD;
          dwOutErrors: DWORD;
          dwOutQLen: DWORD;
          dwDescrLen: DWORD;
          bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
        end;

       //
        PTMibIfTable = ^TMIBIfTable;
        TMibIfTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..ANY_SIZE- 1] of TMibIfRow;
        end;

      //---ADAPTER INFO структура-----------------------

        PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
        TIP_ADAPTER_INFO = packed record
          Next: PTIP_ADAPTER_INFO;
          ComboIndex: DWORD;
          AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       //
данні
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// данні
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // данні
          Index: DWORD;
          aType: UINT;
          DHCPEnabled: UINT;
          CurrentIPAddress: PTIP_ADDR_STRING;
          IPAddressList: TIP_ADDR_STRING;
          GatewayList: TIP_ADDR_STRING;
          DHCPServer: TIP_ADDR_STRING;
          HaveWINS: BOOL;
          PrimaryWINSServer: TIP_ADDR_STRING;
          SecondaryWINSServer: TIP_ADDR_STRING;
          LeaseObtained: LongInt ; // UNIX час, секунди з 1970
          LeaseExpires: LongInt;   // UNIX час, секунди з 1970
          SpareStuff: array [1..200] of char ;   // данні- простір для списку IP
адрес
        end;

      //--------TCP структура---------------------

        PTMibTCPRow = ^TMibTCPRow;
        TMibTCPRow = packed record
          dwState: DWORD;
          dwLocalAddr: DWORD;
          dwLocalPort: DWORD;
          dwRemoteAddr: DWORD;
          dwRemotePort: DWORD;
        end;
        //
        PTMibTCPTable = ^TMibTCPTable;
        TMibTCPTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..0] of TMibTCPRow;
        end;
        //
        PTMibTCPStats = ^TMibTCPStats;
        TMibTCPStats = packed record
          dwRTOAlgorithm: DWORD;
          dwRTOMin: DWORD;
          dwRTOMax: DWORD;
          dwMaxConn: DWORD;
          dwActiveOpens: DWORD;
```

```
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА -------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE- 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
  end;

//------IP СТРУКТУРА ------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
```

```
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

  //
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPForwardRow;
  end;

//---- ICMP-СТРУКТУРА ------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
```

```
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;

//---------імпорт до IPHLPAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі баги при використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
    Result := True;
    if IpHlpModule <> 0 then Exit;

// відкрити DLL
```

```
            IpHlpModule := LoadLibrary (IpHlpDLL);
            if IpHlpModule = 0 then
            begin
                Result := false;
                exit ;
            end ;
            GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
            GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' )
;
            GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
            GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
            GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
            GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
            GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
            GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
            GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
            GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
            GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
            GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
            GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
            GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
            GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
        end;

        initialization
            IpHlpModule := 0 ;
        finalization
            if IpHlpModule <> 0 then
            begin
                FreeLibrary (IpHlpModule) ;
                IpHlpModule := 0 ;
            end ;

        end.
```

Файл IPHelper.pas- функції роботи з IP-Протоколом

```
unit IPHelper;


interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

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


const
    // тільки найбільш популярні сервіси...
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Ping    }
    ( 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'  ),     { File Transfer Protocol- данні}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Часовий протокол }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { Сервіс – Кто це  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP Сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP Кієнт }
    ( 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'  ),     { Протокол SUN Remote Procedure Call }
    ( Prt: 119; Srv: ' NNTP   '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( 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   '  )
```

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

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_ПОМИЛКА' , ' 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'
  );


//-----Перетворення різних перерахованих величин у рядки---------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

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

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

  IPForwProtos  : array[1..18] of string =
    ( ' інший' , ' 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 ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі данні---------------------

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 );

// функції 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;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
```

```
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;

//-----------------------------------
{ перетворення числового MAC-адреса 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' ;
    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 крапкової десяткової IP-адреси 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 номеру порту в мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

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

//-----------------------------------
{ перетворення номеру порту в 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;

//-----------------------------------
{ голона частина, фіксація мережний параметрів }

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( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // данні
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // данні
begin
```

```
    InfoSize := 0 ;   // данні
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  // данні
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // данні
    GetMem (FixedInfo, InfoSize) ;                    // данні
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   // данні
    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 ;  // данні
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                    NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // данні
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                          Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                  // данні
    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;


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

// включення байтів у/з для кожного адаптера

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 ; // данні
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // данні
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
```

```
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      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( ' данних 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) ;  // конвертуємо Юнікод
до рядка
             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),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // данні, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна 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);  // очищаємо буфер з  W98 не беремо
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
```

```
end ;

//-----------------------------------
{ інформація про інстальовані 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 конвертуємо в 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) ;  // більше пам' яті
        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 ;
  //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( ' данних немає ' )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' |'  + Description ); // jpt : не
використовується
            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 : не використовується
            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 ;

//-----------------------------------
{ моні торимо час доступу 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; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
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 );   // данні
  //
  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 ) ); // бермо 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 ) );

  // необхідно відновити показник!
  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 довжину таблиці
```

```
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // данні
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо 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 останій 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 виходу         : '  + IntToStr( dwRTOMin ) + '
ms'  );
      List.Add( ' Максимальний час виходу         : '  + 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( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні 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 довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // виділяємо пам' ять, 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 останій запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
```

```
      end;
    end
    else
      List.Add( ' немає даних.'  );
  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 );  // данні
  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( ' немає даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик!
  dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації; }
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 ;   // данні
          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( ' немає даних.'  );
  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( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// данні
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації  : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована  : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      // данні
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( ' Датаграми (In)    : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів         : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
```

```
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//-------------------------------------*//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     // данні
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( ' Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                 : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений         : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит         : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь           : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка                 : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений         : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено           : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит         : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо відповідь           : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + 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 TCP_IP.pas- монітор TCP/IP з'єднань

```
unit TCP_IP;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
var
```

```
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Відбулася помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( ' Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл Stat.pas- статистика мережі

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл About.pas- довідка

```
unit About;

interface

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

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```