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
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Кіблик І.О.
«____» _____ 2024 р.

Керівник проекту
канд. фіз.-мат. наук, доцент
_____ Якименко Н.М.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Кіблику Івану Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric*

2. Керівник роботи *Якименко Наталія Миколаївна, канд. фіз.-мат. наук, доцент*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 132-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric*

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

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

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

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Якименко Н.М.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Кіблик І.О.
(прізвище та ініціали)

АНОТАЦІЯ

Кіблик І.О. Програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Метою розробки є програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Результат роботи – програмна реалізація системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

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 з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерна інженерія, ЦОД, Fabric

ABSTRACT

Kiblyk I.O. Software for a system of virtualized data centers using Fabric technological solutions. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the system of virtualized data centers using Fabric technological solutions.

The purpose of the development is software for a system of virtualized data centers using Fabric technological solutions.

The result of the work is the software implementation of the system of virtualized data centers using Fabric technological solutions.

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 a PC with Windows 10/11 OS.

The program was developed in the Delphi 10 environment.

Keywords: computer engineering, data center, Fabric

ЗМІСТ

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. Для реалізації хмарних обчислень центри обробки даних їхня мережна архітектура повинні відповідати певним вимогам. Упоратися з новими завданнями допоможе інноваційне рішення мережної архітектури центра обробки даних Fabric, що має наступні переваги:

– Гнучкість: висока масштабованість хмарних сервісів і канали для передачі великих даних (Big Data).

– Спрощення процесів: хмарне рішення для мережі в 10 разів прискорює активацію хмарних сервісів.

– Відкритість: спрощення хмарних обчислень за рахунок безшовного підключення до основних хмарних платформ.

Рішення Fabric наступного покоління дозволяє створити гнучку, зручну в керуванні й відкритій мережі для стабільного розвитку хмарних сервісів.

Підприємства й оператори всіх розмірів зосереджені на будівництві мереж центрів обробки даних з підтримкою стабільного розвитку хмарних сервісів.

У міру росту популярності хмарних сервісів і додатків SDN центри даних еволюціонують убік великих даних, нових послуг і розмаїтості систем. При цьому існує кілька серйозних проблем:

– Застосування великих даних пов'язане з інтенсивним збільшенням мережного трафіку.

– IT-системи повинні швидко реагувати на більше швидкі й часті відновлення послуг.

– Для хмарних екосистем потрібні більше відкриті мережі.

Хмарне рішення швидко допоможе вам адаптуватися до нових умов.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

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

– Огляд існуючих систем віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Дослідження системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Програмна реалізація системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Хмарне рішення дозволяє замовникам будувати гнучкі, прості, відкриті мережі центрів даних наступного покоління з підтримкою стабільного розвитку хмарних послуг.

У хмарних мережах використовуються високопродуктивні комутатори центра дані серії CloudEngine (CE). Дана серія на базі універсальної платформи маршрутизації (VRP) надає різні функції для послуг центрів даних. У той же час гнучкий контролер відповідає за керування й планування ресурсів ІКТ, дозволяючи замовникам швидко реалізовувати хмарні сервіси.

Хмарні мережі сумісні із хмарними платформами основних виробників. Вони підтримують широкий спектр хмарних послуг і додатків. Рішення прекрасно сумісне з Інтернетом. Воно підходить для використання в сфері фінансів і енергетики. До того ж, воно застосовується урядами, великими підприємствами й операторами. Продукти багаторівневої мережі спрощують будівництва мережі центра даних. Специфікації комутаторів опорної мережі серії CE12800 відповідають найвищим світовим стандартам. Високопродуктивні комутатори CE7800/CE6800/CE5800 використовуються для організації мереж з високим рівнем масштабованості.

Система передачі, маршрутизації, безпеки й мережного керування є комплексним рішенням мережі центра даних. Воно складається з декількох допоміжних рішень:

– З'єднання усередині центра даних (рішення хмарного зв'язку).

– З'єднання між центрами даних з відновленням після збоїв.

– Оптимізація захисту центра даних і його додатків.

– Керування мережею центра даних.

1.2 Область застосування

На зміну багато років домінувала в IT архітектурі « клієнт-сервер» приходить сервіс-орієнтована архітектура. Якщо раніше більшість процесів серверної частини виконувалися на одному фізичному сервері, а тому при взаємодії між ними мережа не задіялася, то зараз для підвищення ефективності використання обчислювальних ресурсів окремі процеси розподіляються по різних серверах, що істотно підвищує навантаження на мережу. У сервіс-орієнтованій архітектурі більшість даних залишається усередині ЦОД і передається між його встаткуванням (напрямок «захід – схід»), тоді як частка трафіку між клієнтами й серверами («північ – південь») у загальному обсязі даних, що пересилаються по мережах, знижується. На першій стадії розвитку технологій віртуалізації вона використовувалася для підвищення ефективності використання ресурсів окремих серверів, що лише незначно збільшувало навантаження на мережу. З появою рішень на зразок VMotion стала можливої міграція віртуальних машин (без переривання роботи додатків) для підвищення ефективності використання ресурсів серверного парку в цілому. Це привело до різкого росту трафіку «захід – схід». Один тільки факт «розвороту на 90°» основного напрямку передачі трафіку вже робить малоефективними як традиційну ієрархічну архітектуру мереж (доступ – агрегація – ядро), так і логічні структури на зразок «дерева», які були оптимізовані для пересилання трафіку від «кореня» до «листя» і назад, тобто по вертикалі «північ – південь». До цьому варто додати ріст інтересу замовників до конвергенції мереж – до впровадження технології FCoE, для якої потрібна гарантована передача трафіку без втрат, а також до повноцінного віртуалізації мережної інфраструктури для підтримки вже віртуалізованих серверів і переходу до хмарної моделі надання/одержання IT-сервісів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Dell Active Fabric

Dell Active Fabric – сучасна мережна архітектура. Вона побудована на основі новітніх технологій, можливостях гнучкого масштабування в міру росту.

Dell Active Fabric надає швидку, однорангову мережну архітектуру з повним взаємозв'язком і резервними шляхами, гнучку й більше підходящу для зростаючого обсягу трафіку між вузлами одного рівня (схід – захід) у віртуалізованих ЦОД-ах і часток хмарах. Рішення Active Fabric роблять плоскої традиційну архітектуру мережі ЦОД-а, використовуючи 10/40 Гбіт/с комутатори з фіксованим формфактором з високою щільністю компонування й малою затримкою, які можна швидко й просто впровадити, забезпечуючи при цьому нарощування мережі до гіпермасштабів.

Рішення Dell Active Fabric забезпечує економію витрат, у середньому, 59% і споживає на 77% менше потужності в порівнянні із традиційним мережним устаткуванням у шасі.

Інтеграція:

– Рішення забезпечує підтримку технологій NVO, використовуючи провідні гипервизори Microsoft,VMware і OpenStack.

– Рішення забезпечує підтримку контролерів на базі OpenFlow від ведучих вендорів, включаючи Big Switch Networks.

– Рішення підтримує успадковані інтерфейси програмування, включаючи Telnet/CLI, TCL, REST, SNMP, скрипти Perl і Python.

Переваги:

– Рішення розроблене спеціально для віртуалізованого, конвергентного й SDN-середовища;

– Високопродуктивна 10 Гбіт/с і 40 Гбіт/с розподілена комутаційна мережа рівнів L2/L3 із множинними шляхами;

– Конвергенція LAN/SAN з використанням протоколів Data Center Bridging: iSCSI, Fibre Channel(FC) і Fibre Channel over Ethernet (FCoE);

– Підтримка OpenFlow;

– Стандартні інтерфейси до вищестоящих і нижчестоящих систем.

Dell Active Fabric Manager – саме передове ПЗ автоматизації комутаційної мережі. Dell Active Fabric Manager (AFM) – перше, у своєму роді, програмний засіб, що автоматизує завдання, пов'язані із плануванням, проектуванням, нарощуванням і моніторингом розподіленої комутаційної мережі. AFM здатно скоротити строки впровадження на величину до 86% у порівнянні із процесом, виконуваним вручну.

Функціонал:

– Майстер розробки. Значно спрощує процес проектування мережі, пропонуючи інтуїтивно зрозумілий графічний інтерфейс, що виконує всі необхідні обчислення, щоб представити оптимальний проект розподіленої комутаційної мережі, і який усуває помилки.

– Автоматизоване планування, перевірка й конфігурування. Забезпечує покроковий підхід на шляху від проектування розподіленої комутаційної мережі до її повнофункціонального впровадження, усуваючи необхідність використання інтерфейсу командного рядка – процесу, що віднімає багато часу й сполучене з помилками.

– Простота інтеграції, рольовий доступ – AFM абстрагує розподілену комутаційну мережу як єдине ціле, а не на рівні пристроїв, дозволяючи іншим програмним засобам у ЦОД-і легко інтегруватися. У сполученні із цим, доступ на основі ролей дозволяє різним підрозділам (напр., адміністраторам серверів і СЗД)

здійснювати моніторинг різних аспектів функціонування комутаційної мережі, не порушуючи її роботу.

Dell Networking S5000 – компактний модульний конвергентний комутатор. Dell Networking S5000 – це перший модульний 10/40 Гбіт/с комутатор LAN/SAN компанії Dell розміром 1RU з убудованими інтерфейсами FC і FCoE, розташовуваний угорі стійки (Top-of-Rack). Це новаторське системне рішення реалізоване на основі апробованої в галузі, функціональної операційної системи, що забезпечує максимальну надійність і безвідмовність. Платформа Dell Open Automation забезпечує інтегровану автоматизацію, скриптинг і адміністрування з можливістю програмування, розширюючи гнучкість мережі у віртуалізованому середовищу.

Відмінні риси:

– Модульність, масштабування в міру росту. Забезпечує більшу гнучкість впровадження й розподілу IT-бюджету в порівнянні з комутаторами з фіксованим набором портів. Шасі S5000 розраховано на чотири модулі, дозволяючи замовникам установити один модуль і потім додавати нові в міру необхідності, а не купувати всі чотири відразу.

– Конвергенція LAN/SAN з високою щільністю компонування. Дозволяючи заощадити на кількості комутаторів і необхідному просторі в стійці, S5000 має щільність портів на відсік в 1,3-2,6 рази вище, ніж аналогічне встаткування в галузі. S5000 включає максимум 64 порту 10 Гбіт/с Ethernet, або 48 портів Ethernet/FC і 16 портів 10-Гбіт/с Ethernet.

– Мережі зберігання з великою функціональністю. Повна підтримка розподіленої комутації по iSCSI, RoCE, NAS, FCoE і FC, усе на одній і тій же платформі.

– Готовність до майбутнього, максимальний захист інвестицій. Завдяки модульності й системному підходу апаратна платформа S5000 готова до майбутнього й забезпечить підтримку нової функціональності й опцій, які будуть запропоновані, без втрати існуючих інвестицій в інфраструктуру.

Простота інтеграції, перевірена функціональна сумісність із продукцією ведучих вендорів мережних адаптерів, комутаторів і СЗД, включаючи Broadcom, Brocade, Emulex, Intel иQlogic.

Рішення 10Gb і 40Gb Active Fabric поставляються корпорацією Dell і її партнерами за допомогою програми PartnerDirect.

CS Fabric Brocade, Software Defined Networking, Комутаційна фабрика, ЦОД

Компанія Brocade оголосила про поліпшення в портфоліо Brocade VCS Fabric. Інновації спрямовані на підвищення гнучкості й динамічності IT-середовища, а також повинні допомогти клієнтам перейти на технології New IP. Інновації в портфелі Brocade VCS Fabric включають комутатор фіксованої конфігурації Brocade VDX 6940, розроблений для динамічних, масштабованих мережних архітектур ЦОД і який володіє високою щільністю портів. Технологія Zero-Touch Provisioning дозволяє настроїти пристрій і ввести його в експлуатацію протягом мінути. Як повідомляється, на відміну від традиційних мереж, де кожний комутатор управляється окремо, Brocade VDX 6940 дозволяє клієнтам масштабувати мережну інфраструктуру, не створюючи додаткових операційних складностей. Використовуючи можливості Brocade VCS Logical Chassis, замовники можуть управляти 48-ю комутаторами як одним логічним пристроєм.

Автоматизований підхід до впровадження й використання комутатора Brocade VDX перевершує нативні можливості рішення VCS. Клієнти зможуть інтегрувати мережну інфраструктуру Brocade VDX з існуючої екосистемою центра обробки даних завдяки підтримці віртуалізації, засобів керування процесами в хмарах (cloud orchestration), технологій SDN, набору інструментів для програмування й інтеграції з DevOps. У результаті клієнти зможуть реалізувати цілісний підхід до керування ЦОД, прискорити впровадження новацій і гарантувати готовність мережних ресурсів при запуску нових сервісів.

Розроблений спеціально для архітектур з горизонтальним масштабуванням, новий комутатор фіксованої конфігурації Brocade VDX 6940

може використовуватися як основний (spine) або відгалужений (leaf). Крім швидкого введення в експлуатацію й простоти використання, новий комутатор забезпечує ідеальний баланс між буферізацією і затримкою, що відповідає вимогам сучасних ЦОД, де підтримка непередбачених профілів трафіку повинна сполучатися з високою продуктивністю додатків. Його неблокуєма архітектура забезпечує комутацію на швидкості каналу для пакетів будь-яких розмірів. Внутрипроцесорний пристрій з динамічної буферізацією забезпечує високу швидкість і мінімальну затримку, критично важливу для середовищ типу Hadoop.

Комутатор Brocade VDX 6940 доступний у двох моделях. Модель Brocade VDX 6940-36Q оснащена 36-ю портами 40Gb QSFP+. За допомогою розвідного кабелю кожний з портів 40Gb можна розбити на чотири порти по 10Gb, тим самим одержавши 14410 Gb-роз'ємів. Це більш ніж на 40% перевершує ємність будь-якого іншого мережного комутатора того ж класу. Модель Brocade VDX 6940-144S у форм-факторі 2U має до 96 портів 10Gb, а також 12 портів 40Gb або 4 порти 100Gb.

Відкрита програмна інтеграція існуючих технологічних екосистем з комутаторами Brocade VDX:

– Підтримка SDN: Для полегшення переходу клієнтів до SDN у січні компанія Brocade оголосила про доступність безкоштовної річної ліцензії на новий контролер Brocade Vyatta (на основі OpenDaylight). У новій версії операційної системи Brocade Network OS (NOS) 6.0, що буде випущена найближчим часом, комутатори Brocade VDX будуть інтегруватися з контролером Brocade Vyatta, а також з контролерами на основі OpenDaylight від сторонніх виробників за допомогою протоколу OpenFlow 1.3. Це також буде сприяти легкому й плавному переходу замовників до SDN рішень.

– Технології керування хмарами: При переході організацій до хмарних архітектур ключовим моментом для використання рішення IaaS (Інфраструктура як Сервіс), стає автоматичне надання дискового простору, обчислювальних і мережних елементів. Для задоволення цієї потреби компанія Brocade в 2012 році

впровадила підтримку OpenStack у сімейство комутаторів Brocade VDX і з тих пор активно підтримує діяльність співтовариства розроблювачів програмного забезпечення з відкритим кодом. У квітні компанія Brocade додасть функціонала VCS Layer 3 у реліз OpenStack Kilo. Нові можливості будуть містити в собі підтримку Inter-VLAN маршрутизації для забезпечення взаємодія різних груп користувачів і списки контролю доступу для запуску рішення FWaaS (Міжмережний екран як сервіс).

– Технології DevOps: Сфера застосування інструментів DevOps розширюється за межі серверів і То комутаторів. Випливаючи цієї тенденції, у версії NOS 6.0 компанія Brocade планує додати підтримку скриптів Puppet і Python для сімейства комутаторів Brocade VDX.

– Розширена інтеграція з VMware: Компанія Brocade продовжує розвивати своє стратегічне партнерство з VMware для поглиблення інтеграції віртуальних серверів і віртуальних мереж на основі NSX з фізичною мережною інфраструктурою. Уже сьогодні користувачам VMware Solutions Exchange доступні рішення Brocade IP Analytics і Content Pack для VMware vRealize Operations, які спрощують керування фізичними й віртуальними ресурсами. Такий підхід підвищує рівень автоматизації і якість що збирається аналітики, що у свою чергу допомагає підтримувати прийнятні рівні обслуговування й зберегти розумний рівень витрат.

Крім того, компанія Brocade поліпшила VCS Gateway для NSX таким чином, що вся фабрика VCS може функціонувати як VXLAN-шлюз. Це дозволить поліпшити стійкість і продуктивність шлюзу, а також значно полегшити керування, оскільки безліч комутаторів, що виконують функцію NSX-шлюзу, може управлятися як єдиний пристрій. Цей функціонал буде доступний у версії NOS 6.0.

Уніфікована фабрика Cisco (Unified Fabric Cisco)

Cisco оголосила два важливих новаторських доповнення до портфеля рішень для уніфікованої фабрики (Unified Fabric). Одне з них – Dynamic Fabric

Automation (DFA) – автоматизує виділення мережних ресурсів, спрощує керування такою фабрикою (Fabric Management) і оптимізує її для підвищення ефективності й масштабованості. Крім того, розширене сімейство Nexus 7000. У ньому з'явилися нові комутатори Nexus 7700 і нові модулі уведення/виводу F3, що підтримують кращу в галузі масштабованість 40G/100G і самий повний набір функцій комутації для ЦОД.

До основних властивостей нової архітектури Cisco для центрів обробки даних відносяться:

– Висока швидкість установки додатків (будь-яке робоче навантаження в будь-який час). Повністю автоматизована програмувальна мережна інфраструктура скорочує строки впровадження додатків.

– Загальна відкрита платформа для фізичної, віртуальної й хмарної інфраструктури. Нова архітектура забезпечить повну інтеграцію між фізичними й віртуальними додатками й нормалізує доступ до них з кінцевих пристроїв. При цьому вона підтримає гнучкість програмного забезпечення, високу продуктивність, масштабованість і прозорість апаратних пристроїв при роботі в будь-якому середовищі: мультивендорскому, віртуалізованому, сугубо апаратному, розподіленому, масштабованому й хмарному.

– Системний характер архітектури – означає цілісний спрощений підхід, інтеграцію інфраструктури, послуг і функцій інформаційної безпеки, одержання телеметричних даних у реальному часі й можливість розширення для підтримки майбутніх послуг.

– Єдина політика, керування й експлуатація. Єдина структура для керування політиками і єдиною операційною моделлю підтримають автоматизацію в інтересах всіх співробітників, відповідальних за мережу, інформаційну безпеку й додатки. У майбутньому ця структура поширюється на обчислювальні ресурси й системи зберігання даних.

– Відкриті інтерфейси API, програмне забезпечення з відкритим кодом і робота в мультивендорскому середовищу. Підтримка великої екосистеми

партнерів, що використовують широкий спектр опублікованих відкритих інтерфейсів API.

– Використання кращих замовлених і загальнодоступних процесорних рішень. Збалансований підхід прискорює інновації й поширення нових рішень у замовників і підтримує перехід до інфраструктури, орієнтованої на додатки й готової до майбутнього. Цей підхід оптимізує ціни, продуктивність, щільність, інформаційну безпеку й енергоспоживання. За допомогою інноваційних оптичних рішень він також захищає інвестиції, вкладені в існуючі мідні кабельні інфраструктури. Даний підхід допоможе замовникам оптимізувати капітальні й поточні витрати в умовах нинішнього переходу до технології 40G і майбутнього переходу до 100G.

Cisco обновила сімейство комутаторів Nexus і модернізувала уніфіковану фабрику (Unified Fabric), що підвищує масштабованість, гнучкість і керованість мереж. Нововведення в сімействі Nexus включають рішення для спрощеного виділення ресурсів, удосконалені функції керування й нові моделі комутаторів.

– Нові рішення для автоматизації динамічної фабрики (Cisco Dynamic Fabric Automation, DFA)

– Оптимізована інфраструктура, що підвищує ефективність і масштабованість. Оптимізована топологія spine-leaf з розширеною передачею даних, розподіленим рівнем керування (control plane) і інтеграцією фізичного й віртуального рівнів дозволяє вільно переміщати фізичні й віртуальні машини в будь-якому місці будь-якої мережі, підвищуючи можливості розширення мереж. Крім того, вона підвищує надійність мереж за рахунок мінімізації збоїв і підвищує масштабованість мультиарендного середовища, підтримуючи більше 10 тисяч мереж або орендарів.

– Спрощене керування фабрикою за допомогою відкритих інтерфейсів API для полегшення завдань експлуатації. Cisco Prime DCNM 7.0 забезпечує централізоване керування фабрикою й підтримує автоматизоване виділення мережних ресурсів, єдину точку доступу до фабрики, а також повну

прозорість хост-систем, мереж і орендарів. Відкриті інтерфейси API створюють більше тісну інтеграцію із засобами координації й автоматизації ресурсів, а також із хмарними платформами.

– Рішення Cisco Prime Data Center Network Manager (DCNM) 7.0. Єдина точка керування, що дозволяє автоматизувати й спростити розгортання інфраструктури, підтримує виділення ресурсів у динамічному режимі для віртуальних машин і надає кошти для пошуку й усунення несправностей.

– Рішення Cisco Prime Network Services Controller 3.6. Це рішення в динамічному режимі створює мережні сервіси, здійснює зв'язок із системами VMware і Cisco Nexus 1000V і передає необхідну інформацію рішенню DCNM.

– Автоматизоване виділення ресурсів для підвищення гнучкості забезпечує автоматизацію виділення ресурсів, спрощуючи підключення фізичних серверів і віртуальних машин і їхнє переміщення у фабриці. Автоматично створюють мережні політики на підставі мережних профілів і встановлює ці політики в мережі в момент, коли системний адміністратор виділяє ресурси для віртуальних і фізичних машин. При переміщенні віртуальної машини мережна політика автоматично застосовується на комутаторі, пов'язаному із цією машиною.

Перераховане вище дає замовникам істотні переваги в порівнянні із чисто програмним оверлейним підходом і в порівнянні з підходом, орієнтованим тільки на фізичні мережі.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies.

Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

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

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі.

Розробити функціональну та структурну схеми системи;

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Нові рішення, які в англомовній літературі одержали найменування Fabric, на думку більшості експертів, дозволять зробити мережі ефективною транспортною основою для віртуалізованих ЦОД. У галузі вже сформувалося загальне розуміння того, які повинні бути основні характеристики «фабрик»:

– підтримка всіх ліній зв'язку в активному стані (без властивому протоколу STP блокувань);

– використання найкоротших шляхів пересилання;

– забезпечення низької затримки і її варіації;

– повна відказостійкість при відсутності точок загальносистемної відмови.

Але варіанти реалізації «фабрик» сильно відрізняються в різних виробників. Деякі компанії вважають, що всі комутатори треба об'єднати в один динамічний комутатор, що для користувача буде виглядати як якийсь «чорний ящик». Він забезпечить відновлення фабрики, балансування навантаження й виконання інших функцій за допомогою «зашитих» у нього фірмових механізмів. Однак такий підхід вимагає використання встаткування одного виробника.

Ключовим елементом фабрики повинен стати контролер, керуючий всіма її елементами. По суті, мова йде про технологію централізовано програмувальних мереж (SDN) і використанні протоколу OpenFlow. Для підтримки ж в активному стані всіх наявних ліній зв'язку Extreme Networks пропонує використовувати протокол Multi-Switch Link Aggregation Groups (MLAG). З його розгляду ми й почнемо аналіз можливих варіантів побудови «фабрик».

Варіант 1. MLAG

Це найбільш проста й добре пророблена технологія для рішення завдання одночасного використання декількох шляхів передачі трафіку (нагадаю, що

протокол STP, щоб уникнути зациклення трафіку, «заморожує» частина канальних ресурсів, залишаючи активним тільки один шлях між двома вузлами). Вона заснована на стандартній функції агрегації каналів (Link Aggregation Group, стандарт IEEE 802.1ad), що дозволяє кілька фізичних каналів поєднувати в один логічний, підвищуючи загальну пропускну здатність мережі. MLAG дає можливість розподілити ці логічні канали між двома різними комутаторами.

Extreme Networks – далеко не єдиний виробник, що використовує MLAG у своїх рішеннях, але деталі реалізації цієї технології, а часто й назви, у кожної компанії свої. Наприклад, подібна технологія в рішеннях Avaya називається Split Multi-Link Trunking (SMLT), а в комутаторах Cisco Nexus 5000 і 7000 – Virtual PortChannel (vPC). Можливості підтримки декількох шляхів реалізовані також у комутаторах компанії Arista Networks, що, як і Extreme Networks, використовує термін MLAG.

Несумісність різних реалізацій унеможливлює використання в парі (кластері) комутаторів різних виробників. Але для зовнішніх пристроїв процес роботи MLAG прозорий: підключений комутатор доступу (або сервер) «не знає», що він з'єднаний із двома різними системами, тому, наприклад, пристрій Cisco цілком можна підключити до групи MLAG комутаторів Extreme або Arista.

Істотним обмеженням технології MLAG є те, що вона дозволяє об'єднати в групу тільки пари комутаторів. При цьому кожний комутатор у групі залишається фізично окремим пристроєм, а виходить, і управляти ним доводиться окремо (централізоване керування, звичайно, може бути реалізовано іншими засобами, наприклад за допомогою контролера SDN, але в самій технології MLAG ця можливість не передбачена). Таким чином, будучи ефективним інструментом для організації декількох активних шляхів передачі трафіку, MLAG реалізує тільки частину функціоналу справжньої «фабрики».

Варіант 2. Віртуальні шасі

Питання єдиного керування групою комутаторів надзвичайно актуальний у складних мережах, і саме потреба в його рішенні багато в чому підстьобує

інтерес до SDN. Таке керування вже досить давно реалізоване для розробок, які можна об'єднати загальним поняттям «віртуальне шасі». Мова йде про об'єднання декількох фізичних комутаторів у логічно єдиний пристрій із загальною системою керування.

Багато рішень класу «віртуальне шасі» ведуть своє походження від звичайних стекових комутаторів. У процесі розвитку цих продуктів короткі шини, що обмежують можливість просторового рознесення комутаторів стека, замінялися високошвидкісними лініями зв'язку (мідними або оптичними), які дозволяли розміщати пристрою в різних стійках ЦОД, – такі рішення іноді називають горизонтальними стеками.

«Віртуальні шасі» пропонують багато виробників. Одні з найвідоміших рішень – це Virtual Switching System (VSS) компанії Cisco і Intelligent Resilient Framework (IRF) компанії HP (отримано нею в результаті покупки 3Com). Відповідно до заяв цих виробників, при використанні 10-гігабітних ліній зв'язку комутатори в рамках одного «віртуального шасі» можуть бути рознесені не тільки по різних стійках одного ЦОД, але й на відстані в десятки кілометрів, що дозволяє побудувати єдину мережу для територіально розподіленого ЦОД.

У більшості рішень класу «віртуальне шасі» передбачається, що один з комутаторів групи одержує статус головного (майстер). Це порушує питання про те, як «віртуальне шасі» поведеться у випадку порушення зв'язку між його компонентами й що буде з тими пристроями, які виявляться відрізаними від майстра. Результат може бути різним – аж до втрати їхньої працездатності. Звичайно в «відрізаному» сегменті швидко вибирається свій майстер, але це може привести до іншої проблеми – конфлікту з основним майстром при відновленні зв'язку. Кожний виробник пропонує докладні рекомендації із грамотного проектування, налаштуванню й обслуговуванню таких систем, але використання нестандартних (фірмових) алгоритмів практично виключає можливість дати які-небудь загальні ради на цей рахунок.

Варто помітити, що більшість віртуальних шасі надають всі переваги підтримки множинних шляхів передачі для зовнішніх пристроїв. Інакше кажучи, такі шасі можуть пропонувати зовні ті ж канали MLAG, але при цьому усередині (між компонентами) використовувати різні фірмові технології. Тому варіанти 1 і 2 тісно зв'язані між собою.

Варіант 3. Розподілений комутатор

У компанії Juniper Networks також є рішення «віртуальне шасі» (для комутаторів Juniper EX), але вона вирішила піти далі й розробила принципово інше рішення Qfabric. Як розповідає Михайло Пергамент, при розробці QFabric було поставлено кілька ключових завдань.

Одна з них – забезпечити високий рівень масштабування (від сотень до тисяч 10-гігабітних портів) при незмінності ключових характеристик. Інакше кажучи, при збільшенні числа портів в «фабриці» ні затримка, ні рівень перепідписки не повинні зростати. (Помічу, що при традиційних методах розширення мережі ці завдання вирішити надзвичайно складно, оскільки підключення кожного нового комутатора збільшує число транзитних вузлів, а тому – і затримку.) Крім того, вся підсистема комутації з погляду керування повинна була виглядати як один пристрій і забезпечувати передачу будь-якого трафіку (L2, L3, FCoS, iSCSI, NAS та ін.).

Відправною точкою для розробки QFabric послужила архітектура звичайного модульного комутатора, що відповідає більшості перерахованих вимог за винятком одного, але дуже важливого: можливості масштабування обмежені розмірами шасі (числом слотів) такого комутатора. Традиційні способи нарощування ємності припускають установку ще одного пристрою, але при цьому очевидно підвищується складність і погіршується керованість системи. В Juniper вирішили кардинально змінити модель масштабування.

По суті, QFabric – це розподілений комутатор, у якому замість пасивної шини, що звичайно з'єднує лінійні карти й плати матриці комутації, використовуються оптичні канали, що зв'язують пристрої QF/Node (виконують

функції лінійних карт) і QF/Interconnect (застосовуються замість традиційних матриць комутації). У такий спосіб усунуте обмеження, що накладається розміром фізичного шасі, при збереженні більшості переваг єдиного пристрою. Керування фабрикою QFabric реалізовано «у стилі» SDN, тобто за допомогою зовнішнього контролера, що у рішенні Juniper називається QF/Director. Службовий трафік передається по виділеній мережі керування (out-of-band). Точніше, таких мереж дві, як і пристроїв QF/Director – для резервування.

Пристрої QF/Interconnect теж завжди встановлюються парами або навіть четвірками. Розгортання «фабрики» від Juniper можна починати з установки як пристрої QF/Interconnect двох невеликих продуктів QFX 3600-I, до яких каналами 40G підключається до 16 вузлів QF/Node (усього 384 порту 10G). У міру розширення ЦОД можна додати ще два пристрої QFX 3600-I, тоді число підтримуваних портів збільшується у два рази – до 768. «Фабрика», у якій функціонал QF/Interconnect реалізується за допомогою пристроїв QFX 3600-I, має суфікс M (Micro).

Наступна стадія масштабування – заміна QFX 3600-I на могутніше встаткування міжз'єднання: модульні пристрої QFX3008 з вісьма слотами вміщають до 128 портів 40G (QSFP+). При використанні чотирьох таких пристроїв загальна ємність фабрики перевищує 6000 портів 10G. Помітимо, що при модернізації пристрою QFX 3600-I викидати не треба: їх можна використовувати в якості QF/Node, для чого, щоправда, кожний порт 40G буде потрібно конвертувати в чотири порти 10G за допомогою спеціального розгалуджувача. Поки максимальна відстань між пристроями QF/Node і QF/ Interconnect становить 100 і 150 м при використанні оптики OM3 і OM4 відповідно.

Цікаво відзначити, що усередині QFabric використовуються протоколи IS-IS і BGP, які відносяться до рівня L3. Наприклад, коли новий вузол QF/Node підключається до мережі, саме протокол IS-IS служить для його автоматичного виявлення, після чого QF/Director автоматично генерує конфігураційні параметри

(IP-адреса й ін.) і направляє їхньому новому вузлу. Протокол BGP потрібний для розподілу динамічної інформації, наприклад нових MAC-адрес.

Інший приклад «розтягування» комутатора – рішення Cisco Fabric Extender (FEX). Його основу становлять «материнські» комутатори (серій Cisco Nexus 5000, Nexus 7000 або UCS Fabric Interconnect), до яких підключаються виноси FEX, що виконують функції віддаленої лінійної плати. Як виноси можуть використовуватися комутатори Nexus 2000, мережні модулі UCS 2100 Fabric Extender для блейд-серверів Cisco UCS, а також рішення Cisco Nexus B22 Fabric Extender для блейдсерверів HP. Унікальна особливість цієї «фабрики» полягає в тому, що вона може охоплювати інтерфейси сервера (за допомогою Cisco Adapter FEX) і навіть віртуальні машини (технологія VM-FEX). Це означає проникнення мережі «усередину» сервера, при цьому в єдиній площині комутації можуть перебувати не тільки порти комутаторів, але й серверні адаптери й віртуальні машини.

Хоча FEX і використовує достандартну реалізацію IEEE 802.1BR, поки це закрите рішення, а виходить, його вибір означає, що ваша мережа буде прив'язана до одному вендору. Власне кажучи, інші розглянуті вище рішення також не можна назвати стандартними. Є чи взагалі шанс побудувати «фабрику» на основі стандартів? Таку надію дають технології Transparent Interconnection of Lots of Links (TRILL) і Shortest Path Bridging (SPB), до розгляду яких ми й переходимо.

Варіант 4. TRILL

Технологія TRILL визначена в серії документів організації IETF (RFC 5556, 6325, 6327, 6349), але деякі механізми перебувають тільки в стадії розгляду. Часто неї називають маршрутизацією на рівні L2. Як відомо, класична маршрутизація виконується на підставі інформації рівня L3, при цьому рішення про вибір маршруту здійснюється за результатами обчислення найкоротшого шляху. TRILL реалізує схожу логіку, але тільки не для IP-, а для MAC-адрес. Не дивно, що « мовою» TRILL підтримуючу цю технологію комутатори називаються маршрутизируючими мостами, або RBridge.

Для обчислення найкращого шляху до пункту призначення комутатори RBridge використовують протокол IS-IS, заснований на відомому алгоритмі Shortest Path First (SPF). Комутатор, що перебуває на вході в хмару TRILL, за допомогою IS-IS відразу визначає 16-розрядний ідентифікатор комутатора на виході. Кожний наступний комутатор (транзитний вузол) у хмарі пересилає трафік на основі цього ідентифікатора, завдяки чому усередині хмари не потрібно підтримувати таблицю зовнішніх MAC-адрес. Вузли оперують дуже невеликим обсягом адресної інформації, що спрощує їхнє завдання, зокрема, по розподілі трафіку по безлічі шляхів. У технології TRILL уводиться такий важливий параметр, як «час життя» – Time To Live (TTL): при проходженні кадром кожного вузла в мережі TRILL значення цього параметра зменшується. Цей механізм відсутній у класичній технології Ethernet, що багато в чому і є причиною зациклення трафіку – без поля TTL кадр Ethernet може нескінченно довго «подорожувати» по мережі, якщо не досягне адресата.

У цей час кілька виробників при описі своїх рішень згадують про технологію TRILL. Зокрема, Cisco називає свою технологію FabricPath, підтримувану пристроями серій Nexus 5000 і 7000, сумісної з TRILL. Однак незалежні експерти відзначають ряд відступів від стандарту – зокрема, інший формат кадру, що використовується для передачі трафіку між комутаторами. Але оскільки Cisco бере активну участь у триваючій стандартизації TRILL, висока ймовірність, що фірмові функції згодом стануть частиною стандартів. Властиво, таке вже багаторазово відбувалося при формуванні стандартів на інші мережні технології. Sxxx – це номера (ідентифікатори) комутаторів, використовувані для доставки кадру усередині мережі FabricPath. Так, у кадр, відправлений вузлом з MAC-адресою A вузлу з MAC-адресою C, на вході в мережу FabricPath додається заголовок, де як номер вихідного комутатора вказується S300, і подальша передача до виходу з мережі FabricPath буде здійснюватися на підставі цього номера.

S100 і S300 – це так звані емульовані комутатори (або домени vPC+), тобто пари комутаторів, підключення до яких здійснюється за технологією Multi-Chassis PortChannel. Кожний з комутаторів у парі має також свій власний ідентифікатор FabricPath (скажемо, S101 і S102, S301 і S302), а емульований комутатор з погляду логічної топології FabricPath (як вона видна в IS-IS) виглядає як той, що знаходиться «за ними»: S100 досяжний через S101 і S102, а S300 – через S301 і S302. Тим самим пристрою, підключені до правого за схемою парі через канали vPC+ (безпосередньо або, як на схемі, через FEX), виявляються досяжні з «рівною вартістю» через S301 і S302, що забезпечує розподіл навантаження між всіма оптимальними каналами.

Зазначені два варіанти підключення виносів FEX – наскрізне (Straight-Through) і EvPC. Оскільки винос FEX – це «продовження» головного пристрою, найбільш простий варіант – підключення FEX тільки до одному «материнському» комутатору. У цьому випадку все налаштування й комутація здійснюється на одному комутаторі, а для резервування на випадок його відмови від серверів організуються канали vPC до виносів FEX, підключених до різних комутаторів у парі vPC. Такий варіант історично називається Straight-Through.

Альтернативний, більше складний варіант – підключення FEX відразу до двох головних комутаторів. У цьому випадку комутація відбувається відразу на обох за рахунок організації підключень vPC від FEX до пари комутаторів, а узгодження налаштувань для портів FEX здійснюється, наприклад, за допомогою автоматичної синхронізації конфігурацій. У такій схемі можуть використовуватися підключення vPC і до серверів, що дозволяє говорити про дворівневий vPC – від комутаторів до FEX і від FEX до серверів. Дана схема одержала назву Enhanced VPC (EvPC).

Але повернемося до реалізацій технології TRILL. Вона покладена й в основу рішення Virtual Cluster Switching – «фабрики» Ethernet, розробленою компанією Brocade. Правда, замість протоколу IS-IS у рішенні компанії використовується протокол Fabric Shortest Path First (FSPF), запозичений з миру

Fibre Channel. Як указують фахівці компанії, протокол FSPF дозволяє кожному комутатору VCS одночасно «бачити» всі вхідні в «фабрику» пристрою й вибирати маршрути з урахуванням стану всієї топології. Комутація трафіку між двома кінцевими пристроями у фабриці здійснюється в режимі балансування навантаження, при якому використовуються всі можливі еквівалентні шляхи з однаковими мінімальними вагами між кінцевими комутаторами. Фізичні канали Ethernet, що зв'язують два суміжних комутатори в Ethernet-фабриці, автоматично поєднуються в одну логічну групу Brocade Fabric Trunk.

Восени 2012 року Brocade представила VDX 8770 – перший модульний комутатор у лінійці пристроїв VDX, призначених для побудови Ethernet-фабрик (до цього будівельникам таких «фабрик» були доступні тільки пристрої з фіксованою конфігурацією). Представлений комутатор істотно збільшує масштабованість і продуктивність Ethernet-фабрики, у якій може налічуватися більше 8000 портів. Як повідомляють у компанії, рішення VDX зараз застосовуються в більш ніж 700 інсталяцій по усьому світі.

Варіант 5. Shortest Path Bridging

Технологія SPB описана в прийнятому інститутом IEEE у березні 2012 року стандарті 802.1aq. У багатьох відносинах вона є конкурентом TRILL. Цікава історія питання: розроблювачі TRILL спочатку запропонували свою технологію для стандартизації як заміна STP інституту IEEE, але, одержавши відмову, звернулися до IETF. Тоді чиновники IEEE ініціювали розробку SPB.

Для збору відомостей про топологію мережі SPB, так само як і TRILL, використовується протокол IS-IS. Однак реалізовані SPB механізми передачі кадрів відрізняються від тих, що використовуються в TRILL. При вході в домен SPB до стандартного кадру Ethernet «приклеюється» додатковий тег з ідентифікатором VLAN (VID), що «знімається» на виході з домену. В SPB є дві модифікації: одна – SPBV – використовує формат кадру 802.1ad (Q-in-Q), а інша – SPBM – 802.1ah (MAC-in-MAC). Другий варіант дозволяє підтримувати

значно більше число вузлів, що робить його підходящої для використання в більших мережах сервісів-провайдерів.

Про підтримку SPB заявили такі компанії, як Alcatel-Lucent, Avaya і, причому вони затверджують про проведення успішних тестів на сумісність. Як бачимо, табори прихильників TRILL і SPB являють собою практично непересічні безлічі, однак є й виключення. Принаймні одна компанія, HP, заявила про намір підтримувати обидві ці технології.

До лідерів на ринку рішень для мережної інфраструктури ЦОД відносяться компанії Cisco, HP, Extreme Networks, Brocade і Juniper.

3.2 Розробка структурної схеми

У даному проекті застосуємо технологію Fibre Channel п'ятого покоління, що забезпечує передачу даних зі швидкістю 10 (для зв'язку між ЦОД) і 16 Гбіт/с. Ці комутатори з надійністю «шість дев'яток» використовують практично всі провідні виробники серверів. Їхньої особливості – апаратний стиск (майже дворазове) і шифрування трафіку, діагностичний порт ClearLink, ПЗ, що розроблений у проекті, для керування інфраструктурою Fabric. Оптичні канали UltraScale ICL (InterChannel Link) використовуються для зв'язку до дев'яти директорів (наприклад, DCX 8510) на відстані до 100 м (така конфігурація застосовується при побудові ядра мережі). Корекція помилок FEC (Forward Error Correction) поліпшує роботу встаткування – до 11 біт виправляється без повторного пересилання пакета.

З розвитком Fabric замовники звичайно приходять до відказостійкої конфігурації – двом ЦОД із синхронною реплікацією даних по протоколі FC (через волоконно-оптичні канали DWDM). Іноді додають і третю площадку, віддалену на велику відстань. Асинхронна реплікація даних на неї здійснюється по FCIP. При відмові одного із ЦОД можна буде продовжувати роботу.

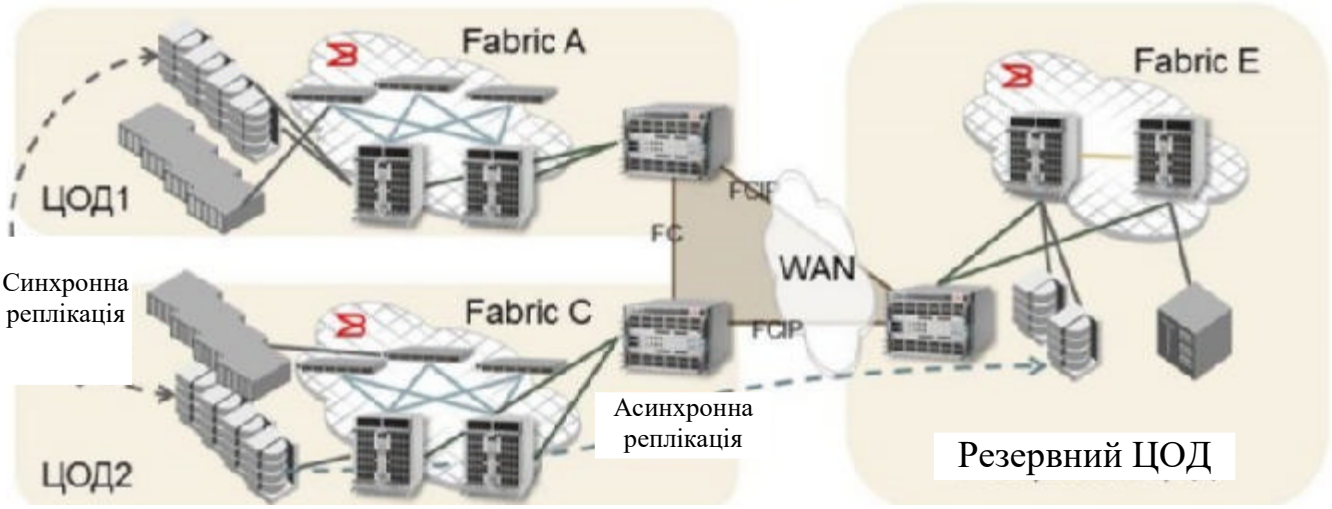

Рисунок 3.1 – Структурна схема системи

Проблема подальшого вдосконалювання технології FC полягає не стільки в підвищенні швидкості або надійності, скільки в керуванні усе більше складними інфраструктурами. Технологія Fabric націлена на підвищення надійності, зниження OPEX і оптимізацію роботи додатків, прискорене впровадження, а також швидке усунення проблем.

Fabric використовує апаратні функції спеціалізованих мікросхем Brocade Condor 3 ASIC і ПЗ, що розроблено в проекті. Його завдання – запобігання аварій, прискорене відновлення, оптимізація продуктивності додатків і прискорення впровадження нових систем.

Condor 3 є основою лінійки комутаторів з підтримкою Fibre Channel 16 Гбіт/с. Зберігання даних на флеш-накопичувачах вимагає ще більш високих швидкостей, тому йде активна робота над технологією FC шостого покоління – 32/128 Гбіт/с.

Для підвищення надійності передачі даних в Gen 6 FC стандартизується використання FEC. Також ведеться робота над стандартизацією паралельного Fibre Channel (32x4 = 128 Гбіт/с). Як очікується, перші комутатори Gen 6 з'являться на ринку в 2016 році. Нові швидкості пред'являють особливі також вимоги до модулів SFP і кабельній інфраструктурі.

FC-фабрики – класичне рішення Fabric високої надійності для побудови мереж зберігання даних ЦОД і комунікацій між площадками на великих відстанях.

Ethernet-фабрики – віртуальний розподілений Ethernet-комутатор. Фактично, це новий погляд на мережу передачі даних, один логічний комутатор, яким можна управляти з високим ступенем автоматизації, єдина точка керування. Зараз поряд з фабриками FC лінійка Fabric містить у собі Ethernet-фабрики – основні її рішення для ЦОД.

Ethernet-фабрика Fabric – віртуальний розподілений комутатор рівня доступу й агрегування. Вона практично не вимагає налаштування – підтримує «самоформування» і автовиявлення пристроїв, будь-які топології, управляється як один віртуальний комутатор. Для запуску фабрики досить задати її домен і ID.

Після цього можна використовувати єдину консоль керування всіма комутаторами), а для підвищення надійності підтримується топологія Full Mesh, що зберігає працездатність системи навіть при виході з ладу одного комутатора.

Фабрика також готова до конвергенції (FC 16/FCoE 10 Гбіт/с). Вона передбачає три рівні балансування трафіку, може агрегувати (без попереднього налаштування) велика кількість каналів. Завдяки покадровому балансуванню забезпечується розподіл навантаження між портами комутаторів.

За допомогою протоколу ECMP здійснюється балансування між еквівалентними за вартістю шляхами, що з'єднують кінцеві точки (незалежно від числа хопів). А балансування навантаження між шлюзами VRRP-E L3 (до чотирьох шлюзів) поліпшує масштабованість і надійність рішення.

Fabric технологія дає можливість створювати усередині VCS «віртуальні фабрики». Виділення логічних фабрик для кожного користувача в рамках сумісно-використовуємої фізичної фабрики на основі TRILL Fine-Grained Labels (IETF draft) забезпечує можливості перевикористання VLAN. У рамках кожної фізичної фабрики підтримується до 8000 віртуальних фабрик VCS.

Ethernet-фабрика може одержувати інформацію про віртуальні машини, їх MAC- і IP-адреси, групи VLAN. Ця інформація автоматично заноситься в Ethernet-фабрику й за замовчуванням створюються «профілі портів», до яких можна застосовувати різні налаштування, наприклад, задавати якість обслуговування QoS. Налаштування можна зробити один раз. Надалі фабрика сама застосовує задані політики за MAC-адресою, на якому б з портів він не з'явився. Одна фабрика поєднує до 48 комутаторів.

Fabric підтримує також VXLAN-шлюз VCS (VTEP) для VMware NSX – міст між віртуальними й фізичними ресурсами. Його можна використовувати для інтеграції із класичними мережами, що не підтримують VXLAN.

Ethernet-фабрику VCS повністю підтримують комутатори Fabric VDX серії 6740 і маршрутизатори 8770. У серії 6740 (SDN-ready) можна використовувати протокол OpenFlow 1.3. Контролер SDN повинен з'явитися в Fabric до кінця цього року. Він буде сполучати функції контролера й комутатора SDN, продовжуючи лінійку VDX 6740. Протокол OpenFlow підтримують також продукти Fabric Fibre Channel.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Функціональна схема складається з двох великих блоків:

– Блок визначення об’єктів моніторингу мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Блок визначення функцій моніторингу мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

З рисунку видно, що блок визначення об’єктів моніторингу мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric локальної мережі складається з трьох блоків:

– Моніторинг трафіку мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Моніторинг обладнання мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Моніторинг ресурсів мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Моніторинг трафіку мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric використовується для контролю вхідного та вихідного трафіку. 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.

Рисунок 3.2 – Функціональна схема системи

Моніторинг обладнання мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric включає в себе побудову списку наявного обладнання та здійснення його контролю. До мережного обладнання, що підлягає моніторингу мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric, відносяться: персональні комп'ютери, ноутбуки, сервери, принтери, IP-телефони.

Моніторинг ресурсів мережі віртуалізованих ЦОД з використанням технологічних рішень Fabricдозволяє переглядати та завантажувати наявні в мережі ресурси, а також розміщувати чи приховувати для загального доступу свої ресурси. До ресурсів локальної мережі відносяться: файли, мультимедіа, бази даних, сервіси інформаційної безпеки, список користувачів.

Блок визначення функцій моніторингу мережі віртуалізованих ЦОД з використанням технологічних рішень Fabric складається з наступних блоків:

– Робота з ресурсами мережі.
– Робота з сесіями.
– Моніторинг трафіку.
– Робота з файлами.
– Монітор з'єднань.
– Статистика подій.
– Функції для роботи з мережею.

Розглянемо детальніше кожний з блоків.

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.

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.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврської дипломної роботи, наведена на рисунку 3.3.

При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи.

Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

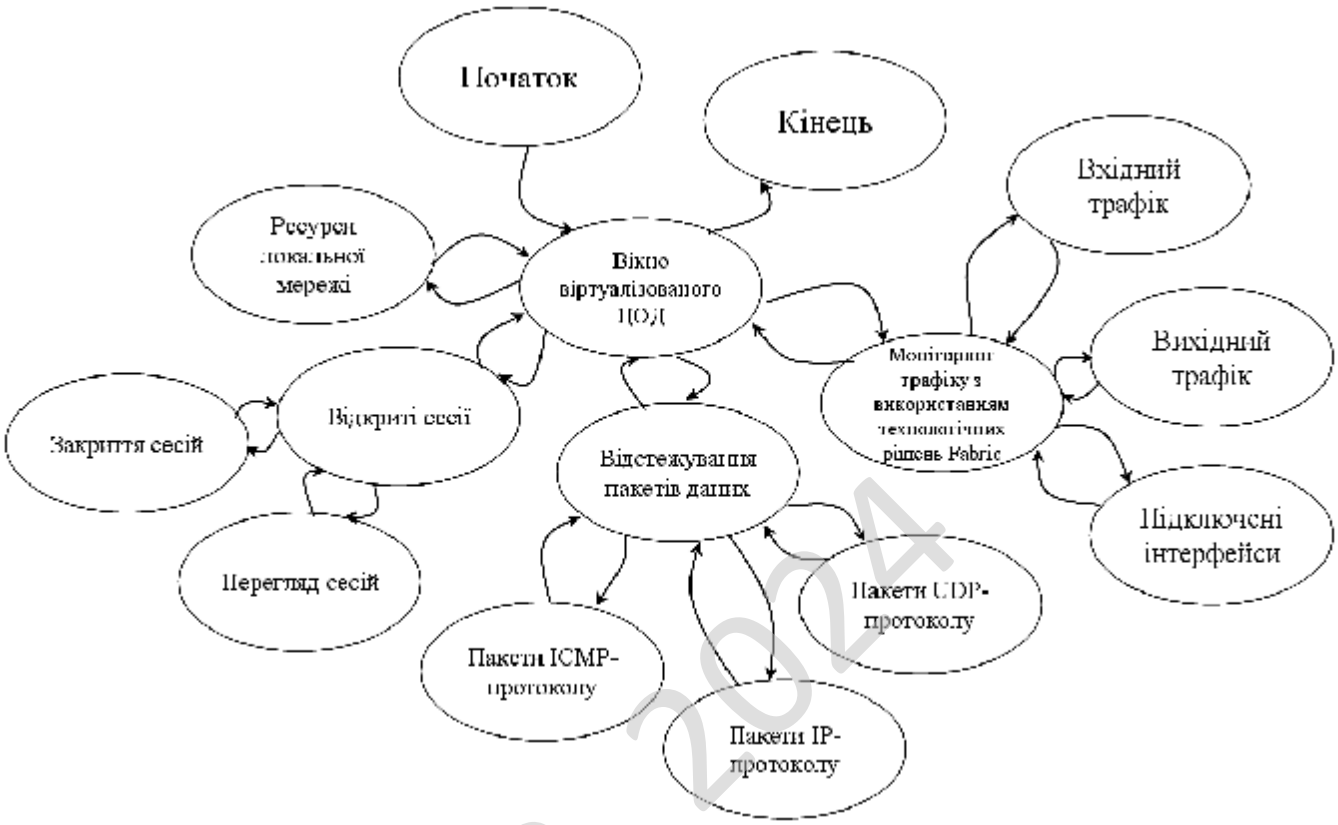

Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю віртуалізованих ЦОД з використанням технологічних рішень Fabric.

На рисунку 4.1 наведено блок-схему основної програми, на рисунку 4.2 зображено роботу підпрограм.

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між

функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

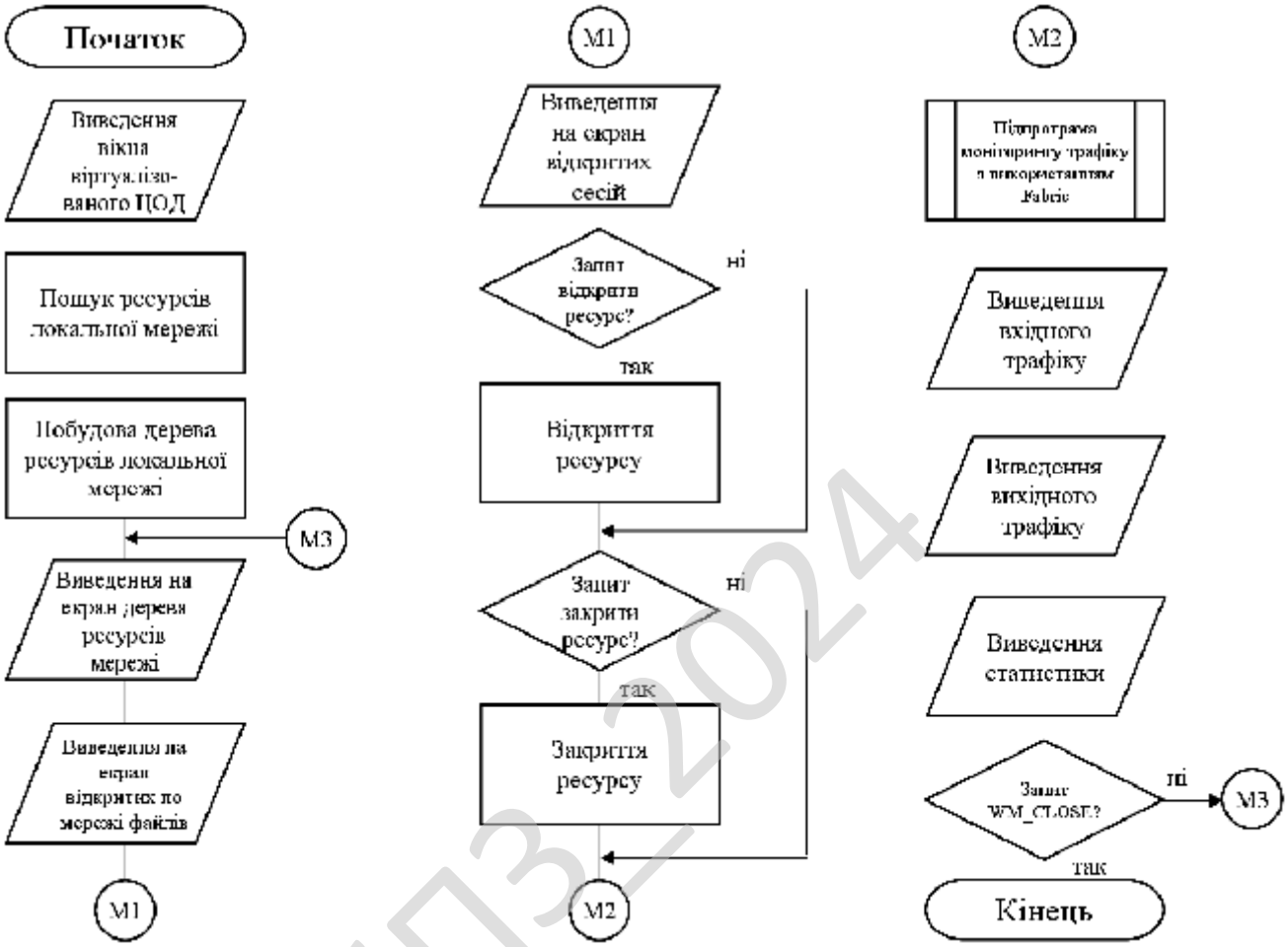

Рисунок 4.1 – Блок-схема основної програми

Функціональні схеми можуть виконуватися в укрупненому і розгорненому вигляді. У першому випадку на схемі зображають найважливіші блоки системи і зв'язки між ними.

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

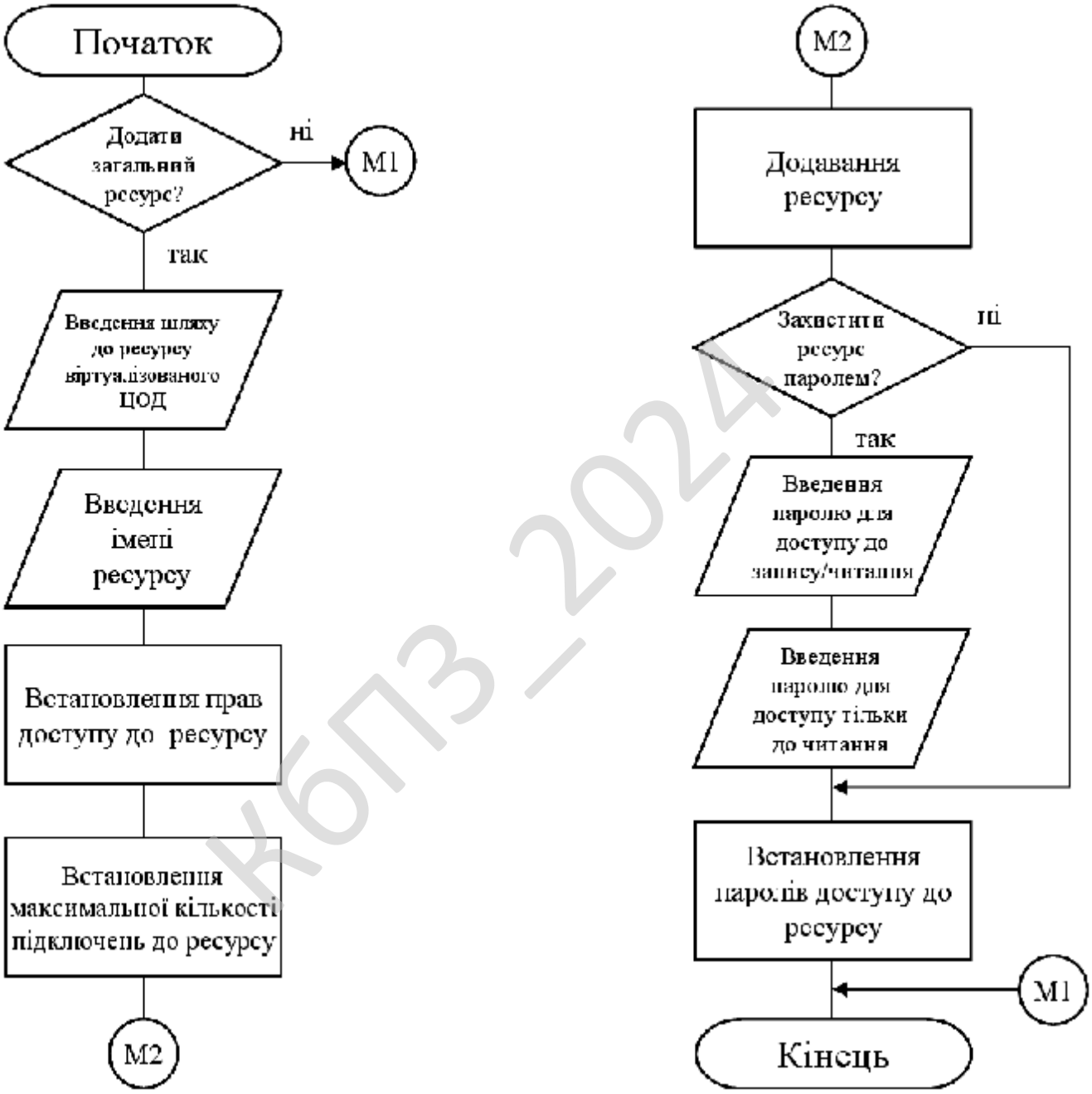

Рисунок 4.2 – Блок-схема роботи підпрограми

Процес – це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення,

візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

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

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

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

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

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

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

Для отримання списку всіх відкритих загальних інструкцій в програмі використовується процедура TMainForm.btnGetSharesClick:

В цій процедурі використовуються наступні змінні:

```
i:Integer;
FLibHandle : THandle;
ShareNT : PShareInfo2Array;
entriesread,totalentries:DWORD;
Share : array [0..512] of TShareInfo50;
pcEntriesRead,pcTotalAvail:Word;
OS: Boolean;
```

Тут PShareInfo2Array визначається як вказівник на масив вказівників на дані типу TShareInfo2, які являють собою структуру із наступними полями:

```
TShareInfo2 = packed record
  shi2_netname : PWChar; // мережне ім’я
  shi2_type: DWORD;  // тип ресурсу
  shi2_remark :PWChar;  // коментар
  shi2_permissions: DWORD; // змінна доступу
  shi2_max_uses : DWORD; // кількість максимальних підключень
  shi2_current_uses : DWORD; // кількість поточних підключень
  shi2_path : PWChar; // шлях до ресурсу
  shi2_passwd : PWChar; // пароль
```

Відповідно,

```
PShareInfo2 = ^ TShareInfo2;
TShareInfo2Array = array [0..512] of TShareInfo2;
PShareInfo2Array = ^ TShareInfo2Array;
```

Змінною цього типу є ShareNT. Аналогічно вводиться змінна TShareInfo50. Спершу завантажуємо бібліотеку та зв'язуємо функцію NetShareEnumNT:

```
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;
```

У випадку вдалого виклику проводиться обробка результатів:

```
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));
```

Подібним чином проводиться одержання всіх відкритих ресурсів для операційних систем Windows.

Розглянемо, як відбувається закриття ресурсів. Для цього використовується процедура TMainForm.btnCloseSharesClick, в якій вводяться наступні змінні:

```
OS:Boolean;
FLibHandle : THandle;
Name9x:array [0..12] of Char;
NameNT:PWChar;
i:Integer;
ShareName: String;
```

Для Windows закриття відбувається наступним чином:

```
FLibHandle := LoadLibrary(‘ NETAPI32.DLL’ );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,’ NetShareDel’ );
    if not Assigned(NetShareDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
```

Далі виділяється пам’ять під змінну:

```
GetMem(NameNT,i);
```

Після цього видаляється ресурс та звільняється пам’ять:

```
StringToWideChar(ShareName,NameNT,i);
    NetShareDelNT(nil,NameNT,0);
    FreeMem(NameNT);
```

Одержання списку сесій відбувається з використанням TMainForm.btnGetSessionsClick, для якої наведемо основну частину виконуваного коду:

```
        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
          begin
            Caption := string(SessionInfo502^[i].sesi502_cname);
SubItems.Add(SessionInfo502^[i].sesi502_username);
            SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
            SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time));
SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));      end;
```

Змінні, які використовуються в цій процедурі:

```
SessionInfo50: array [0..512] of TSessionInfo50;
SessionInfo502 : PSessionInfo502Array;
TotalEntries,EntriesReadNT: DWORD;
```

Тип PSessionInfo502Array визначається наступним чином:

```
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;
```

Для завершення сесії використовується TMainForm.btnCloseSessionClick:

```
FLibHandle := LoadLibrary(' NETAPI32.DLL' );
if FLibHandle = 0 then Exit;
@NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
if not Assigned(NetSessionDelNT) then
begin
  FreeLibrary(FLibHandle);
  Exit;
end;
CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
NetSessionDelNT(nil,CNameNT,nil);
```

Для одержання списку відкритих файлів слугує процедура TMainForm.btnGetFilesClick. В цій процедурі використовуються такі змінні:

```
FLibHandle : THandle;
FileInfoNT: PFileInfo3Array;
FileInfo9x: array [0..512] of TFileInfo50;
TotalEntries,EntriesReadNT: DWORD;
EntriesRead,TotalAvial: Word;
i:integer;
```

Тут змінна FileInfoNT, яка задає масив відкритих файлів (точніше

показник на масив) задається як змінна типу PFileInfo3Array:

```
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;
```

Відкриті файли записуються до структури:

```
    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
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id));
        SubItems.Add(FileInfoNT^[i].fi3_pathname);
        SubItems.Add(FileInfoNT^[i].fi3_username);
      end;
```

Тут Caption є ідентифікатором, поле fi3_pathname означає шлях до файлу, а поле fi3_username – ім’я користувача.

Закриття файлу відбувається за допомогою TMainForm.btnCloseFileClick:

```
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));
```

Вхідний та вихідний трафік визначаються за допомогою процедури TMainForm.tmrTrafficTimer. Спершу вводиться допоміжна функція, що перетворює MAC адресу до традиційного виду.

```
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;
```

Визначаємо спеціальний тип, щоб можна було передати у функцію масив type TMAC = array [0..7] of Byte. Як перше значення приймається масив, друге значення – розмір даних у масиві.

В самій процедурі TMainForm.tmrTrafficTimer використовуються наступні змінні:

```
FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
```

Тут таблиця Table має тип TMibIfTable, який описується таким чином:

```
PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;
```

```
    dwOperStatus: DWORD;          dwLastChange: DWORD;
    dwInOctets: 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;
```

При визначенні вхідного-вихідного трафіку після вимкнення таймера виконуються наступні операції по визначенню кількості відправлених та прийнятих байт:

```
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));
SubItems.Add(IntToStr(Table.Table[i].dwOutOctets));
```

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою алгоритму DES.

DES є класичною мережею Фейштеля із двома гілками. Дані шифруються 64-бітними блоками, використовуючи 56-бітний ключ. Алгоритм перетворить за кілька раундів 64-бітний вхід в 64-бітний вихід. Довжина ключа дорівнює 56 бітам. Процес шифрування складається із чотирьох етапів. На першому з них виконується початкова перестановка (IP) 64-бітного вихідного тексту (забілювання), під час якої біти переставляються у відповідності зі стандартною таблицею. Наступний етап складається з 16 раундів однієї й тої ж функції, що використовує операції зрушення й підстановки. На третьому етапі ліва й права

половини виходу останньої (16-й) ітерації міняються місцями. Нарешті, на четвертому етапі виконується перестановка IP^{-1} результату, отриманого на третьому етапі. Перестановка IP^{-1} інверсна початковій перестановці.

Рисунок 4.3 – Загальна схема DES

Праворуч на рисунку показаний спосіб, яким використовується 56-бітний ключ. Спочатку ключ подається на вхід функції перестановки. Потім для кожного з 16 раундів підключ K_i є комбінацією лівого циклічного зрушення й перестановки. Функція перестановки та сама для кожного раунду, але підключи K_i для кожного раунду виходять різні внаслідок повторюваного зрушення біт ключа.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення віртуалізованих ЦОД з використанням технологічних рішень Fabric складається з наступних функціональних блоків:

– Навігаційне меню: Файли; Ресурси; Сесії; Трафік; Параметри; Довідка.

– Підрозділу представлення древа мережних ресурсів.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ: Відкрити ресурс; Закрити ресурс; Додати ресурс; Оновити; Моніторинг; Статистика.

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

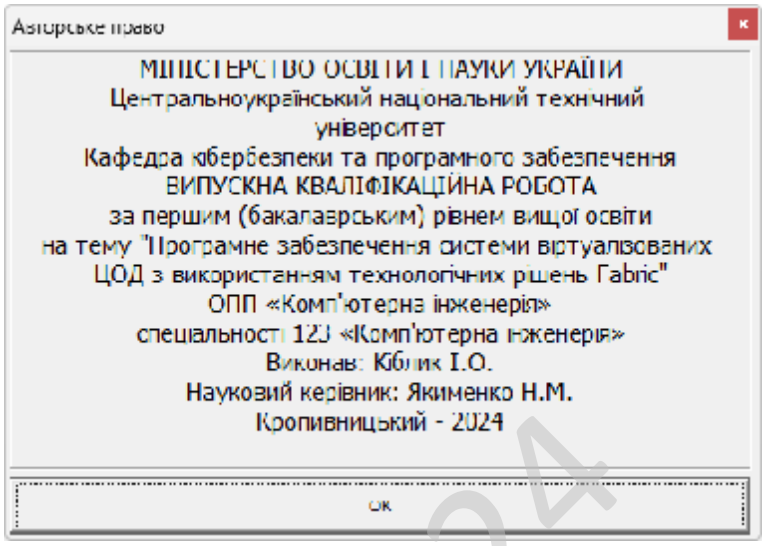

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію. Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт. Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення. Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– Досліджена система віртуалізованих ЦОД з використанням технологічних рішень Fabric.

– На основі отриманих результатів досліджень створена програмна реалізація системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віртуалізованих ЦОД з використанням технологічних рішень Fabric.

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

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

Програма реалізована на мові високого рівня Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи віртуалізованих ЦОД з використанням технологічних рішень Fabric. 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 DES.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оліфер В.Г. Комп'ютерні мережі. Принципи, технології, протоколи. Підручник / В.Г. Оліфер, Н.А.Оліфер. – [5-е вид.]. – 2016. – 944 с.

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

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

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

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

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

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

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

9. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

10. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

11. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

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

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

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

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

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

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

18. 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.

19. 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.

20. 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.

21. 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.

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

23. 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.

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

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

26. 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.

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

28. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

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

30. 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.

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

32. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

33. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду

абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

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

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

36. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

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

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

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

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

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

42. 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.

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

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

45. 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).

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

47. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 132-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) 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;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи віртуалізованих ЦОД з використанням технологічних рішень Fabric;

– 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, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 67 аркушів.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

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

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 5.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

першим (бакалаврським) рівнем вищої освіти

_____ Якименко Н.М.

Програмне забезпечення системи віртуалізованих ЦОД з використанням технологічних рішень Fabric

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2024 року

Файл TCP_IP.pas- монітор TCP/IP з'єднань віртуалізованих ЦОД з використанням технологічних рішень Fabric

```
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 віртуалізованих ЦОД з використанням технологічних рішень Fabric

```
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.
```

Основна програма

Файл Monitor_Net_Data_Center_Fabric. dpr основної програми

```
program Monitor_Net_Data_Center_Fabric;

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 віртуалізованих ЦОД з використанням
технологічних рішень Fabric
  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( ' Exo запит        : ' + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo 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( 'Exo запит        : ' + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Exo 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 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.
```