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 управління у розподілених Cloud-системах GENI”

Виконав здобувач вищої освіти
IV курсу, групи КМ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Китун Е.В.
«____» _____ 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

Китуну Едуарду Васильовичу
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи управління у розподілених Cloud-системах GENI*

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 № 133-02 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 управління у розподілених Cloud-системах GENI*

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
_____ Китун Е.В.
(прізвище та ініціали)

АНОТАЦІЯ

Китун Е.В. Програмне забезпечення системи управління у розподілених Cloud-системах GENI. 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 управління у розподілених Cloud-системах GENI.

Метою розробки є програмне забезпечення системи управління у розподілених Cloud-системах GENI.

Результат роботи – програмна реалізація системи управління у розподілених Cloud-системах GENI.

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 10.4.1.

Ключові слова: комп'ютерна інженерія, Cloud-система, GENI

ABSTRACT

Kytun E.V. Management system software in distributed GENI Cloud systems. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the management system in distributed cloud-systems of GENI.

The goal of the development is the management system software in distributed GENI Cloud systems.

The result of the work is the software implementation of the management system in distributed GENI Cloud systems.

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 10.4.1 environment.

Keywords: computer engineering, Cloud system, GENI

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 2
ВСТУП 3
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 5
1.1 Призначення системи 5
1.2 Область застосування 5
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 7
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти 7
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 16
2.3 Розгорнута постановка завдання 22
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 23
3.1 Опис функціонування системи 23
3.2 Розробка структурної схеми 31
3.3 Розробка функціональної схеми 35
3.4 Розробка діаграми процесів 44
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 46
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 46
4.2 Захист розробленого програмного забезпечення 56
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 58
6 ОСНОВНІ ВИСНОВКИ 62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	Internet CoSntrol Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	Internet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted random early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

ВСТУП

Актуальність теми. Світове наукове співтовариство активно вивчає й тестує технічні рішення, які повинні стати основою майбутнього Інтернету й розподілених хмарних систем. Зокрема, такі роботи ведуться в рамках американського проекту Global Environment for Network Innovations (GENI) і європейського Future Internet Research & Experimentation (FIRE). У багатьох університетах і лабораторіях по усьому світі вже створені тестові стенди FIDC – Future Internet and Distributed Cloud. Такі стенди являють собою стійки з обчислювальними системами (серверами), засобами зберігання інформації (СЗД) і мережними пристроями. Вони можуть складатися з різного встаткування й ПЗ, але в них реалізовані дві ключові характеристики: глибока програмуємість і можливість поділу на «шари» (slice), які стануть ключовими для мереж майбутнього.

Під глибокої програмуємістю розуміється можливість програмними засобами визначати (конфігурувати й програмувати) всі наявні в складі тестового стенда ресурси: сервери, СЗД, мережні засоби. Така програмуємість досягається, зокрема, завдяки вже добре відомому підходу програмно-конфігуруємих мереж – Software Defined Network (SDN). Цей підхід дозволяє дослідникам (а в майбутньому – замовникам) формувати віртуальні мережі для забезпечення взаємодії ресурсів для конкретного додатка або експерименту.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи управління у розподілених Cloud-системах GENI.

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

– Огляд існуючих систем управління у розподілених Cloud-системах GENI.

– Дослідження системи управління у розподілених Cloud-системах GENI.

– Програмна реалізація системи управління у розподілених Cloud-системах GENI.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі управління у розподілених Cloud-системах GENI.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління у розподілених Cloud-системах GENI, 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

В умовах постійно зростаючих обсягів даних і більш високих вимог до продуктивності інфраструктур, питання про вибір ефективної й продуктивної системи зберігання даних є актуальним для багатьох підприємств.

Сучасна система зберігання повинна забезпечувати високий рівень безпеки зберігання. Немаловажним параметром є й можливості масштабування систем, оскільки багато організацій просто не можуть собі дозволити придбати відразу дорогий варіант. Найчастіше взагалі потреби в більше дорогому дисковому масиві виникають не відразу, а в міру розширення організації.

1.2 Область застосування

Системи зберігання даних були розроблені для консолідації блокових і файлових рішень зберігання в однієї системи зберігання. Ці системи зберігання прекрасно підійдуть для підтримки різних систем віртуталізації, поштових систем, транзакційних СУБД, корпоративних порталів, ERP-систем і інших подібних рішень.

Унікальною характеристикою цих систем зберігання є їхня гнучка масштабованість. Як правило, рішення аналогічного класу не дозволяють істотно збільшувати кількість системних контролерів або перевищувати деякий максимальний поріг обсягу даних. Для подолання цих обмежень користувач змушений міняти всю систему зберігання даних.

Здобуваючи кожну з конфігурацій, у тому числі й саму молодшу, ви можете нарощувати її можливості до максимальних характеристик у міру необхідності.

Ще однією важливою особливістю цих систем є інтелектуальні мікрокод, що здійснює аналіз навантаження в реальному часі й оптимізує використання швидких SSD-дисків як додаткова кеш-пам'ять. Завдяки цьому збільшується ефективність роботи цих систем.

Системи підтримують функції синхронної й асинхронної реплікації, гарантуючи надійний захист даних у розподілених інфраструктурах і на будь-яких відстанях.

У розпорядження користувача надається повний набір сучасних функцій: реплікація, шифрування, моментальні знімки, Flash (SSD) Cache, Storage Tiering.

Ще одною перевагою систем є їхня уніфікованість. Системи зберігання даних оснащуються єдиним уніфікованим контролером SAN/NAS, що рятує користувача від необхідності покупки двох різних масивів (блокового й файлового).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи управління у розподілених Cloud-системах GENI, 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

SAS/DAS – Server Attached Storage/Direct Attach Storage

Рішення IBM ринку систем зберігання JBOD представлені наступними моделями: EXP2512, EXP2524, EXP3000, EXP3512, EXP3524, EXP395.

Лінійка HP систем такого ж класу: Modular Disk System HP 600; Дискові Полки (Disk Enclosure): HP D2000 і HP D6000.

Dell: PowerVault MD1000, MD1200, MD3000, MD3000i.

Кластерні системи зберігання

Як приклад рішень даного типу можна привести рішення HP StoreVirtual на основі кластера із систем HP P4000 LeftHand.

Системи зберігання з безліччю контролерів

Як приклад таких систем можна привести рішення HP StoreServ 3Par. Основу лінійки 3PAR StoreServ 7000 становлять дві моделі, 3PAR StoreServ 7200 і 3PAR StoreServ 7400. Перша оснащується двома дисковими контролерами, а в другу можна опціонально встановлювати два або чотири. У контролери масиву встановлюється великий обсяг кешу: 24 GB для StoreServ 7200 і 32 GB для StoreServ 7400.

Масив **3PAR StoreServ 7400** у чотирьох-контролерній конфігурації – це фактично по архітектурі й функціоналові « High-End масив за ціною масиву класу midrange». У дискові полки можна встановлювати диски SSD, SAS і SATA форм-факторів SFF (2,5 дюйми) і LFF (3,5 дюйми), причому в одному полку можуть спільно працювати різні типи дисків одного форми-фактора. Продуктивність контролерів при цьому не стає вузьким місцем при використанні

високопродуктивних SSD-дисків: середня пропускна здатність становить 2,5 Гбайт у секунду на диск для 3PAR StoreServ 7200 і 4,8 Гбайт для 3PAR StoreServ 7400. При цьому в максимальній конфігурації середня кількість IOPS (операцій уведення-виводу) у секунду становить 150.000 і 320.000 відповідно.

У систему вже убудовані чотири 8-гігабітних порти FiberChannel, а загальна їхня кількість може бути розширене до 12 у випадку 3PAR StoreServ 7200 і 24 для 3PAR StoreServ 7400. Також опціонально можна встановити порти FCo або iSCSI, 4 в 3PAR StoreServ 7200 або 8 в 3PAR StoreServ 7400.

Системи зберігання зі швидким доступом на флеш дисках

Рішення **IBM FlashSystem** забезпечують збільшення швидкості обробки транзакцій, знижуючи час відгуку на 90%, скорочують час обробки даних на 85%.

Наприклад, при роботі **IBM FlashSystem** з додатком Oracle RAC, на кластер із чотирьох вузлів, два мільйони запитів обробляються за 1,3 мінути. У той час як стандартні системи зберігання (без твердотільних накопичувачів SSD) виконують обробку такого ж обсягу за 12,25 мінут, тобто збільшення швидкості більш ніж в 9 разів.

Рішенням компанії Hewlett Packard є HP VMA-series Memory Array – масив пам'яті корпоративного класу **HP VMA3205/VMA3210** включає апаратний RAID на базі модулів flash-пам'яті з підтримкою "гарячого" підключення для забезпечення надійного захисту даних і безпикових затримок, тривалістю менш 100 мікросекунд.

Flash VIMMs – агрегування мікросхем

– 5ТБ і 10ТБ «сирих» в 3U (VMA3205, VMA3210) – SLC.

– постійна швидкість запису (не деградує згодом).

– можливість гарячої заміни модулів.

Flash vRAID – захист даних

– не залежить від механіки, у тому числі по затримках.

– 80% ефективності (проти 50% в RAID-1).

– Fail-in-place (самовідновлення, заміна великих модулів).

– надійність 99.999%.

– термін служби 5 років при макс. швидкості запису (8ТБ у годину).

Продуктивність

– 250 000 IOPS на апарат (до 2 700 000 IOPS на шафу).

– Рішення справляється з 200 000 000 транзакцій у годину.

– Затримка <100 мікросекунд (а в дисків в 30 разів більше).

– Передбачувана затримка (а в дисків розкид через механіку).

– 1,4Гбайт/сек на апарат (понад 12Гбайт/сек на шафу).

– Підключається прямо по PCIe до сервера.

Огляд програмного забезпечення для масивів OceanStor V3

Програмні функції систем зберігання даних відіграють важливу роль. Вони багато в чому визначають вартість рішення і його функціональність. Приведемо огляд програмних функцій систем зберігання даних Huawei OceanStor v3. Основне призначення цього програмного забезпечення – збільшити продуктивність систем зберігання даних, захистити інформацію й спростити керування системами. У третім поколінні масивів з'явилися функції SmartErase і SmartMigration, змінилася модель ліцензування функцій SmartThin, SmartMulti-tenant, SmartMigration, SmartErase.

SmartVirtualization: переміщення даних між системами

Механізм SmartVirtualization переміщає дані між системами. За рахунок цього можна віртуалізувати СЗД сторонніх вендорів у відповідності зі списком сумісності. Механізм застосовується при консолідації гетерогенних масивів. Завдяки цій функції застарілі системи зберігання, які вже є в складі інфраструктури використовуються разом з масивами Huawei. Всі сторонні масиви, що входять у список сумісності, консолідуються через одну СЗД, на новому масиві створюється віртуальний LUN. Міграція виробляється непомітно для хосту.

SmartMotion: горизонтальне переміщення даних

Механізм дозволяє вирівняти внутрішнє навантаження накопичувачів. Розподіляють рівномірно дані й продуктивність при додаванні нових накопичувачів. Підтримує роботу при настроюванні ресурсів, забезпечує баланс між ємністю сховища й продуктивністю. Важливо пам'ятати, що для функціонування RAID 2.0+ на керуючому контролері й полках потрібна ліцензія.

SmartQo: інтелектуальний контроль якості послуг

Функція SmartQo потрібна при наданні критично важливих послуг, підійде для варіантів зі змішаними послугами. Дозволяє налаштувати пріоретизацію уведення-виводу: збільшити пріоритет виділеному LUN за рахунок інших. Політики прив’язуються до конкретного LUN, діапазон віддачі, затримка й пропускна здатність налаштовуються за принципом «не більше – не менш».

SmartTier: вертикальне переміщення даних

SmartTier управляє розміщенням даних і оптимізує використання ресурсів зберігання. Функція стежить, щоб архівні дані не займали місце на дорогих швидких накопичувачах, скорочуючи витрати енергії. Міграція відбувається в ручному й автоматичному режимі. Користувач може налаштувати тривалість моніторингу уведення-виводу й міграції даних, а також швидкість міграції. У системі доступні чотири міграційних політики: автоматична міграція, міграція на рівень із більше високою продуктивністю, міграція на рівень із більше низькою продуктивністю, без міграції.

SmartThin: динамічне виділення ємності

По суті це «тонкі» LUN. Якщо на додаток до цієї функції у вас придбана ліцензія SmartMigration ви можете робити «тонкі» LUN звичайними й навпаки. Функція SmartThin допомагає в режимі онлайн збільшити системну ємність основних систем обслуговування. До таких систем висувають високі вимоги безперервності діяльності, наприклад, банківські системи. Придасться ця функція й там, де важко точно оцінити ріст обсягу системних прикладних даних, наприклад, у службах електронної пошти. Щоб вирішити проблему

невикористаних ємностей ця функція виділяє фізичний простір на вимогу. Функція добре працює в застосуваннях з маленьким розміром блоків, тому що виділяє ресурси з розбивкою по 64 КБ.

SmartPartition: інтелектуальний поділ кеш-пам'яті

SmartPartition розділяє основні системні ресурси й підвищує продуктивність. Корисна при наданні змішаних послуг. Розділи кеш-пам'яті закріплюються за певними LUN, ізоляція сервісів робить роботу надійніше. Регулюється розмір на читання й запис. Рекомендується, щоб залишок вільного кешу становив не менш 50%. Цю ліцензію варто здобувати, якщо ви впевнені, що розмір кешу фіксований. Якщо такої впевненості ні, краще вибрати SmartQo.

SmartCache: інтелектуальне кешовані твердотільного накопичувача

Функція SmartCache використовує твердотільні накопичувачі як кеш-пам'ять зчитування. Працюючи з кешем ОЗП прискорює зчитування даних для LUN, файлових систем, прискорює дедуплікацію позначка-даних. Придасться для сервісів з більшою кількістю операцій уведення-виводу довільного зчитування малого обсягу, наприклад, оперативна обробка транзакцій, бази даних, файлові й веб-служби. Потрібно враховувати, що твердотільні накопичувачі використовуються в складі кешу монопольно. Для звичайних операцій вони недоступні. SSD-диски можна додавати й забирати на ходу, дисковий кеш використовується тільки для читання, а розмір блоків система визначає сама.

SmartErase: знищення даних

Функція видаляє дані, запобігаючи витік інформації двома способами: програмним методом на перезаписуваних носіях (трикратний перезапис) і що набудовується. У першому випадку використовуються 8-розрядний символ для перезапису всіх адрес, додаткові коди й випадковий символ для перезапису всіх адрес. У другому випадку дані генеруються на базі внутрішніх алгоритмів системи й використовуються при перезаписі всіх адрес LUN .

SmartMulti-Tenant: багатокористувальницький режим

Функція SmartMulti-Tenant розділяє дані ресурсів користувачів і закріплює за користувачами керування ресурсами. Придасться для ізоляції конфіденційної інформації (наприклад, кадрової або фінансової), а також ізоляції хост-середовища від виставляння рахунків з боку провайдеру. Всіма ресурсами й користувачами управляє адміністратор сховищ, а адміністратор групи користувачів управляє ресурсами окремої групи.

HyperSnap: захищає локальні дані на базі інкрементних копій.

Миттєвий віртуальний знімок – це один з різновидів моментальних знімків. По суті це копія вихідних даних, зроблена в певний момент часу. Створення такого знімка залежить від вихідного логічного пристрою LUN.

Механізм HyperSnap захищає локальні дані за допомогою інкрементних копій. Після запуску механізму створюється таблиця відповідностей, що містить взаємозв'язок структуруємих даних. Потім виконується запис копіюється.

HyperClone: захищає локальні дані на базі повних копій.

Механізм HyperClone – один з видів моментальних знімків. Він створює повні копії вихідних даних у певний момент часу. Механізм застосовується для резервного копіювання даних, захисту, аналізу й тестування даних. На відміну від класичного моментального знімка на створення клону потрібне певний час, клон не вантажить основний LUN. Швидкість клонування настроюється вручну, одночасно можна створювати до 16 клонів. Каскадування клонів немає. Клон постійно синхронізується з вихідним LUN, стан клону можна зафіксувати в будь-який момент. Якщо головний LUN ушкоджений, другорядні LUN залишаються недоторканими. Другорядний LUN можна зробити головним. У головного LUN може бути трохи другорядних.

HyperCopy: захищає дані між пристроями.

Механізм HyperCopy копіює дані в цільовий LUN з вихідного в рамках однієї системи зберігання або між декількома системами. Цей механізм відрізняється від клонування тим, що клон створюється в одній системі, а

копіювання відбувається в іншу. У момент копіювання хост не працює на уведення й вивід. Механізм не передбачає запис змін у копію (немає інкрементного копіювання).

HyperReplication: надає захист за допомогою аварійного відновлення між датами-центрами.

Механізм HyperReplication синхронізує головне й другорядне сховище в режимі реального часу, забезпечуючи повний захист цілісності даних. Механізм зводить до мінімуму втрати даних у випадку збоїти. Якщо віддалена СЗД не відповідає вчасно, реплікація автоматично перемикається із синхронного режиму в асинхронний. Реплікація може здійснюватися з 1 на 32 СЗД і навпаки. Старші моделі можуть активно реплікуватися з молодшими й навпаки.

HyperSnap: моментальний знімок

Механізм HyperSnap створює моментальні знімки. Моментальні знімки файлової системи охоплюють її цілком, зробити знімок окремої директорії або файлу не можна. Каскадування знімків немає. Під час створення знімка копіюється й зберігається тільки кореневий вузол файлової системи. Дані користувача не копіюються. На створення знімка йде від однієї до двох секунд.

Створення моментального знімка другого рівня

Для набору файлів не потрібно додаткового простору, тому що перед зміною даних набір файлів знімка й основна файлова система використовують те саме простір файлової системи. При видаленні знімків другого рівня вивільняються покажчики кореневого вузла й дані, які використовуються знімком безпосередньо. Ніяких змін у файловій системі при цій не відбувається. Спільно використовувані дані не віддаляються, відновлюється безпосередньо зайнятий простір.

Видалення моментального знімка другого рівня. Сірим кольором відзначені вузли, які будуть віддалені, синім – вузли, що використовуються.

Моментальні знімки віддаляються за лічені секунди, а простір, що вони займали поступово заповнюється фоновими завданнями.

HyperMirror: дзеркалювання томів

Технологія HyperMirror створює кілька фізичних копій LUN, дублюючи дані для захисту. За допомогою цієї технології підвищується надійність і доступність LUN. Принцип роботи HyperMirror аналогічний RAID1. Дзеркало тому можна перенести на другий масив, віддалений на відстань до 500 м. Дзеркалювання виробляється непомітно для хосту. Технологія корисна там, де критична висока доступність даних.

HyperReplication: віддалена реплікація

Функція HyperReplication реалізує віддалену асинхронну реплікацію між файловими системами. Це одна з технологій аварійного відновлення, вона управляє копіями даних, розміщених у декількох віддалених друг від друга сховищах. Дозволяє зберегти дані при виникненні збоїв.

HyperLock: технологія WORM

Технологія HyperLock працює так само як і звичайна файлова система. Потрібно тільки вказати тип носія WORM і властивості. Необхідна для захисту документів від зміни: записаний один раз файл можуть читати багато користувачів, але змінити його можна тільки через установлений час після створення документа.

Технологія HyperLock

Споконвічно всі файли перебувають у початковому стані, користувачі можуть змінювати їх. Якщо файл заблокований, його не можна видалити, перейменувати або змінити його вміст. Файл у стані «прострочений» доступний для перегляду, зчитування даних і видалення цілком. Інші операції з ним зробити не можна. Файл у стані «Доданий» можна дозаписати. Видалити або урізати його не можна.

Функції для блокових систем

Функціональні можливості для файлових систем

Пакет програм для підвищення ефективності.

– SmartVirtualization: Переміщення даних між системами.

– SmartMotion: Горизонтальне переміщення даних.

– SmartQo: Інтелектуальний контроль якості послуг.

– SmartTier: Вертикальне переміщення даних.

– SmartThin: Динамічне виділення ємності.

– SmartDedupe & SmartCompression: Інтелектуальна дедуплікація й стиск даних.

– SmartPartition: Інтелектуальний поділ кеш-пам'яті.

– SmartCache: Інтелектуальне кешовані твердотільного накопичувача.

– SmartErase: Знищення даних.

– SmartMigration: Переміщення логічного диска (LUN).

– SmartMulti-Tenant: Багатокористувальницький режим.

– SmartThin: Динамічне виділення ємності.

– SmartQo: Інтелектуальний контроль якості послуг.

– SmartPartition: Інтелектуальний поділ кеш-пам'яті.

– SmartCache: Інтелектуальне кешовані твердотільного накопичувача.

– SmartDedupe: Інтелектуальна дедуплікація даних.

– SmartCompression: Онлайн-стиск.

– SmartQuota: Керування квотою.

– HyperSnap: Моментальний знімок.

– HyperReplication: Віддалена реплікація.

– HyperLock: Технологія WORM (з однократним записом і багаторазовим зчитуванням).

Пакет програм для захисту даних:

– HyperSnap: захищає локальні дані на базі інкрементних копій.

– HyperClone: захищає локальні дані на базі повних копій.

– HyperCopy: захищає дані між пристроями.

– HyperReplication: надає захист за допомогою аварійного відновлення між датами-центрами.

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 управління у розподілених Cloud-системах GENI.

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

Існують різні варіанти організації доступу до систем зберігання:

– SAS/DAS (Server/Direct Attached Storage), система зберігання, підключена до сервера.

– NAS (Network Attached Storage), система зберігання, підключена до мережі;

– SAN (Storage Area Network), мережа зберігання даних.

– Сучасний підхід до побудови СЗД – сполучення технологій (SAS/iSCSI/FC).

– Кластерні системи зберігання.

– Системи зберігання з безліччю контролерів.

– Системи зберігання зі швидким доступом на флеш дисках.

SAS/DAS – Server Attached Storage/Direct Attach Storage

SAS/DAS (Server Attached Storage/Direct Attach Storage), система зберігання, підключена до сервера. Знайомий усім, традиційний спосіб підключення системи зберігання даних до високошвидкісного інтерфейсу в сервері, як правило, до SAS інтерфейсу. Використання окремого корпуса для системи зберігання в рамках топології SAS не є обов'язковим.

Основна перевага DAS/SAS у порівнянні з іншими варіантами – низька ціна й висока швидкодія з розрахунку одна система зберігання для одного сервера. Така топологія є самою оптимальною у випадку використання одного сервера, через який організується доступ до масиву даних. Але в неї залишається ряд проблем, які спонукали проектувальників шукати інші варіанти організації доступу до систем зберігання даних.

До особливостей DAS/SAS можна віднести:

– Доступ до даних залежить від ОС і файлової системи (у загальному випадку).

– Складність організації систем з високою доступністю.

– Висока швидкодія в рамках одного сервера.

– Зменшення швидкості відгуку при завантаженні сервера, що обслуговує систему зберігання.

Переваги:

– Досить низька вартість. По суті ця СЗД являє собою дисковий кошик з жорсткими дисками, винесену за межі сервера.

– Простота розгортання й адміністрування.

– Висока швидкість обміну між дисковим масивом і сервером.

Недоліки:

– Низька надійність. При виході з ладу сервера, до якого підключене дане сховище, дані перестають бути доступними.

– Низький ступінь консолідації ресурсів – вся ємність доступна одному або двом серверам, що знижує гнучкість розподілу даних між серверами. У результаті, необхідно закуповувати або більше внутрішніх жорстких дисків, або ставити додаткові дискові полки для інших серверних систем.

NAS – Network Attached Storage

Модуль NAS являє собою окремий комп'ютер, що може бути побудований на довільній архітектурі. Основним призначенням цього комп'ютера є надання сервісів для зберігання даних іншим пристроям у мережі. Операційна система й програми NAS-модуля забезпечують роботу сховища даних і файлової системи, доступ до файлів, а також контроль над функціями системи. Пристрій не призначений для виконання звичайних обчислювальних завдань, хоча запуск інших програм на ньому може бути можливий з технічної точки зору. Звичайно NAS пристрою не мають екрана й клавіатури, а управляються й налаштовуються по мережі, часто за допомогою браузера, підключаючись до пристрою по його

мережній адресі. По можливостях і форм-факторові мережні сховища розрізняються по кількості слотів для установки жорстких дисків: від одного до 12 і більше. Традиційно вважається, що чим більше слотів в NAS, тим до більше високого класу ставиться пристрій: для домашнього використання, для домашнього офісу й офісу малого підприємства, для середнього бізнесу, для виробництва й офісу корпорації. Втім, зв'язок між кількістю слотів і класом не абсолютна: більшістю компаній-виробників випускаються як моделі домашнього класу із чотирма слотами, так і моделі класу «для великого бізнесу» – із двома. Розподіл же залежить ще й від списку можливостей NAS, його конструкції й програмного забезпечення.

Як правило, енергоспоживання NAS на 90 % залежить від кількості й типу встановлених дисків, і лише потім – від убудованого процесора й пам'яті.

Особливості NAS:

– Виділений файл-сервер.

– Доступ до даних не залежить від ОС і платформи.

– Зручність адміністрування.

– Максимальна простота установки.

– Низька масштабованість.

– Конфлікт із трафіком LAN/WAN.

SAN – Storage Area Network

Мережа зберігання даних являє собою виділену мережу, що забезпечує доступ до консолідованого, блокового сховища даних. Системи SAN у першу чергу використовуються як пристрої зберігання, таких як дискові масиви, стрічкові бібліотеки, і оптичні накопичувачі з автоматичною зміною дисків. Сервери одержують до них доступ, і для операційної системи вони виглядають як локальні пристрої зберігання. SAN звичайно складається із власної мережі із систем зберігання, які звичайно не доступні для інших пристроїв по локальній мережі. Ціна й складність SAN систем значно впала на початку 2000-х років до рівня який дозволив їхнє більше поширення, як у секторі великого бізнесу, так і в

малих і середніх компаніях. SAN не надає абстракцію даних на рівні файлів, надаються тільки операції блокового рівня. Проте, файлові системи, побудовані поверх SAN, забезпечують файловий доступ, і відомі як файлові системи SAN або системи з поділюваними дисками.

До основних переваг SAN можна віднести практично всієї її особливості:

– Зручне централізоване керування.

– Відсутність конфлікту із трафіком LAN/WAN.

– Зручне резервування даних без завантаження локальної мережі й серверів.

– Висока швидкодія.

– Висока масштабованість.

– Висока гнучкість.

– Висока готовність і відказостійкість.

SAN не надає абстракцію даних на рівні файлів, надаються тільки операції блокового рівня. Проте, файлові системи, побудовані поверх SAN, забезпечують файловий доступ, і відомі як файлові системи SAN або системи з поділюваними дисками.

На сьогоднішній день функціонал SAN систем зберігання містить у собі наступний основний функціонал:

– Easy Tier – допомагає автоматично оптимізувати продуктивність і використання накопичувачів у багаторівневих і однорівневих системах. Зміст полягає в тім, що дані, до яких відбувається більше інтенсивний обіг, переміщаються на більше швидкі диски типу SSD, що випливає рівень – дані з меншою кількістю запитів – на SAS дисках, і дані з найменшою кількістю обігів зберігаються на дисках SATA.

– Снепшоти – (SnapShot, FlashCopy) створюють, копії фізичних розділів на певний момент часу для захисту користувальницьких даних, таких як відновлення файлів або резервного копіювання. FlashCopy це логічний еквівалент фізичного розділу, але який створюється набагато швидше ніж фізичне

копіювання, а також з мінімальними порушеннями роботи додатків і роботи користувачів.

– Копіювання тому (VolumeCopy) – створює фізичну копію роздягнула. VolumeCopy використовується разом з FlashCopy для того щоб створити фізичну копію даних з мінімальним порушенням роботи додатків і виробничого процесу.

– Дедуплікація (Deduplication) – спеціальна технологія стиску даних, що виключає дублювання повторюваних копій даних. Дана технологія використовується для збільшення ефективності використання сховища, а також для скорочення кількості байт переданих по мережі.

– «Ощадливий розподіл» даних (Thin Provisioning) – дозволяє виділяти дисковий простір для серверів на лету, а також у тільки необхідній кількості. Thin Provisioning у середовищі з поділюваним сховищем, є методом по оптимальному використанню дискового простору – покладається на динамічне виділення блоків даних на відміну від традиційного виділення всіх блоків відразу.

Сучасний підхід до побудови СЗД – сполучення технологій (SAS/iSCSI/FC)

Сучасні системи дозволяють упоратися зі зростаючим обсягом даних і в той же час знизити витрати. Високопродуктивні масиви з підтримкою технології 8 Гб FC, 1 Гб або 10 Гб Ethernet iSCSI, 6 Гб SAS, забезпечують ефективну консолідацію зберігання за доступною ціною. Дані системи дозволяють спільне використання ресурсів масиву по декількох протоколах відповідно до потреб і бюджетом різних підрозділів і компаній. Завдяки можливості використання різних типів підключення доступ до файлів може здійснюватися як на блоковому, так і на файловому рівні. Можливо одночасне використання дискових накопичувачів SSD і SAS корпоративного рівня й SATA для архівації (підтримуються диски великого й малого типорозмірів). Це рішення поставляється в недорогій конфігурації з одним контролером або в конфігурації із двома контролерами, що забезпечує більше високу доступність і продуктивність для рішення завдань початкового рівня.

Кластерні системи зберігання

На сьогоднішній день системи зберігання для малого й середнього бізнесу з одним/двома контролерами являють собою надійне, але не повністю відказостійке рішення. При відмові одного з контролерів продуктивність системи зберігання значно падає, це не говорячи вже про втрату даних при виході з ладу другого контролера.

Як альтернатива даним системам зберігання можна використовувати кластерні системи зберігання, що використовують технологію iSCSI, забезпечують функціональність масивів корпоративного рівня за доступною ціною.

Завдяки використанню кластерної архітектури, такі рішення володіє поруч унікальних функціональних можливостей. Насамперед, це подвійний захист даних – не тільки на апаратному, але й на мережному рівні. Крім того, при нарощуванні ємності системи зберігання, одночасно збільшуються продуктивність і надійність кластера. А такі технології, як синхронна й асинхронна реплікації, Thin Provisioning, Snapshot і SmartClone включені в кожне рішення без додаткової плати.

На відміну від традиційних дискових масивів, ці системи використовують розподілену кластерну технологію зберігання, що складає з окремих вузлів і єднального їхнього програмного забезпечення. Кожний вузол – це компактний сервер зберігання високої доступності, що містить власні диски, процесор, і пам'ять. Кластерне ПЗ поєднує окремі модулі в єдину систему із загальним пулом дискового простору, розподіляючи навантаження оптимальним образом. Вся доступна ємність і весь потенціал продуктивності вузлів стає доступний кожному логічному той кластерної системи зберігання. Завдяки цьому в міру додавання модулів можна нарощувати продуктивність і ємність без зупинки доступу користувачів до бізнес-додаткам.

Висока доступність забезпечується завдяки технології Network RAID. Кластерні системи зберігання здійснюють синхронну реплікацію, розподіляючи

блоки даних і контрольні суми по різних вузлах кластера. Таким чином, при виході з ладу окремого диска або контролера, або навіть модуля SAN цілком, бізнес-інформація підприємства буде доступна користувачам. Процедури перемикання на резервну копію у випадку збоїти, синхронізація даних і міграція LUN між вузлами кластера здійснюються прозоро для користувачів.

Архітектура

– Кластер з вузлів зберігання.

– Кожний вузол представляє із себе повноцінну систему зберігання.

– Такий масив масштабується від 2-х до більше 16-ти вузлів, керованих як єдина система.

– Логічні томи розташовуються на всіх вузлах.

– З додаванням нових вузлів продуктивність і надійність зростають.

– Автоматичний перерозподіл даних по вузлах.

– Підтримка безлічі площадок і міграція даних у режимі «онлайн».

– Єдина консоль керування.

Подвійна відказостійкість кластерних систем**:**

– Забезпечує максимальну відказостійкість, додатково до апаратного ровню RAID кожного вузла.

– Забезпечує синхронну реплікацію між вузлами кластера й між площадками.

– Задається на рівні окремих томів.

– Забезпечує функціонал високої доступності даних (High Availability).

Системи зберігання з безліччю контролерів

Одна з особливостей архітектури дискових масивів полягає в тому, що кожний логічний тім обслуговується всіма контролерами масиву, які працюють в active-active кластері. Так, в 4-ьох контролерної системі, при падінні одного контролера – втратимо не більше чверті продуктивності, але саме головне, що кеш дзеркалюється між що залишилися 3-мя контролерами. Це принципова відмінність, у порівнянні з будь-якими масивами конкурентів класу midrange, у

яких логічний тім обслуговується парою контролерів. А падіння 1 контролера, приводить до відключення кеш – сильна втрата продуктивності, або, якщо кеш не відключається – ризик втратити дані.

Системи зберігання зі швидким доступом на флеш дисках

Сьогодні процесори, мережі, пам'ять, внутрішні шини стали в рази швидше. За останнє десятиліття швидкість процесорів збільшилася в 8-10 разів, швидкість мереж виросла в 100 разів, а швидкість дисків збільшилася на 20%.

Якщо системи зберігання даних працюють швидко, то й всіх елементах мережної інфраструктури працюють швидше. Тепер флеш-технології й використовується як прискорювач. Флеш-пам'ять надає продуктивність для збільшення швидкості роботи додатків.

Краще застосування флеш-пам'ять знайшла в дата-центрах (FC і InfiniBand) і при необхідності росту продуктивності IT-Інфраструктури підприємства й більших зростаючих обсягів даних, поділюваних системах зберігання, де потрібні гнучка масштабованість, централізоване керування, мобільність і надійний захист даних.

Системи зберігання на флеш-дисках забезпечують високу продуктивність, ефективність, надійність для поділюваних середовищ систем зберігання підприємства, допомагаючи замовникам справлятися із проблемами продуктивності найбільш важливих додатків і інфраструктури по усьому світі. Дані системи зберігання можуть доповнити або ж повністю замінити традиційні масиви жорстких дисків, навіть ті, котрим необхідні SSD диски інші flash технології для використання в додатках. В основному використовуються наступні додатки:

– обробка транзакцій у реальному часі, (OLTP);

– бізнес-аналітика (BI);

– аналітична обробка в реальному часі (OLAP);

– інфраструктура віртуальних десктопів;

– високопродуктивні обчислення;

– також рішення по наданню контенту (такі як хмарні системи зберігання або відео по запиті).

Дані системи зберігання дозволяють досягати продуктивності 500,000 IOPS на читання й пропускної здатності 5 GBps із затримкою менш 100 мілісекунд.

3.2 Розробка структурної схеми

«Шари» являють собою групу фізичних і/або віртуальних ресурсів (обчислювальних, СЗД, мережних), які виділяються й конфігуруються як єдина система для рішення завдань конкретного користувача або користувачів. Таких «шарів» може бути безліч, вони будуть ізольований друг від друга. По суті, мова йде про формування програмно-конфігуруємих, віртуальних інфраструктур під конкретні завдання.

У проекті GENI основні елементи – це стійки GENI, що містять обчислювальні ресурси, СЗД і мережне встаткування. У рамках однієї стійки для формування «шарів» мережі (її віртуалізації) використовується технологія VLAN. Для організації «шарів» між стійками можливе застосування різних технологій, наприклад VLAN або SDN/OpenFlow. До числа недоліків VLAN ставиться, зокрема, те, що вона не забезпечує належного рівня ізоляції в частині продуктивності й програмного контролю. Тому для віртуалізації зв'язків між стійками GENI переважніше OpenFlow/SDN.

Як територіально розподілена мережа для їхньої взаємодії активно використовується експериментальна мережа Internet2, що також заснована на OpenFlow.

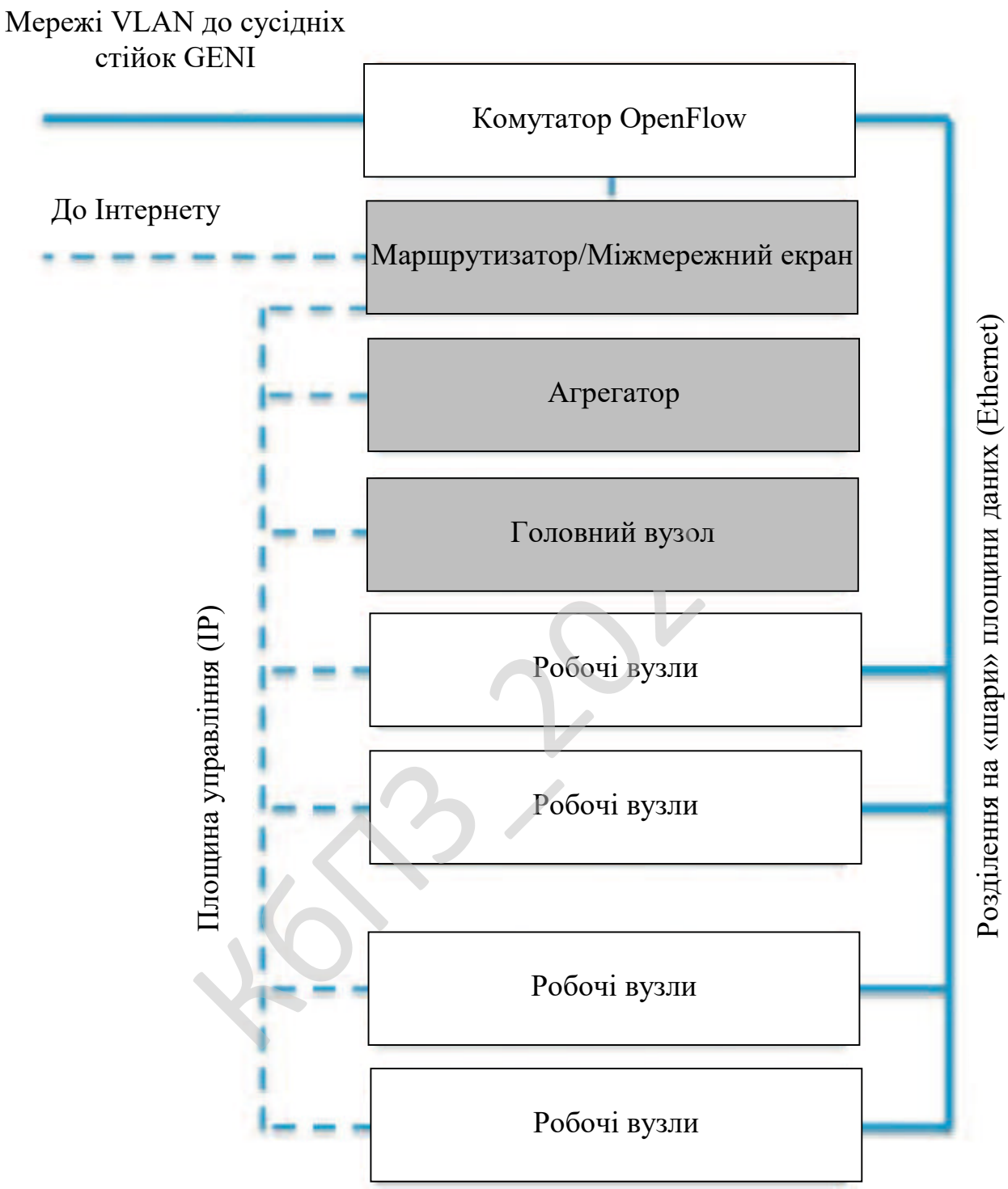

Рисунок 3.1 – Структурна схема системи

GENI дозволяє проводити «масштабні» експерименти на обчислювальному й комунікаційному встаткуванні – наприклад, прогнозувати

поводження складних великомасштабних мереж, тестувати нові мережні архітектури й сервіси. Крім того, завдяки мережі GENI дослідники незалежно від свого місця розташування можуть одержати доступ до обчислювальних ресурсів у рамках всієї мережі.

Важливими новими тенденціями в області майбутніх мереж і розподілених хмар є міжнародна федерація стендів FIDC, а також розробка програмно-конфігуруємих пунктів обміну (Software Defined eXchanges, SDX) і програмно-конфігуруємих інфраструктур (Software Defined Infrastructure, SDI).

Питання створення федерації – скоріше організаційний, а не технологічний. Для його рішення важливо виробити, погодити й прийняти правила використання поділюваних ресурсів, а також створити брокер, якому будуть довіряти всі учасники федерації. Ключовими повинні стати три «А»: Authentication (автентификація учасників), Authorization (авторизація), Accounting (облік споживаних ресурсів).

Технічні компоненти – найпростішому. Що дійсно є викликом [при реалізації міжнародної федерації] – продумати політики «зчеплення» ресурсів і домовитися різним інститутам і організаціям».

Вузли SDX повинні стати одним із ключових елементів для формування федерацій. Їх можна визначити як фізичне або віртуальне «місце зустрічі», де буде відбуватися обмін трафіком між різними мережами SDN з дотриманням політик, прийнятих у кожній такій мережі. У свою чергу, SDI – це збори поділюваних ресурсів, мереж і пунктів обміну SDX, які користувачі можуть задіяти для побудови мультидоменних програмно-конфігуруємих «шарів».

Те, що ми маємо сьогодні, – це окремі острівці SDN. Завдання SDX – об'єднати їх. Розгортання SDN і SDX здатне кардинально змінити архітектуру побудови Інтернету.

Поки що SDX – це скоріше концепція, комерційних рішень не існує, однак уже створений ряд прототипів. Один з таких прототипів розроблений Міжнародним центром iCAIR разом з партнерами й розгорнуть на площадці

StarLight у Чикаго. Ця площадка служить для обміну трафіком на фізичному рівні (оптичні потоки) і рівні L2 між більшим числом національних і міжнародних наукових мереж. Щоб представити масштаб цієї площадки, скажемо, що на неї приходить майже 30 оптичних потоків по 100 Гбіт/с, велика кількість потоків 40G, а також кілька сотень 10-гігабітних потоків.

Створення StarLight SDX стало важливим кроком на шляху реалізації ініціативи iGENI, націленої на побудову глобальної експериментальної мережі з використанням технологій OpenFlow/SDN. Цю мережу планується побудувати на основі глобальної оптичної інфраструктури (Global Lambda Integrated Facility, GLIF), що охоплює в тому числі й Україну. StarLight – один із ключових вузлів GLIF.

StarLight SDX – великий віртуальний комутатор, фізичною основою якого служать кілька комутаторів SDN/OpenFlow. Його головне завдання, як уже говорилося, – забезпечити взаємодія безлічі острівців SDN, але при цьому планується, що він буде також підтримувати мережні сервіси L2 і безпосередньо оптичні сервіси.

Мережна взаємодія між вузлами SDX здійснювалася з використанням трьох окремих мереж.

У проекті планується використовувати SDN-контролер.

Контролер підтримує великий набір мережних сервісів і додатків: маршрутизацію L2/L3 з урахуванням параметрів якості обслуговування (QoS), багатопотокову маршрутизацію, фільтрацію трафіка, трансляцію адрес (NAT), балансування навантаження, віртуалізацію мереж, анти-DDoS, верифікацію мережі, інтеграцію із системами керування ЦОД і ін.

Нові підходи до побудови мереж, які в главу кута ставлять їх програмуємість, відкривають перед українськими компаніями можливість стати помітними гравцями на ринку мережних технологій, на якому до останнього часу Україна виступав лише як споживач чужих розробок і продуктів.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2.

З рисунку видно, що розроблена система складається з наступних функціональних частин:

– Блок інтерфейсу користувача.

– Блок визначення параметрів QoS.

– Блок примусового завдання параметрів QoS.

– Блок моделювання завантаженості мережі системи управління у розподілених Cloud-системах GENI.

– Блок моніторингу мережі системи управління у розподілених Cloud-системах GENI.

– Блок дослідження можливостей механізмів WRED.

– Блок дослідження можливостей механізмів WFQ.

– Блок призначення пріоритетів.

– Блок організації та обслуговування черг.

– Блок управління навантаженням.

– Блок формування трафіка.

Розглянемо ці блоки більш детально.

Існує не занадто багато способів розрахунків показників QoS у системи управління у розподілених Cloud-системах GENI. Найпростіший з них – збільшення смуги пропущення мережі системи управління у розподілених Cloud-системах GENI за рахунок нарощування апаратних можливостей устаткування мережі системи управління у розподілених Cloud-системах GENI. Можна використовувати й такі прийоми, як завдання пріоритетів даних, організація черг, запобігання перевантажень і формування трафіку. Керування мережею за заданими правилами в перспективі повинне об'єднати всі ці способи в єдину автоматизовану систему, що буде гарантувати якість послуг абсолютно на всіх ділянках мережі.

Збільшення апаратної потужності, безсумнівно, є найбільш ефективним засобом реалізації QoS у мережі системи управління у розподілених Cloud-системах GENI. Тиск із боку конкурентів, необхідність підвищення ефективності виробництва, поява нових технологій, що дозволяють оснащувати спеціалізовані мікросхеми (ASIC) найрізноманітнішими функціями, – все це змушує постачальників комутаційного встаткування для локальних мереж викидати на ринок усе більше швидкодіючі пристрої за цінами, порівнянним з вартістю моделей колишнього покоління.

Малоймовірно, що в доступному для огляду майбутньому даний підхід до підтримки QoS у мережах системи управління у розподілених Cloud-системах GENI перестане бути пріоритетним. Оскільки в мережах системи управління у розподілених Cloud-системах GENI вдається забезпечити гарантовану якість послуг, не прибігаючи до дорогої модернізації усього встаткування й серйозних змін у системі керування мережею, мережні адміністратори будуть звертати увагу й на програмні засоби, що дозволяють реалізувати QoS.

Отже, найбільше поширення, швидше за все, одержить комбінований підхід. Деякі виробники висловлюються на його користь, затверджуючи, що найкраще збільшувати пропускну здатність мережі не прямо, а за рахунок інтелектуальних можливостей устаткування, що має засоби розрахунків показників QoS. Правда, виробники мережних пристроїв навряд чи можуть бути об'єктивними в цьому питанні, тому що вони зацікавлені в збуті тих самих продуктів, які підтримують гарантовану якість послуг.

У глобальних мережах нарощування апаратних потужностей використовується рідше. Звичайно, зниження вартості смуги пропущення зробило б передачу даних по глобальних мережах доступною для більше широкого кола користувачів (і навіть трохи знизило б актуальність впровадження гарантованої якості послуг). Але в найближчому майбутньому вартість смуги пропущення в глобальних мережах буде залишатися досить високою, тому й нарощування апаратної потужності не стане настільки популярним, як у мережах системи управління у розподілених Cloud-системах GENI.

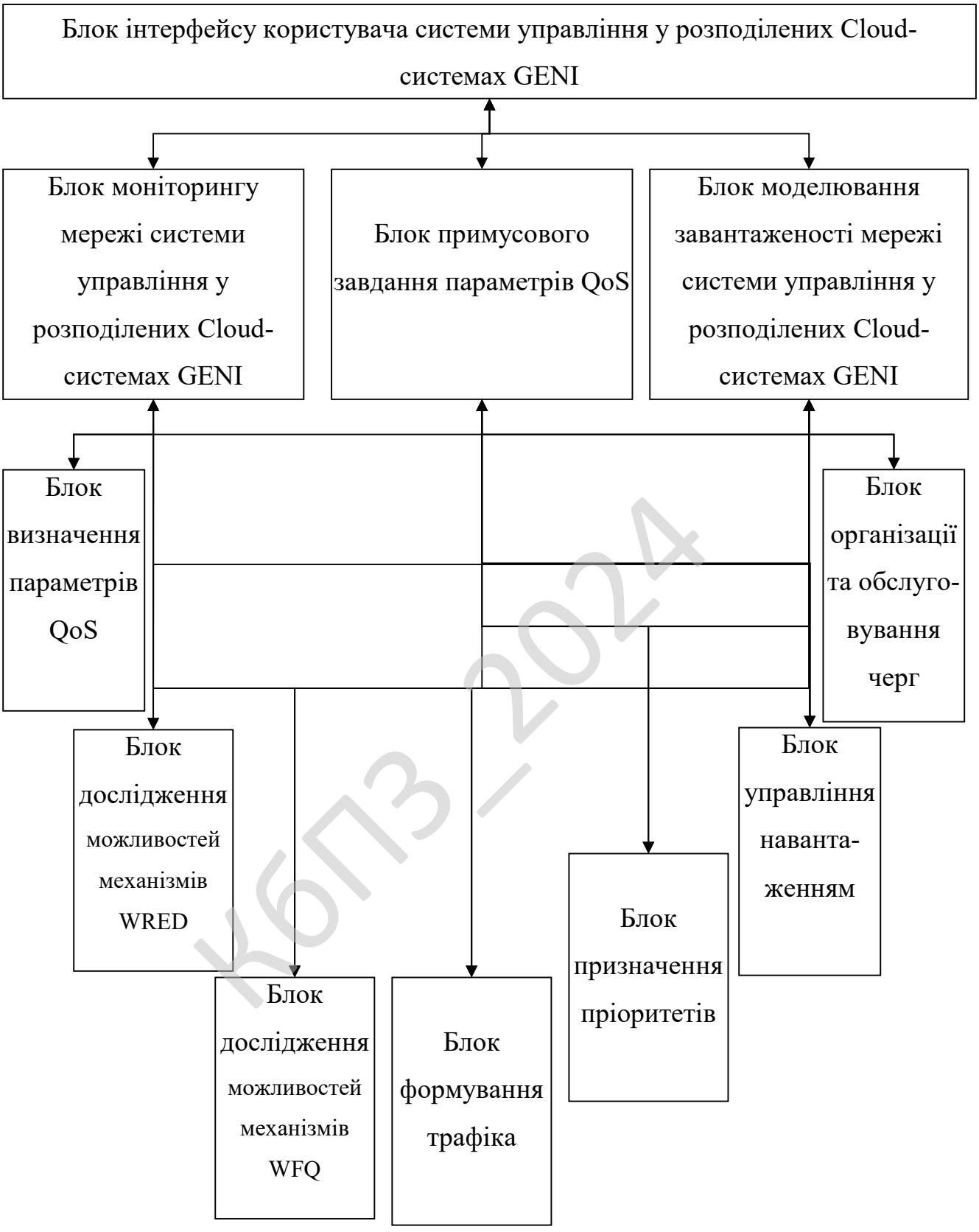

Рисунок 3.2 – Функціональна схема системи

Блок інтерфейсу користувача

Призначений для реалізації взаємодії користувача, або дослідника з системою.

Блок організації та обслуговування черг

Після того як переданим по мережі даним призначені відповідні пріоритети (за допомогою явних або неявних методів), потрібно визначити порядок передачі цих даних, задавши алгоритм обслуговування черг із необхідною якістю (рівнем QoS). По суті, черги являють собою області пам'яті комутатора або маршрутизатора, у яких групуються пакети з однаковими пріоритетами передачі. Алгоритм обслуговування черги визначає порядок, у якому відбувається передача пакетів, що зберігаються в ній. Зміст застосування всіх алгоритмів зводиться до того, щоб забезпечити найкраще обслуговування трафіку з більш високим пріоритетом за умови, що й пакету з низьким пріоритетом гарантується відповідна увага.

При використанні способів завдання явних і неявних пріоритетів алгоритм обробки черг визначає порядок їхнього обслуговування. Відповідно до цього алгоритму на кожні два пакети, переданих у мережу із черги 1 (з високим пріоритетом) доводиться по одному пакету із черг 2 і 3. Пакети з однаковими пріоритетами передаються за принципом FIFO (”першим прийшов – першим вийшов”).

Якщо в мережі виникає перевантаження, служба черг не гарантує своєчасного досягнення пункту призначення найбільш важливими даними. Гарантується лише те, що ці пакети будуть передані раніше, ніж ті, що мають більш низький пріоритет.

Сучасні служби QoS вирішують таке завдання за рахунок резервування смуги пропущення. Кожній із черг (або їхніх груп) виділяється заздалегідь задана величина смуги пропущення, що гарантує певну смугу пропущення для черги з більш високим пріоритетом. Для критичних ситуацій, коли обсяг даних у черзі перевищує розміри смуги пропущення, в алгоритмах обслуговування звичайно

передбачається передача трафіку з високим пріоритетом на смугу пропущення, “приналежну” чергам з низьким пріоритетом, і навпаки. Найпростіші алгоритми обслуговують кожну чергу за принципом FIFO. При цьому передача кадрів великого розміру, що мають високий пріоритет, може приводити до затримок трафіку іншого додатка з настільки ж високим пріоритетом, але меншим обсягом.

У більше складних алгоритмах уживає спроба “справедливої” обробки черг. Наприклад, алгоритм рівномірного пропорційного (або зваженого) обслуговування (WFQ – Weighted Fair Queuing), розроблений компанією Cisco, підрозділяє додатки на потребуючі великої й малої ширини смуги пропущення, а сама смуга пропущення розподіляється між всіма додатками нарівно. Слід зазначити, що основні виробники маршрутизаторів самі розробляють алгоритми обслуговування черг і використовують для їхнього опису власну термінологію.

Істотним недоліком сучасних маршрутизаторів і комутаторів є те, що вони підтримують мале число черг. Найчастіше виробники організують служби QoS, що використовують чотири черги, хоча чим більше черг, тим більше різних пріоритетів можна привласнити переданим пакетам і тим “справедливіше” розподілити смугу пропущення між додатками. Наприклад, адміністратор у стані задати пріоритети таким чином, щоб перевага при передачі віддавалося пакетам, адресованим на більше віддалені вузли.

Блок управління навантаженням

Служба QoS дає можливість використовувати для керування мережею системи управління у розподілених Cloud-системах GENI два важливих механізми – керування в умовах перевантаження й запобігання перевантажень. Перший з них дозволяє кінцевій станції відразу знижувати швидкість передачі даних, коли в мережі починається втрата пакетів. У протоколах TCP/IP і SNA цей механізм підтримується вже протягом декількох років. І хоча сам по собі він не гарантує якості передачі, при його використанні разом з механізмом запобігання перевантажень результати виявляються набагато кращими. У мережах TCP/IP механізм запобігання перевантажень застосовується досить давно, але лише в

останні роки він стає стандартом “де-факто” для маршрутизаторів телекомунікаційних мереж і Internet.

Стандартним способом запобігання перевантажень у мережі стало застосування механізму випадкового виділення пакетів (Random Early Detection, RED). При заповненні черг вище певної критичної оцінки цей механізм змушує маршрутизатор вибирати із черги за випадковим законом деякі пакети й “втрачати” їх. Швидкість передачі даних станціями-відправниками знижується, що й дозволяє уникнути переповнення черги.

Механізм пропорційного випадкового виділення пакетів – WRED (Weighted RED) – можна вважати наступною, більше зробленою “версією” RED. Він передбачає, що вибір пакетів, які повинні “втратитися”, буде відбуватися з обліком їх пріоритезації згідно IP TOS.

Блок формування трафіка

Формування трафіку – це загальний термін, яким прийнято позначати різні способи маніпулювання даними для підвищення якості їхньої передачі. Один їхній таких способів – сегментація пакетів. У мережах системи управління у розподілених Cloud-системах GENI гарантовано високий рівень QoS досягається в тому числі й за рахунок малого розміру переданих пакетів (осередків – у термінології системи управління у розподілених Cloud-системах GENI). Максимальний час затримки при передачі будь-якого пакета мережі системи управління у розподілених Cloud-системах GENI – це час передачі одного осередку.

Запозичаючи корисні механізми технології системи управління у розподілених Cloud-системах GENI, виробники маршрутизаторів і комутаторів починають забезпечувати у своїх продуктах можливість сегментації пакетів. Деякі пристрої, призначені для мереж frame relay, сегментують пакети, передані по каналах глобальних мереж, щоб гарантувати конкретний час передачі й мінімізувати затримки.

Ще один спосіб формування трафіку – його “вирівнювання”. Для таких протоколів, як наприклад, AppleTalk, характерна нерівномірна передача пакетів,

що часом приводить до появи в мережі послідовностей або ланцюжків пакетів, а отже – до її перевантаження. Процедура вирівнювання трафіку дозволяє розчленувати ланцюжки шляхом розміщення пакетів у буфері перед їхньою передачею в мережу. Для забезпечення більше рівномірної передачі даних можна також вирівнювати трафік кінцевих вузлів мережі.

Блок визначення параметрів QoS

Призначений для визначення існуючих параметрів якості обслуговування (QoS). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок примусового завдання параметрів QoS

Призначений для примусового завдання одного, або декількох параметрів якості обслуговування (QoS). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок моделювання завантаженості мережі системи управління у розподілених Cloud-системах GENI

Призначений для моделювання завантаженості мережі системи управління у розподілених Cloud-системах GENI з визначеним трафіком та заданими параметрами якості обслуговування (QoS).

Блок моніторингу мережі системи управління у розподілених Cloud-системах GENI

Призначений для аналізу поточного стану мережі системи управління у розподілених Cloud-системах GENI.

Блок дослідження можливостей механізмів WRED

Одним з методів QoS, призначених для забезпечення необхідних вимог до різних потоків даних – запобігання перевантажень (congestion avoidance). Він заснований на обмеженні розмір черги, сигналізуючи джерелам даних про необхідність зменшити швидкість передачі інформації (WRED – Weighted random early detection).

Блок дослідження можливостей механізмів WFQ

Другим з методів QoS, призначених для забезпечення необхідних вимог до різних потоків даних – керування перевантаженням (congestion management). Він заснований на присвоєнні квот і пріоритетів потокам, і у випадку перевантаження, потоки одержують якість, обмежену їхньою квотою й пріоритетом (WFQ – Weighted Fair Queuing).

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

Блок призначення пріоритетів

Нарівні з нарощуванням апаратного забезпечення мережі для реалізації QoS застосовуються й засобу типу завдання пріоритетів даних і організації черг. Маршрутизатори підтримують ці механізми протягом багатьох років, як і деякі з нових комутаторів для каналів Gigabit Ethernet. Однак ПЗ для керування мережею за заданими правилами, яких необхідно для практичного втілення цієї технології, поки не розроблено. Серед нових комутаторів такого класу можна назвати CoreBuilder 3500, CoreBuilder 9000 і SuperStack II компанії 3Com, пристрою серії Accelar фірми Bay Networks, SmartSwitch Router компанії Cabletron Systems, а також Catalyst 5000 і Catalyst 8000 виробництва Cisco.

Способи пріоритезації даних можна умовно підрозділити на явні й неявні.

При неявному призначенні пріоритетів маршрутизатор або комутатор автоматично привласнює послугам відповідні рівні, виходячи із заданих адміністратором мережі критеріїв (наприклад, типу додатка для застосовуваного протоколу передачі або адреси джерела). Кожний вхідний пакет аналізується (фільтрується) на відповідність цим критеріям. Механізм неявної пріоритезації підтримують практично всі маршрутизатори.

Деякі комутатори теж здатні задавати пріоритети, але мають обмежений набір функцій. Так, комутатори можуть забезпечувати пріоритезацію даних по типу віртуальної локальної мережі, адресі джерела або адресата, але не використовують інформацію більш високого рівня (протокол передачі або тип додатка). Розроблювальні в цей час системи керування мережею за заданими правилами дозволять реалізувати більше зроблені схеми пріоритезації даних при роботі з такими комутаторами.

При явній пріоритезації даних користувач або додаток запитує певний рівень служби, а комутатор або маршрутизатор намагається задовольнити запит. Імовірно, самим популярним механізмом явної пріоритезації стане протокол IP Precedence (протокол старшинства), що одержав другу назву IP TOS (IP Type Of Service), – один з розділів четвертої версії протоколу IP.

IP TOS резервує спеціальне поле в заголовку пакета, де можуть бути зазначені ознаки QoS, що визначають час затримки, швидкість пропущення й рівень надійності передачі пакета. Однак найдеться небагато популярних додатків – за винятком мультимедійного ПЗ, – у які реалізована підтримка протоколу IP TOS.

Зараз розробляється протокол резервування ресурсів RSVP, що передбачає більше складний, чим в IP TOS, механізм передачі від додатка до машрутизатору запиту на гарантовану якість послуг. Як і IP TOS, протокол RSVP поки не одержав широкої підтримки розроблювачів – він реалізований лише в окремих типах маршрутизаторів. Поширення RSVP стримується через те, що не вирішені деякі питання, пов'язані із сумісністю різних мереж. До того ж застосування

RSVP значно збільшує навантаження на маршрутизатори й може привести до зниження швидкодії цих пристроїв.

Видимо, у доступному для огляду майбутньому неявна пріоритезація, не потребуюча серйозних обчислювальних потужностей маршрутизатора, залишиться більше популярною, чим явна. Крім того, при явному завданні пріоритетів значно ускладнюється керування мережею. Кінцеві користувачі, швидше за все, будуть набудовувати своє програмне забезпечення на запит найвищого з можливих рівнів послуг. Відповідно, адміністраторові мережі прийдеться розробляти правила керування користувачами й, можливо, навіть побудовувати служби з гарантованою якістю для кожного користувача окремо.

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.

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.

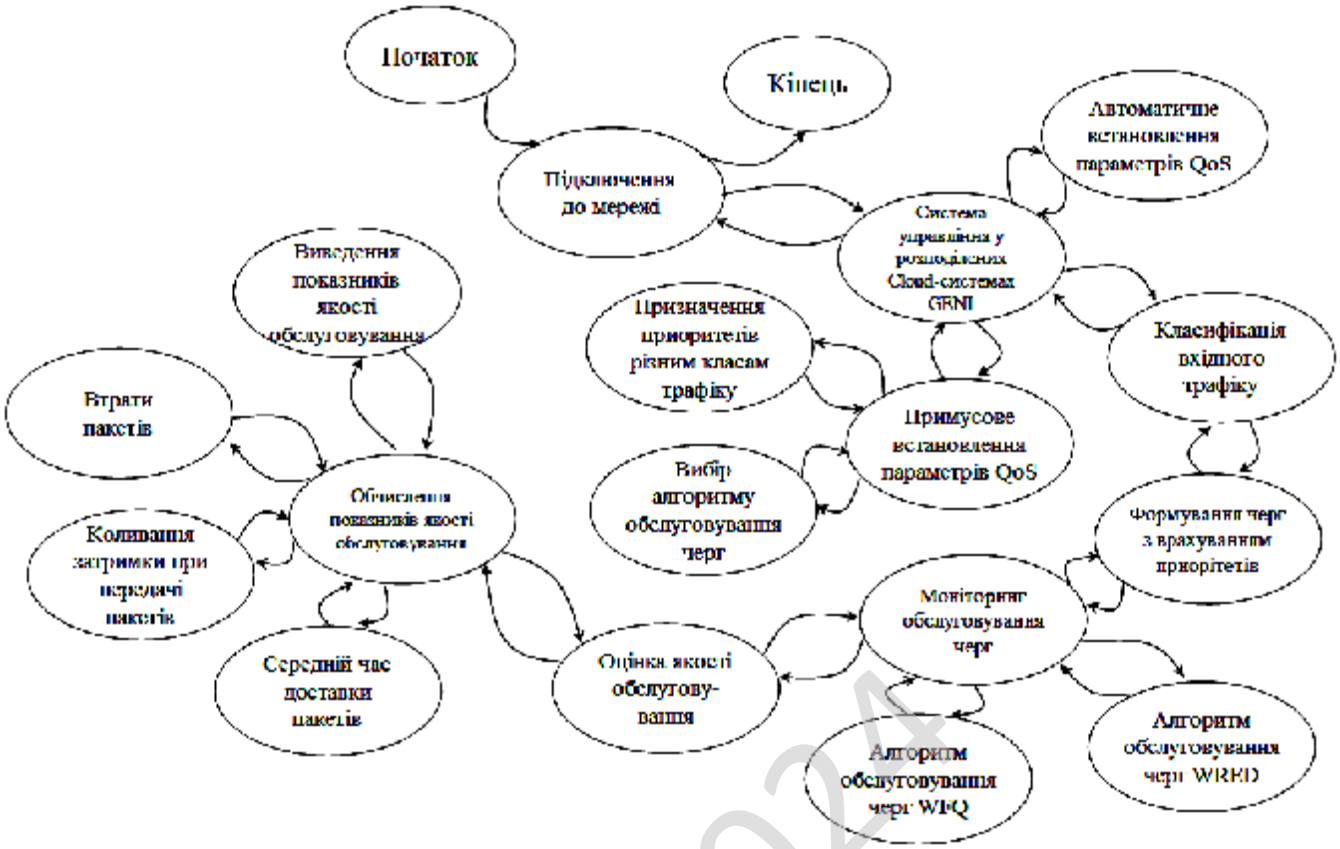

Рисунок 3.3 – Діаграма взаємодії процесів

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

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

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

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 системи управління у розподілених Cloud-системах GENI.

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.

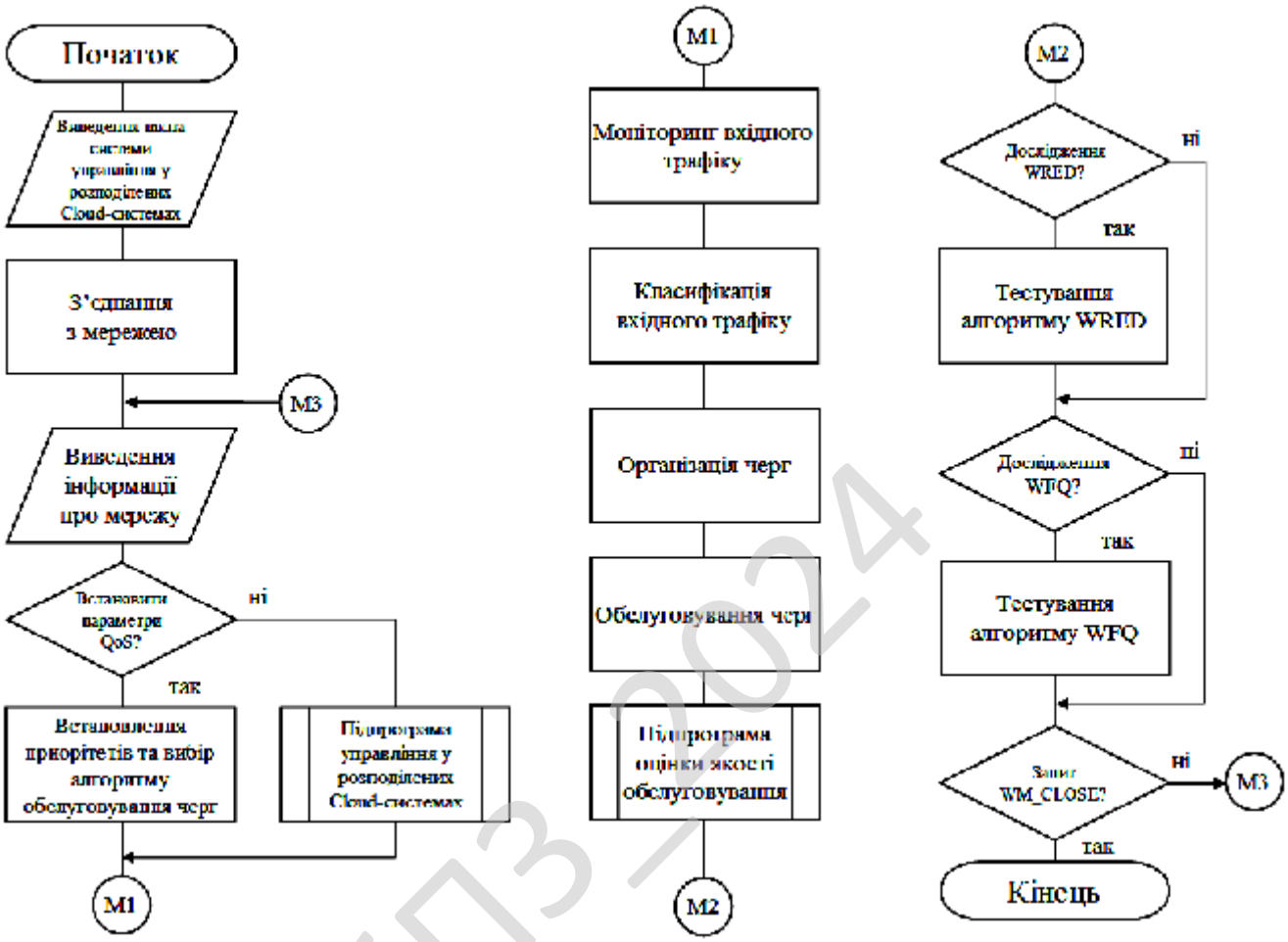

Рисунок 4.1 – Блок-схема основної програми

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.

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

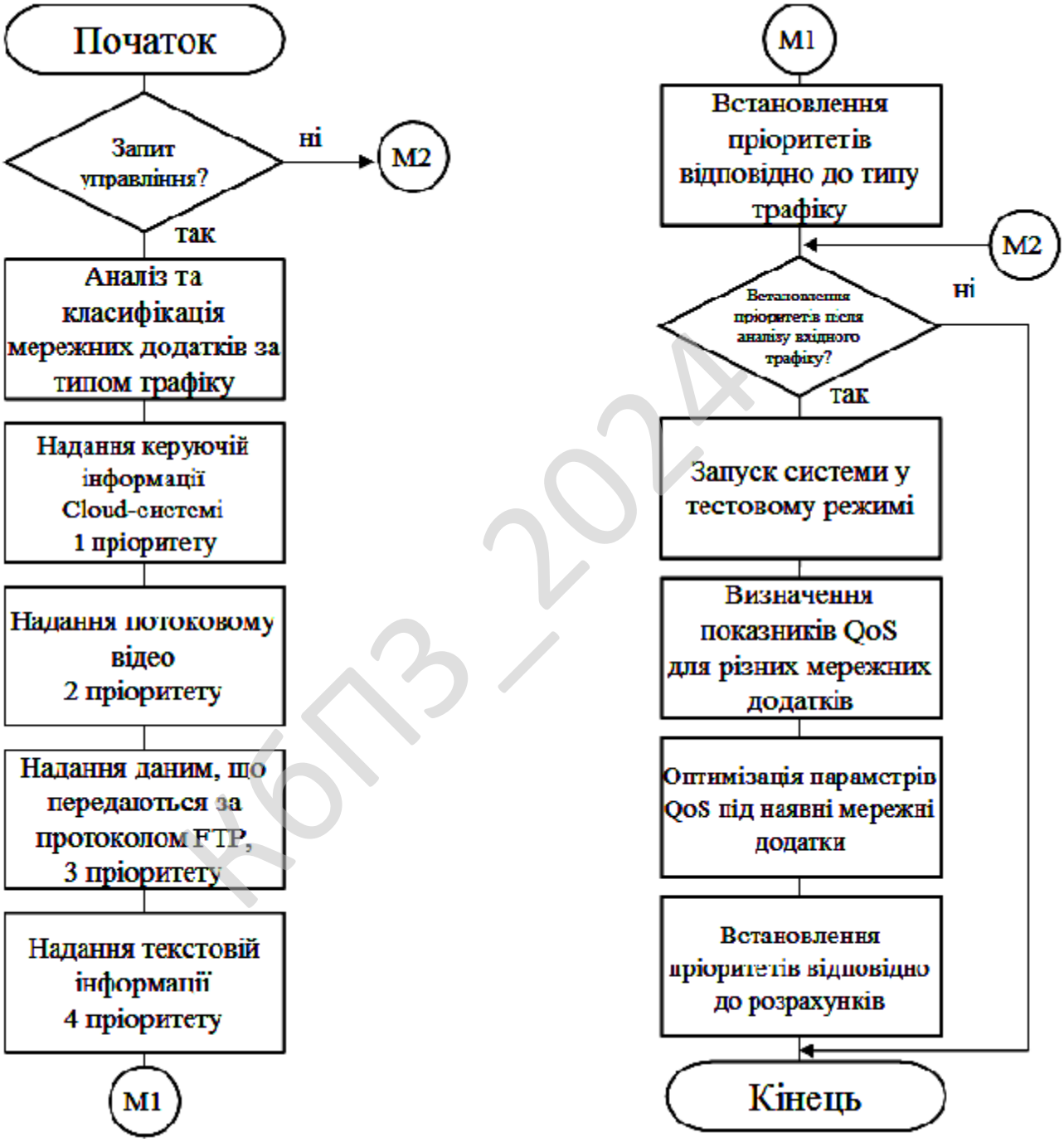

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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.

Отримання робочих параметрів мережі для даної підпрограми відбувається наступним чином:

```
procedure Get_Net_ workParams( List: TStrings );
var
    Net__workParams: TNet__workParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNet__workParams (Net__workParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with Net__workParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NET__BIOS тип : '  + NET__BIOSTypes[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 IpHlpNet__workParams (var Net__workParams: TNet__workParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
```

```
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNet__workParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNet__workParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    Net__workParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        Net__workParams.HostName := trim (HostName) ;
        Net__workParams.DomainName := trim (DomainName) ;
        Net__workParams.ScopeId := trim (ScopeID) ;
        Net__workParams.NodeType := NodeType ;
        Net__workParams.EnableRouting := EnableRouting ;
        Net__workParams.EnableProxy := EnableProxy ;
        Net__workParams.EnableDNS := EnableDNS ;
        Net__workParams.DnsServerNames [0] := DNSServerList.IPAddress ;  // дані
        if Net__workParams.DnsServerNames [0] <> ' '  then
                                    Net__workParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            Net__workParams.DnsServerNames [Net__workParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (Net__workParams.DnsServerTot) ;
            if Net__workParams.DnsServerTot >=
                        Length (Net__workParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                    // дані
    end ;

end;
```

Дані з таблиці маршрутизації досліджуємої мережі у розподілених Cloud-системах GENI отримуються за допомогою наступних підпрограм:

```
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 ;   // дані
          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 ) );
// дані
      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 ;      // дані
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 ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;
//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
    ICMPIn.Add( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
    ICMPIn.Add( ' Помилка               : '  + IntToStr( dwErrors ) );
    ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs ) );
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
```

```
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
end;

    with ICMPStats.OutStats do
begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs));
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs ) );
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( '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;
```

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний

ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P-1)(Q-1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\text{mod } \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m (\text{mod } N)$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D. Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E (\text{mod } N)$. Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): $m = h(M)$. Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти модуль N на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису S буде позитивним тільки в тому випадку, якщо при обчисленні S був використаний секретний ключ D, що відповідає відкритому ключу E. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

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 системи управління у розподілених Cloud-системах GENI складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Моніторинг; Тестування; Параметри; Довідка.

– Підрозділ представлення параметрів вибору та обчислення.

– Вікна виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

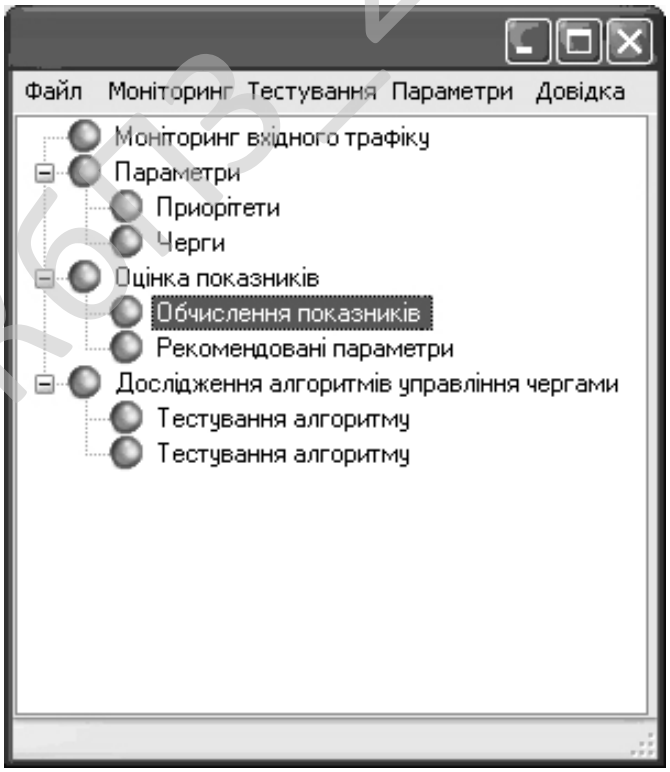

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

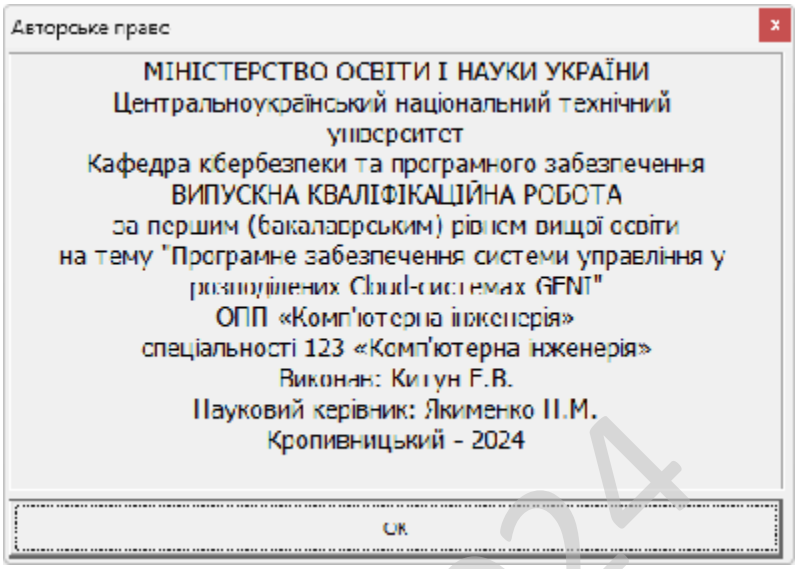

Рисунок 5.2 – Вікно розробника ПЗ

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 – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

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

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

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи управління у розподілених Cloud-системах GENI.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем управління у розподілених Cloud-системах GENI.

– Досліджена система управління у розподілених Cloud-системах GENI.

– На основі отриманих результатів досліджень створена програмна реалізація системи управління у розподілених Cloud-системах GENI.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання управління у розподілених Cloud-системах GENI.

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

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

Програма реалізована на мові високого рівня Delphi 10.4.1. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи управління у розподілених Cloud-системах GENI. 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 RSA.

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. 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.

2. 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.

3. 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.

4. 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.

5. 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.

6. 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.

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

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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).

38. 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

39. 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.

40. 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.

41. 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.

42. 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.

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

44. 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.

45. 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.

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

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

48. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Доренський О.П., Дрєєв О.М., Вялкова В.І. Комп’ютерні мережі. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 233 с.

50. Смірнов О.А., Мелешко Є.В., Семенов С.Г. Методи та засоби обробки сигналів і даних в інформаційних системах. Навчальний посібник. – Кіровоград: КНТУ 2012. – 250 с.

51. Смірнов О.А., Євсеєв С.П., Жукарев В.Ю., Король О.Г., Сорокін В.Є., Мелешко Є.В. Технології і стандарти комп’ютерних мереж. Навчальний посібник. – Кіровоград: КНТУ 2012. – 454 с

Додаток А
(обов’язковий)

Технічне завдання

Зміст

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

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи управління у розподілених Cloud-системах GENI.

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. № 133-02 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 управління у розподілених Cloud-системах GENI.

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 управління у розподілених Cloud-системах GENI;

– 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 10.4.1.

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 – 70 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 управління у розподілених Cloud-системах GENI

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2024 року

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;
    Net_Cloud_GENI_Tree: 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 Net_Cloud_GENI_TreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure Net_Cloud_GENI_TreeDblClick(Sender: TObject);
    procedure Net_Cloud_GENI_TreeGetImageIndex(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
Net_Cloud_GENI_ContainerToOpen: PNet_Cloud_GENI_Resource);
  //  function OpenEnum(const Net_Cloud_GENI_ContainerToOpen:
PNet_Cloud_GENI_Resource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNet_Cloud_GENI_Enum: THandle):
UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_net_Cloud_GENI_name : 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_net_Cloud_GENI_name : 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
Net_Cloud_GENI_ShareEnumNT:function (    servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

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

var
Net_Cloud_GENI_ShareDelNT:function (servername: PWideChar;
                        net_Cloud_GENI_name: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
Net_Cloud_GENI_ShareDel:function (  pszServer,
                        pszNet_Cloud_GENI_Name:PChar;
                        usReserved:Word): DWORD; stdcall;

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

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

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

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

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

var
Net_Cloud_GENI_SessionDel:function( pszServer:PChar;
```

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

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

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

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

var
Net_Cloud_GENI_FileClose2: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 тьа вище -
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 досліджуємої мережі
//

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(' NET_Cloud_GENI_API32.DLL' ); //Завантажуємо
бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @Net_Cloud_GENI_ShareEnumNT := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareEnum' );
    if not Assigned(Net_Cloud_GENI_ShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if Net_Cloud_GENI_ShareEnumNT(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_net_Cloud_GENI_name));
  end else begin //Код для 9 x-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @Net_Cloud_GENI_ShareEnum := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareEnum' );
    if not Assigned(Net_Cloud_GENI_ShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if Net_Cloud_GENI_ShareEnum(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_net_Cloud_GENI_name));
  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(' NET_Cloud_GENI_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_ShareDelNT := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareDel' );
    if not Assigned(Net_Cloud_GENI_ShareDelNT) 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
    Net_Cloud_GENI_ShareDelNT(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;
    @Net_Cloud_GENI_ShareDel := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareDel' );
    if not Assigned(Net_Cloud_GENI_ShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    Net_Cloud_GENI_ShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Показ діалогу вибору директорії
//

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
```

```
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
end;

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

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

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

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NET_Cloud_GENI_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_ShareAddNT := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareAdd' );
    if not Assigned(Net_Cloud_GENI_ShareAddNT) 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_net_Cloud_GENI_name := 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); // Кількість максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кількість тік підкл.

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

    ShareNT.shi2_passwd := ' ' ; //Пароль

    Net_Cloud_GENI_ShareAddNT(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;
    @Net_Cloud_GENI_ShareAdd := GetProcAddress(FLibHandle,'
Net_Cloud_GENI_ShareAdd' );
    if not Assigned(Net_Cloud_GENI_ShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_net_Cloud_GENI_name,
SizeOf(Share9x.shi50_net_Cloud_GENI_name), #0);
    move(TmpName[1],Share9x.shi50_net_Cloud_GENI_name[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
    Net_Cloud_GENI_ShareAdd(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 кількість 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 досліджуємої системи управління у розподілених Cloud-
системах GENI
//

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(‘ NET_Cloud_GENI_API32.DLL’ );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_SessionEnumNT := GetProcAddress(FLibHandle, ‘
Net_Cloud_GENI_SessionEnum’ );
    if not Assigned(Net_Cloud_GENI_SessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if Net_Cloud_GENI_SessionEnumNT(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;
   @Net_Cloud_GENI_SessionEnum := GetProcAddress(FLibHandle, ‘
Net_Cloud_GENI_SessionEnum’ );
    if not Assigned(Net_Cloud_GENI_SessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if Net_Cloud_GENI_SessionEnum
(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
досліджуємої системи управління у розподілених Cloud-системах GENI
        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(' NET_Cloud_GENI_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_SessionDelNT := GetProcAddress(FLibHandle, '
Net_Cloud_GENI_SessionDel' );
    if not Assigned(Net_Cloud_GENI_SessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    Net_Cloud_GENI_SessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_SessionDel := GetProcAddress(FLibHandle, '
Net_Cloud_GENI_SessionDel' );
    if not Assigned(Net_Cloud_GENI_SessionDel) 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
    Net_Cloud_GENI_SessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Одержання списку відкритих файлів досліджуємої системи управління у
розподілених Cloud-системах GENI
//

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(‘ NET_Cloud_GENI_API32.DLL’ );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_FileEnumNT := GetProcAddress(FLibHandle, ‘
Net_Cloud_GENI_FileEnum’ );
    if not Assigned(Net_Cloud_GENI_FileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FileInfoNT := nil;
    if Net_Cloud_GENI_FileEnumNT(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;
   @Net_Cloud_GENI_FileEnum := GetProcAddress(FLibHandle, ‘
Net_Cloud_GENI_FileEnum’ );
    if not Assigned(Net_Cloud_GENI_FileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if Net_Cloud_GENI_FileEnum (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(' NET_Cloud_GENI_API32.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_FileClose := GetProcAddress(FLibHandle, '
Net_Cloud_GENI_FileClose' );
    if not Assigned(Net_Cloud_GENI_FileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;

Net_Cloud_GENI_FileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриває
мо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @Net_Cloud_GENI_FileClose2 := GetProcAddress(FLibHandle, '
Net_Cloud_GENI_FileClose2' );
    if not Assigned(Net_Cloud_GENI_FileClose2) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    Net_Cloud_GENI_FileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік досліджуємої системи управління у
розподілених Cloud-системах GENI
//

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 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 з досліджуємої системи управління у розподілених Cloud-системах GENI
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу для оцінки якості обслуговування (QoS)
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;


 function OpenEnum(const Net_Cloud_GENI_ContainerToOpen:
PNet_Cloud_GENI_Resource; ResScope, ResType, ResUsage: DWORD): THandle;
var
 hNet_Cloud_GENI_Enum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNet_Cloud_GENI_OpenEnum(ResScope, ResType, ResUsage,
                                  Net_Cloud_GENI_ContainerToOpen,
hNet_Cloud_GENI_Enum))
 then ShowMessage(' Помилка!' )
 else Result:=hNet_Cloud_GENI_Enum;
end;

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

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNet_Cloud_GENI_Resource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNet_Cloud_GENI_EnumResource(hNet_Cloud_GENI_Enum,
EntriesToGet,
                               @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
```

```
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNet_Cloud_GENI_EnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані 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 Net_Cloud_GENI_ContainerToOpen:
PNet_Cloud_GENI_Resource);
var
 hNet_Cloud_GENI_Enum: THandle;
begin
 hNet_Cloud_GENI_Enum:=OpenEnum(Net_Cloud_GENI_ContainerToOpen, ResScope,
ResType, ResUsage);
 if (hNet_Cloud_GENI_Enum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNet_Cloud_GENI_Enum);
 if (NO_ERROR<>WNet_Cloud_GENI_CloseEnum(hNet_Cloud_GENI_Enum))
 then ShowMessage(' WNet_Cloud_GENI_CloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 Net_Cloud_GENI_Tree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET_Cloud_GENI_;
  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(Net_Cloud_GENI_Tree.Items.Add(nil, ' Net_Cloud_GENI_work
Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;
```

```
end;

procedure TMainForm.Net_Cloud_GENI_TreeCustomDrawItem(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.Net_Cloud_GENI_TreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(Net_Cloud_GENI_Tree.Selected.Text),' ` ,' `
,SW_SHOW);
end;

procedure TMainForm.Net_Cloud_GENI_TreeGetImageIndex(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.
```

Основна програма

Файл Cloud_GENI.dpr основної програми

```
program Cloud_GENI;

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 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 ( NET_Cloud_GENI_BIOS)
  BROADCAST_NODETYPE = 1;
  PEER_TO_PEER_NODETYPE = 2;
  MIXED_NODETYPE = 4;
  HYBRID_NODETYPE = 8;

  NET_Cloud_GENI_BIOSTypes  : 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_Cloud_GENI__ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET_Cloud_GENI_            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_Cloud_GENI__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_Cloud_GENI_' , ' tokenring' , ' FDDI' , '
PPP' , ' loopback' , ' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet_Cloud_GENI_' , ' ' , '
' , ' 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;    // дані
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу досліджуємої системи управління у
розподілених Cloud-системах GENI-----------------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED
//
      // може передавати деякі дані. 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
      // дані 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_Cloud_GENI_ = 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;       //
дані
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// дані
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
          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 ;   // дані- простір для списку IP
адрес досліджуємої системи управління у розподілених Cloud-системах GENI
        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 ----------------------

 //
  PTMibIPNet_Cloud_GENI_Row = ^TMibIPNet_Cloud_GENI_Row;
  TMibIPNet_Cloud_GENI_Row = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNet_Cloud_GENI_Table = ^TMibIPNet_Cloud_GENI_Table;
  TMibIPNet_Cloud_GENI_Table = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNet_Cloud_GENI_Row;
  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;

    GetNet_Cloud_GENI_workParams: 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;

    GetIpNet_Cloud_GENI_Table: function ( pIpNet_Cloud_GENI_Table:
PTMibIPNet_Cloud_GENI_Table;
      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 помилки 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' ) ;
          GetNet_Cloud_GENI_workParams := GetProcAddress (IpHlpModule, '
GetNet_Cloud_GENI_workParams' ) ;
          GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
          GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
          GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
          GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
          GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
          GetIpNet_Cloud_GENI_Table := GetProcAddress (IpHlpModule, '
GetIpNet_Cloud_GENI_Table' ) ;
          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- дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),     { File Transfer Protocol- управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET_Cloud_GENI_ '  ),
    ( 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 Net_Cloud_GENI_work News
Transfer Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),     { Протокол Net_Cloud_GENI_work Time
protocol          }
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Протокол Location Service
}
    ( Prt: 137; Srv: ' NBNAME '  ),     { NET_Cloud_GENI_BIOS сервіс імен
}
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NET_Cloud_GENI_BIOS сервіс датаграм
}
    ( Prt: 139; Srv: ' NBSESS '  ),     { NET_Cloud_GENI_BIOS сервіс сесій
}
    ( Prt: 143; Srv: ' IMAP   '  ),     { Протокол Internet_Cloud_GENI_ Message
Access Protocol }
```

```
    ( Prt: 161; Srv: ' SNMP    '  ),      { Протокол Simple Net_Cloud_GENI_w.
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_Cloud_GENI__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' , ' NET_Cloud_GENI_MGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNet_Cloud_GENI_workParams
  TNet_Cloud_GENI_workParams = 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 дані--------------------

function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNet_Cloud_GENI_workParams (var Net_Cloud_GENI_workParams:
TNet_Cloud_GENI_workParams): integer ;
procedure Get_Net_Cloud_GENI_workParams( 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 досліджуємої мережі для оцінки
якості обслуговування (Cloud_GENI)}

procedure Get_Net_Cloud_GENI_workParams( List: TStrings );
var
    Net_Cloud_GENI_workParams: TNet_Cloud_GENI_workParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNet_Cloud_GENI_workParams (Net_Cloud_GENI_workParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with Net_Cloud_GENI_workParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NET_Cloud_GENI_BIOS тип : '  +
NET_Cloud_GENI_BIOSTypes[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 IpHlpNet_Cloud_GENI_workParams (var Net_Cloud_GENI_workParams:
TNet_Cloud_GENI_workParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
begin
    InfoSize := 0 ;   // дані
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNet_Cloud_GENI_workParams( Nil, @InfoSize );  // дані
    if result <> ERROR_BUFFER_OVERFLOW then exit ; // дані
    GetMem (FixedInfo, InfoSize) ;                    // дані
    try
    result := GetNet_Cloud_GENI_workParams( FixedInfo, @InfoSize );   // дані
    if result <> ERROR_SUCCESS then exit ;
    Net_Cloud_GENI_workParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        Net_Cloud_GENI_workParams.HostName := trim (HostName) ;
        Net_Cloud_GENI_workParams.DomainName := trim (DomainName) ;
        Net_Cloud_GENI_workParams.ScopeId := trim (ScopeID) ;
        Net_Cloud_GENI_workParams.NodeType := NodeType ;
        Net_Cloud_GENI_workParams.EnableRouting := EnableRouting ;
        Net_Cloud_GENI_workParams.EnableProxy := EnableProxy ;
        Net_Cloud_GENI_workParams.EnableDNS := EnableDNS ;
        Net_Cloud_GENI_workParams.DnsServerNames [0] := DNSServerList.IPAddress
;  // дані
        if Net_Cloud_GENI_workParams.DnsServerNames [0] <> ' '  then
                                    Net_Cloud_GENI_workParams.DnsServerTot
:= 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            Net_Cloud_GENI_workParams.DnsServerNames
[Net_Cloud_GENI_workParams.DnsServerTot] :=
                                    PDnsServer^.IPAddress ;  // дані
            inc (Net_Cloud_GENI_workParams.DnsServerTot) ;
            if Net_Cloud_GENI_workParams.DnsServerTot >=
                          Length (Net_Cloud_GENI_workParams.DnsServerNames)
then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                      // дані
    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 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам’ яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  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] )  // дані, 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 ;

//-----------------------------------
{  моніторимо  час  доступу  до  IP  досліджуємої  мережі  для  оцінки  якості
обслуговування (Cloud_GENI)}
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
  IPNet_Cloud_GENI_Row      : TMibIPNet_Cloud_GENI_Row;
  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 := GetIPNet_Cloud_GENI_Table( Nil, @TableSize, false );   // дані
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNet_Cloud_GENI_Table( PTMIBIPNet_Cloud_GENI_Table( pBuf ),
@TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNet_Cloud_GENI_Table( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNet_Cloud_GENI_Row := PTMIBIPNet_Cloud_GENI_Row( PBuf )^;
        with IPNet_Cloud_GENI_Row do
          List.Add( Format( ' %8x | %12s | %16s | %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNet_Cloud_GENI_Row ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // необхідно відновити показник!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNet_Cloud_GENI_Row ) );
      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 );  // дані
  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 );  // дані
  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 досліджуємої мережі для оцінки якості
обслуговування (Cloud_GENI); }
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 ;   // дані
          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 ) );
// дані
      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 ;      // дані
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 ;     // дані
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;

//-----------------------------------
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;
begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( '  Прийнято повідомлень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка               : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPIn.Add( ' Запит маски адрес : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats.OutStats do
    begin
      ICMPOut.Add( ' Повідомлення вдправлено        : '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка               : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягенно    : '  + IntToStr( dwDestUnreachs
) );
```

```
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключено        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Переназначено            : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит        : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо відповідь         : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Відповідь мітки часу    : '  + IntToStr( dwTimeStampReps )
);
      ICMPOut.Add( 'Запит маски адрес: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Відповідь маски адрес   : '  + IntToStr( dwAddrReps ) );
    end;
  end
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- монітор TCP/IP з'єднань досліджуємої мережі У розподілених Cloud-системах GENI (Cloud_GENI)

```
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 У розподілених Cloud-системах GENI (Cloud_GENI)

```
unit Stat;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, IPHelper, IpHlpApi, Buttons;

type
  TForm3 = class(TForm)
    StaticText7: TStaticText;
    TCPStatMemo: TMemo;
    StaticText5: TStaticText;
    IPStatsMemo: TMemo;
    StaticText12: TStaticText;
    ICMPInMemo: TMemo;
    ICMPOutMemo: TMemo;
    StaticText4: TStaticText;
    UDPStatsMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

  Get_TCPStatistics( TCPStatMemo.Lines );
  Get_IPStatistics( IPStatsMemo.Lines );
  Get_UDPStatistics( UDPStatsMemo.Lines );
  Get_ICMPStats( ICMPInMemo.Lines, ICMPOutMemo.Lines );

end;

procedure TForm3.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm3.btRTTIClick(Sender: TObject);
var
  IPadr         : dword;
  Rtt, HopCount : longint;
  Res           : integer;
```

```
begin
  btRTTI.Enabled := false;
  Screen.Cursor := crHOURGLASS;
  IPadr := Str2IPAddr( edtRTTI.Text );
  Res := Get_RTTAndHopCount( IPadr, 128, RTT, HopCount );
  if Res = NO_ERROR then
    ShowMessage( ' Час запиту '
      + inttostr( rtt ) + ' ms, '
      + inttostr( HopCount )
      + ' hops to : ' + edtRTTI.Text
      )
  else
    ShowMessage( ' Помилка:' + #13
                 + ICMPErr2Str( Res ) ) ;
  btRTTI.Enabled := true;
  Screen.Cursor := crDEFAULT;

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end
  else
      ShowMessage( 'Інтернет помічник DLL не є доступним, або не підтримується'
) ;
end;

end.
```

Файл About.pas - довідка

```
unit About;

interface

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

type
  TForm1 = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Button1: TButton;
    Image2: TImage;
    Image1: TImage;
    Image3: TImage;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form1.Close;
end;

end.
```