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 віртуалізацій мережних функцій NFV”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-3СК
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Яковенко А.Е.
«____» _____ 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

Яковенку Артему Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи віртуалізацій мережних функцій NFV*

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

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

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

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи віртуалізацій мережних функцій NFV*

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
_____ Яковенко А.Е.
(прізвище та ініціали)

АНОТАЦІЯ

Яковенко А.Е. Програмне забезпечення системи віртуалізацій мережних функцій NFV. 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 віртуалізацій мережних функцій NFV.

Метою розробки є програмне забезпечення системи віртуалізацій мережних функцій NFV.

Результат роботи – програмна реалізація системи віртуалізацій мережних функцій NFV.

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 Sydney.

Ключові слова: комп'ютерна інженерія, віртуалізація, NFV

ABSTRACT

Yakovenko A.E. NFV network function virtualization system software. 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 virtualization system of NFV network functions.

The goal of the development is the software of the virtualization system of NFV network functions.

The result of the work is the software implementation of the virtualization system of NFV network functions.

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 Sydney environment.

Keywords: computer engineering, virtualization, NFV

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. Багато хто вважає, що традиційні монолітні мережні системи й архітектури незабаром зникнуть. Програмно конфігуруємі мережі, мобільні технології й Big Data настільки стрімко міняють наше подання про архітектуру й адміністрування корпоративних мереж, що через п'ять років останні будуть лише віддалено нагадувати те, що ми маємо на увазі зараз, уживаючи цей термін.

Хвиля трансформації торкнулася й операторів зв'язку – постачальників телекомунікаційних послуг. Ці компанії дотепер живуть у світі пропрієтарних продуктів, заснованому на застарілій моделі обчислень. При такому підході для кожної сервісно-орієнтованої мережної функції – будь те IMS, прикордонний контролер сеансів (SBC), SSGN/GGSN, брандмауер, глибока перевірка пакетів (Deep Packet Inspection, DPI) або мережа доставки контента (CDN) – застосовуються цілі стійки із пропрієтарною інфраструктурою. Всі ці компоненти здатні ефективно взаємодіяти з іншою мережею тільки після складного процесу установки й інтеграції, а за їхню роботу відповідають мережні адміністратори, що добре вивчили особливості саме цих конкретних продуктів.

Тривалі життєві цикли пропрієтарних рішень і тверді галузеві стандарти обмежують інновації й модернізацію, у результаті оператори зв'язку зазнають великого утиску з боку постачальників інтернет-послуг з їх більш гнучкою моделлю бізнесу. Провайдери, використовуючи оптимізовані процеси й гнучку інфраструктуру для надання аналогічних послуг по більш низьких цінах, оперативно реагують на зміну запитів ринку й швидко надають нові затребувані послуги. Тому оператори зв'язку все частіше зіштовхуються із дилемою – вибирати еволюційний шлях або йти з ринку.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи віртуалізацій мережних функцій NFV.

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

– Огляд існуючих систем віртуалізацій мережних функцій NFV.

– Дослідження системи віртуалізацій мережних функцій NFV.

– Програмна реалізація системи віртуалізацій мережних функцій NFV.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі віртуалізацій мережних функцій NFV.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи віртуалізацій мережних функцій NFV, 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

Віртуалізація мережних функцій (Network Function Virtualization, NFV) відкриває шлях для еволюції. У самому загальному виді цей підхід працює в такий спосіб:

– Спочатку ролі різних виділених пристроїв, призначених для реалізації мережних функцій, віртуалізуються, для чого вони розбиваються на ряд віртуальних мережних функцій (VNF). Останні виконуються на стандартизованих серверах, а для їхнього ефективного розгортання потрібно віртуалізація фізичної мережної інфраструктури.

– Для надання замовникові конкретної послуги створюються зв'язки між віртуальними мережними функціями. Для цього формуються сервісні ланцюжки, зміна, адміністрування й моніторинг яких здійснюється за допомогою відкритих платформ (наприклад, OpenStack).

– Після цього оператори зв'язку можуть інтегрувати цю оптимізовану віртуальну інфраструктуру у свою існуючу або оновлену систему оркестрації. На цьому етапі можлива інтеграція із системою OSS/BSS, що забезпечує облік, біллінг, динамічне надання й швидке створення нових послуг.

Одержувані в результаті використання такої моделі величезні переваги постачальники телекомунікаційних послуг можуть, у свою чергу, транслювати своїм клієнтам. Заміна пропрієтарного устаткування на стандартизовані сервери й мережні компоненти дозволяє гнучко й динамічно надавати нові послуги, скоротивши при цьому капітальні й операційні витрати. Така трансформація звільняє операторів від прив'язки до постачальників устаткування, що приводило до додаткових витрат. Клієнтам вона теж приносить чималу користь, тому що їм не доводиться знову й знову зіштовхуватися з нездатністю операторів оперативно забезпечувати відповідність своїх послуг мінливий бізнес-потребам. Крім того,

перенос всіх мережних функцій у єдину віртуалізовану інфраструктуру із загальною платформою знижує операційні витрати, тому що спрощуються моніторинг і адміністрування операцій.

У результаті оператори зв'язку можуть швидше підключати нових користувачів до своїх мереж, а для вже існуючих спрощується й прискорюється доступ до додаткових послуг. Але важливіше всього те, що технологія NFV дозволяє надавати нові послуги, що забезпечують окупність інфраструктури операторів зв'язку.

1.2 Область застосування

Однак перевагами технології NFV можна буде скористатися в масштабах галузі тільки після рішення деяких важливих завдань.

– **Накладені мережі SDN і NFV.** Створення накладених мереж, які зможуть успішно ізолювати фізичну й віртуальну мережу, принципово важливо для успіху як SDN, так і технології NFV.

– **Оптимізація стандартизованих серверів.** Перенесення віртуальних мережних функцій на стандартний сервер – ключовий елемент концепції NFV. З ним зв'язані специфічні складності, що стосуються підвищення продуктивності й взаємодії з іншими мережними пристроями. Основні зусилля потрібно зосередити на розробці високопродуктивного віртуалізованого ПЗ, що працює на стандартному устаткуванні.

– **Керованість.** Додаткові труднощі пов'язані з підключенням і адмініструванням віртуальних мережних функцій при формуванні сервісних ланцюжків засобами OpenStack (або аналогічними). Сервісам-провайдерам потрібно складне ПЗ для керування віртуальними мережними функціями, щоб забезпечити оркестрацію, моніторинг, обслуговування й біллінг послуг. Крім того, необхідно трансформувати традиційні мережні функції, щоб вписати їх у середовище на базі платформи OpenStack (або аналогів).

– **API-інтерфейси для оркестрації.** Поверх платформи інтеграції OpenStack (або аналога) перебуває рівень оркестрації сервіс-провайдеру, що підтримує функції OSS/BSS. У міру того як традиційні мережні функції будуть замінятися віртуальними, для систем OSS/BSS будуть вимагатися нові API-інтерфейси, що враховують ці зміни. В умовах швидко мінливої парадигми архітектури мереж ефективну допомогу може зробити методологія DevOps.

– **Демонстрація й тестування в робітничому середовищі.** Постачальники телекомунікаційних послуг звичайно не поспішають із впровадженням нових технологій, причому тривале ретельне тестування стало галузевий нормою. Методологія NFV повинна вийти за межі теоретичних розробок і демонстрацій і перейти в практичну площину. Це завдання потрібно вирішити в найближчі два роки. Щоб перейти від концептуальних розробок до реальної експлуатації, потрібно якнайшвидше приступитися до розгортання інноваційних систем у робочих мережах.

Для рішення кожної із цих завдань потрібні спеціальні знання й досвід. Постачальники досліджують різні можливості, вибираючи між пропрієтарним і відкритим підходом.

Найбільш перспективним є надання постачальникам телекомунікаційних послуг волі вибору в рамках відкритої платформи. Використання відкритих середовищ змушує виробників приділяти більше увагу функціональної сумісності своїх продуктів, а для забезпечення переходу на NFV саме потрібна сумісність всіх віртуалізованих функцій. Крім того, відкритий підхід успішно сполучається з іншими рішеннями (наприклад, SDN і хмарними технологіями, що широко поширилися) і дозволяє забезпечити синхронізовану роботу мережі – при такій умові замовникам вдається зберегти повний контроль над всіма своїми продуктами.

Відкритість IT і воля вибору = відкрита екосистема + відкрита оркестрація + відкриті протоколи + відкриті стандарти + відкритий код. Постачальники й оператори зв'язку, учасники групи розробки ETSI NFV, займаються створенням

відкритої платформи. Вона покликана об'єднати NFV, SDN і хмарні технології, а також доповнити переваги описаної вище відкритої екосистеми чіткими інструкціями й розширеною підтримкою, які допоможуть постачальникам послуг здійснити перехід на технологію NFV. Щоб прискорити вивід на ринок протестованих і перевірених замовниками рішень, члени цієї групи тісно співробітничають із телекомунікаційними компаніями й організаціями по розробці стандартів, наприклад з Європейським інститутом телекомунікаційних стандартів (ETSI). Кінцева мета – створення комерційної хмарної платформи для NFV, що забезпечить операторам зв'язку вибір рішень, найбільш відповідним їхнім вимогам.

Поза всяким сумнівом, принципи, на яких заснована технологія NFV, будуть рано або пізно реалізовані. Постійний розвиток відкритих екосистем і стандартів принципово важливо для майбутнього технології NFV, так що, будемо сподіватися, постачальники, зацікавлені в одержанні максимальної віддачі від своїх рішень NFV, підтримають цю ініціативу.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи віртуалізацій мережних функцій NFV, 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

Spiceworks

Відсутність плати виділяє Spiceworks серед інших продуктів управління мережею, але користувачам прийдеться примиритися з показом у програмі ненав'язливих рекламних оголошень. Якщо реклама дратує або заборонена правилами компанії, від її можна позбутися, заплативши 45 дол. на місяць. Якщо договір укладається на рік, надається невелика знижка (495 дол. замість 540). Інша відмітна риса Spiceworks – повнофункціональна служба підтримки.

Важливу роль грає й активне співтовариство користувачів Spiceworks. Наприклад, можна встановити контакт із місцевими користувачами Spiceworks через службу SpiceCorps або брати участь у традиційних форумах на сайті співтовариства (community.spiceworks.com).

Установити Spiceworks просто. Після завершення процедури установки потрібно створити початковий обліковий запис і пароль. Потім з'являється основний запит відносно початку роботи. Вибрати можна із трьох варіантів: Inventory (інвентаризація), Help Desk (служба підтримки) і Configuration Backup (резервне копіювання конфігурації).

З розділу Inventory запускається IP-сканування мережі. Майстер допоможе підготувати підходящі облікові дані для Windows, UNIX, Apple і інших серверів, доступних через оболонку Secure Shell (SSH), а також для принтерів, комутаторів або інших пристроїв SNMP. Для пошуку, виконання процедури реєстрації й інвентаризації всіх мережних пристроїв у моїй тестовій мережі треба було кілька хвилин.

Відразу після завершення сканування від Spiceworks по електронній пошті прийшло повідомлення з докладним описом результатів. На головній сторінці інвентаризації є поточний журнал із вказівкою виявлених об'єктів і примітками щодо неполадок. Наприклад, Spiceworks виявив сервер VMware ESXi, але не зміг одержати докладні відомості про нього, тому що уведені мною ім'я користувача й пароль були неправильними. По клацанню на сервері ESXi було видано меню, за допомогою якого вдалося усунути проблему перевірки дійсності. У ході чергового сеансу сканування сервер ESXi був виявлений, виконані реєстрація й дані інвентаризації оновлені. Spiceworks не тільки надав інформацію про кожну віртуальну машину, але й перелічив імена сховищ даних VMware із вказівкою вільного простору в кожному з них. Завдяки інтуїтивно зрозумілому інтерфейсу ви зможете без праці знаходити й виправляти типові помилки конфігурації.

Я якийсь час провів спостереження за приладовою панеллю, показаної на рисунку 2.1. За замовчуванням на ній показаний загальний вид мережі й відображаються відомості з 12 основних категорій, у тому числі стан антивірусного захисту, дані Microsoft Exchange Server, зведення інвентаризації, строки завершення дії гарантій і попередження. Приладова панель відрізняється гнучкістю налаштування, наприклад можна додавати, видаляти й переміщати розділи.

Я також протестував службу підтримки Spiceworks. Як адміністраторові служби технічної підтримки університету мені було цікаво подивитися, наскільки це рішення придатне для виконання масштабних завдань. У допомогу початківцем є чотири заздалегідь підготовлених квитки, які дозволяють познайомитися з організацією служби підтримки і її налаштувань. Я швидко переконався, що служба підтримки елементарна. Наприклад, у ній немає групових функцій, типів запитів і шляхів переходу на більше високий рівень – всі ці можливості дуже важливі для великої IT-організації.

Рисунок 2.1 – Приладова панель Spiceworks

Одна за переваг служби підтримки – добування даних для квитків з повідомлень електронної пошти. Користувачі просто відсилають свої запити по електронній пошті, а служба підтримки збирає інформацію з повідомлень і автоматично формує квиток. Інша вдала знахідка – довідковий портал, через який користувач може зайти на веб-сторінку (з перевіркою дійсності Active Directory) і представити свої квитки.

Така служба підтримки ідеальна для компаній з декількома користувачами. Помнете тільки, що вона погано піддається масштабуванню при збільшенні числа користувачів або адміністраторів.

В Spiceworks представлений 21 шаблоновий звіт, у тому числі про інформацію у квитках служби підтримки, списки комп'ютерів без антивірусних програм і комп'ютерів з невеликою кількістю вільного простору на дисках. Можна також завантажити підготовлені співтовариством звіти із сайту співтовариства. Моя увага залучили два звіти: в одному втримувалася інформація про використання Exchange (наприклад, час останньої реєстрації користувача,

розмір поштової скриньки користувача й загальна кількість об'єктів), а в іншому перераховані локально підключені принтери.

Завершує перелік компонентів цієї повнофункціональної платформи мобільний додаток Spiceworks. Воно дозволяє перевіряти квитки служби підтримки, стежити за попередженнями щодо інвентаризації, читати новітні коментарі в співтоваристві Spiceworks і т.д. Якщо доводиться обслуговувати кілька сайтів, можна підготувати профіль для кожного сайту, щоб не запам'ятовувати комбінації ім'я користувача/пароль/адреса.

Spiceworks – чудовий інструмент. Він повнофункціональний і надається безкоштовно. Навіть при платі за відсутність реклами 45 дол. на місяць або 495 дол. у рік покупка буде дуже вигідною. Це оптимальний варіант для малих компаній, що не бідують у дорогому рішенні.

Foglight NMS

Другий продукт даного огляду, Foglight Network Management System (NMS) компанії Quest Software, також надається безкоштовно. За допомогою цієї програми з потужною функціональністю можна безкоштовно відслідковувати до 100 пристроїв у мережі. Якщо потрібно контролювати більше пристроїв, купите повну версію Foglight NMS, заплативши 99 дол. за кожний пристрій понад сотню. Тому, наприклад, якщо в мережі 200 пристроїв, прийде заплатити 9 900 дол. (100 ? 99 дол.). При цьому ви зможете використовувати три модулі розширення повної версії (Traffic Analysis, IPSLA-VoIP і Remote Site Monitoring with Pollers) з усіма пристроями. Можна також придбати два інших модулі розширення: для моніторингу продуктивності Performance Monitoring і керування конфігурацією Configuration Management.

На жаль, на сайті Quest нелегко знайти інформацію про п'ять модулів розширення. Зацікавленим користувачам варто безпосередньо звернутися в компанію Quest.

До складу як безкоштовної, так і повної версій Foglight NMS входять модулі Network Flow Analyzer Module, Remote Agent Module, VoIP Monitoring

Module і Wireless Monitoring Module. У продажі є й додаткові компоненти, у тому числі vFoglight для віртуальних машин.

Як і Spiceworks, Quest користується активною підтримкою співтовариства. У продукті є посилання Community, зв'язана прямо із сайтом співтовариства (www.quest.com/communities), де можна задати питання колегам і навіть довести свої пропозиції до групи розроблювачів. Однак більшість форумів орієнтована строго на продукти Quest, а загальні теми обговорюються не занадто активно. Наприклад, форум SharePoint був порожній, а на форумі Oracle я знайшов усього дев'ять публікацій, більшість із яких з'явилося більше шести місяців назад.

Foglight NMS варто встановлювати на окремому комп'ютері або віртуальній машині. У ході повністю автоматизованого процесу встановлюються необхідні продукти, у тому числі Microsoft.NET Framework 4.0 і SQL Server Compact. Як відзначається в документації по системних вимогах, SQL Server Compact використовується при апробуванні Foglight NMS або завантаженню певної кількості пристроїв. У виробничих умовах необхідно використовувати SQL Server Standard або Enterprise Edition.

Після завершення установки я виконав реєстрацію в Foglight NMS Studio і зареєстрував продукт. На даному етапі варто вибрати безкоштовну версію або ввести ліцензійний ключ, якщо потрібно відслідковувати більше 100 пристроїв. Хоча в мене був ліцензійний ключ, я розглянув безкоштовну версію.

На екрані-заставці приводяться посилання на статті в довідковій системі, у яких описуються кроки по початковому налаштуванню системи моніторингу. Контролювати пристрою можна через протокол SNMP, за допомогою інструментів керування Windows (WMI), розгортаючи віддалені агенти або збираючи дані системного журналу, NetFlow або пасток SNMP.

Я вирішив використовувати SNMP і WMI для моніторингу пристроїв у тестовій мережі. Я нажав кнопку Add Device(s), і на екрані з'явився майстер, за допомогою якого можна додавати пристрою через функцію виявлення мережі SNMP або розгортання агентів. Функція виявлення мережі SNMP успішно

виявила в тестовій мережі всі пристрої. Після цього в меню ліворуч було показано, що 11 пристроїв перебувають у стані Credential Not Set (облікові дані не задані). Клацаючи елементи цього меню, можна одержати список пристроїв, у які програмі Foglight NMS не вдалося виконати вхід. Серед них були сервер Apple, мережні пристрої й Windows Home Server. Із цього списку мені вдалося призначити дійсні облікові дані для кожного пристрою. Інтерфейс настільки інтуїтивно зрозумілий, що мені не довелося користуватися довідковою системою або шукати допомоги в Інтернеті. Після призначення облікових даних Foglight NMS активно відслідковував всю тестову мережу.

Хоча продукт надається безкоштовно, він має у своєму розпорядженні широку функціональність. Зокрема, я ознайомився з функцією створення карти мережі. Спочатку потрібно ввести піктограми, що представляють пристрої, а потім об'єднати їхню конфігурацію, що відповідає фізичній структурі мережі, у графічному інтерфейсі на основі Adobe Flash Player. Можна підготувати більше однієї карти, щоб показати кілька рівнів. Наприклад, одну карту можна накласти на зображення території США. Кількість деталей на кожній наступній карті може збільшуватися.

У ході роботи я помітив, що сервер ESXi був невірно визначений програмою Foglight NMS. Можливо, причина в тім, що в мене була стара версія (3.x). Крім того, в 2010 році ESXi був перейменований в vSphere Hypervisor. Настроїти Foglight NMS вручну для розпізнавання сервера як пристрій VMware було неважко. Після уведення вірних облікових даних програма негайно приступилася до збору статистики про використання процесора, пам'яті й т.д. Є статистика навіть для віртуальних машин.

Якщо в мережі є кілька фізичних сайтів, корисно мати сервер Foglight NMS скрізь. У цьому випадку можна підготувати різні сайти, щоб було простіше організувати відслідковуються устройства, що. Потім ці сайти передають дані центральному серверу Foglight NMS.

Можна призначити попередження, які не тільки сповіщають адміністратора, але й запускають певні дії, наприклад зупиняють службу або запускають спеціальний сценарій.

В Foglight передбачена можливість збору статистики NetFlow. Вибираючи пункт NetFlow у меню Tools у перший раз, користувач одержує пропозицію завантажити пакет PacketTrap Tool Suite (розмір 10 Мбайт). Це пробна версія, але термін дії – до 2031 року, тому часу для ознайомлення досить. Як повідомляє Келли О'двайер-Мануель, старший аналітик і фахівець зі зв'язків із громадськістю компанії Quest, у недалекому майбутньому інструмент буде перейменований в Quest Free Network Tools, і пробна версія просто стане безкоштовним додатком. Однак таке додавання не робить сильного враження. Властиво Foglight NMS – відмінна програма, але спосіб інтеграції PacketTrap із продуктом не можна визнати зробленим.

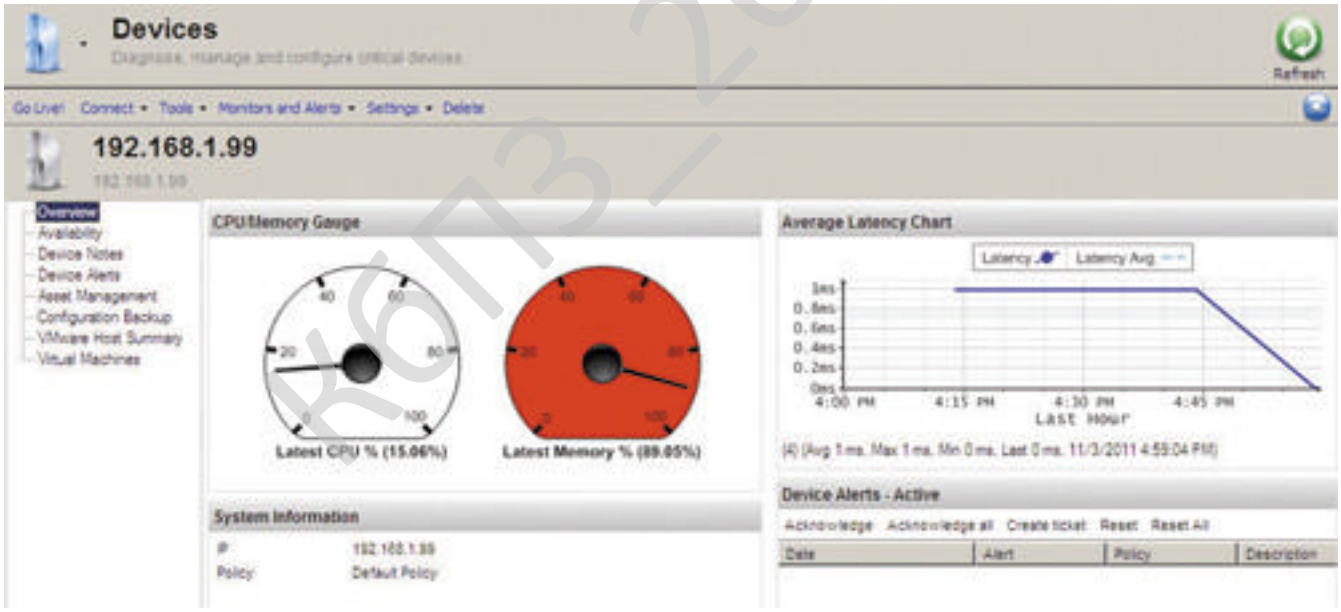

Рисунок 2.2 – Приладова панель Foglight NMS

Foglight NMS – додаток із продуманою структурою й відмінною приладовою панеллю, як показано на рисунку 2.2. Якщо є служба підтримки, який потрібно переглянути стан мережі, то через сайт можна сформувати

надання, доступне тільки для читання. Для цього досить додати порт 5053 до ім'я сервера (наприклад, https://Foglight:5053).

У цілому Foglight NMS – інтуїтивно зрозумілий і простій для використання продукт. Середнім компаніям варто звернути увагу, що він надається безкоштовно для обслуговування 100 пристроїв. Мені, наприклад, потрібний безкоштовний мережний монітор для домашньої мережі, і, схоже, я знайшов його.

WhatsUp Gold

WhatsUp Gold компанії Ipswitch – зрілий продукт, що існує вже більше 10 років, і це наочно проявляється в його глибокій і потужній функціональності. Фахівці Ipswitch прислухалися до користувачів і внесли поліпшення на підставі їхніх коментарів і пропозицій.

Випускається три редакції WhatsUp Gold: Standard, Premium і Distributed. До складу Premium Edition входять усе компоненти Standard Edition, а також функції моніторингу через WMI, моніторингу UNIX і Linux, моніторингу бездротових вузлів доступу й т.д. Редакція Distributed доповнена функціональністю для мереж, розподілених по більших територіях. Докладний список компонентів кожної редакції опублікований на сайті WhatsUp Gold. Якщо виникає потреба переходу на більше високий рівень, повторна установка не обов'язкова. Досить увести новий ліцензійний ключ, і продукт автоматично обновляється до нової версії.

Я познайомився з WhatsUp Gold v15 Premium Edition. Процес установки простий, тому що в ньому передбачені всі умови, зокрема установка. NET Framework 4.0, IIS і SQL Server Express 2005. Власники великої мережі, можливо, зволіють використовувати екземпляр SQL Server 2008 або SQL Server 2005 замість SQL Server Express. У пакеті установки втримуються дві надбудови: WhatsConnected і WhatsConfigured. Для даного огляду я встановив тільки WhatsUp Gold.

Після установки майстер проводить користувача по етапах налаштування оповіщень SMTP і облікових даних для різних пристроїв, підключених до мережі. У процесі сканування мережі можна спостерігати, як WhatsUp Gold виявляє кожний пристрій, перетворить його ім'я вузла й намагається визначити його роль (наприклад, вузол VMware, сервер Windows).

WhatsUp Gold успішно виявив і ідентифікував пристрою, а помилки у визначенні було легко виправити. Достоїнство продукту – можливість налаштування будь-яких параметрів з консолі Windows або веб-консолі.

У більших мережах можуть бути присутнім сотні й навіть тисячі пристроїв. В WhatsUp Gold вони групуються, щоб адміністраторові було простіше знайти потрібні пристрої. Існує два види груп: нединамічні й динамічні. Перші управляються вручну. Наприклад, якщо є два місця розташування, можна створити групу «Київ» і групу «Кіровоград», а потім перенести пристрої за допомогою методу drag-and-drop у відповідні групи.

За замовчуванням в WhatsUp Gold сформовано 20 динамічних груп, а в базі дані довідки є ще 14. Вони заповнюються з використанням інструкцій SQL. Наприклад, щоб помістити всі пристрої Cisco в одну динамічну групу, інструкція SQL може мати такий вигляд:

```
SELECT DISTINCT Device.nDeviceID
FROM Device
WHERE Device.bRemoved = 0 AND
Device.sSnmpOID LIKE N'1.3.6.1.4.1.9%'
```

Цих прикладів досить, щоб навіть адміністратори з поверхневими знаннями про SQL могли створити власні динамічні групи. Але такий метод здасться незручним для людей, які рідко звертаються до SQL. Хотілося б, щоб розроблювачі WhatsUp Gold зберегли можливість працювати вручну з функціями SQL (для користувачів, знайомих з SQL), але додали майстер, що допомагає будувати інструкції SQL. Компанія Ipswitch публікує схему бази даних на своєму сайті, що спрощує підготовку запитів і створення спеціалізованих звітів. Природно, запис ззовні додатка WhatsUp Gold неможлива, але дозволяється доступ для читання, якщо це допоможе зібрати потрібні дані.

Крім звичайної перевірки за допомогою команди ping, WhatsUp Gold контролює характеристики продуктивності (наприклад, використання процесора й пам'яті) і служби (наприклад, DNS, HTTP). Є також «пасивні монітори» для контролю пасток SNMP, системних журналів і журналів подій Windows.

При установці й налаштуванні інструмента моніторингу зручніше користуватися консоллю Windows, а у веб-консолі краще показаний стан кожного пристрою. У випадку відмови пристрою відповідна піктограма стає жовтою, і у веб-консолі відображається вказівка, який монітор (наприклад, ping або DNS) видає помилку й протягом якого часу. Якщо пристрій як і раніше не відповідає, колір піктограми стає червоним. Клацнувши на пристрої, можна одержати додаткові відомості, що полегшують усунення неполадок.

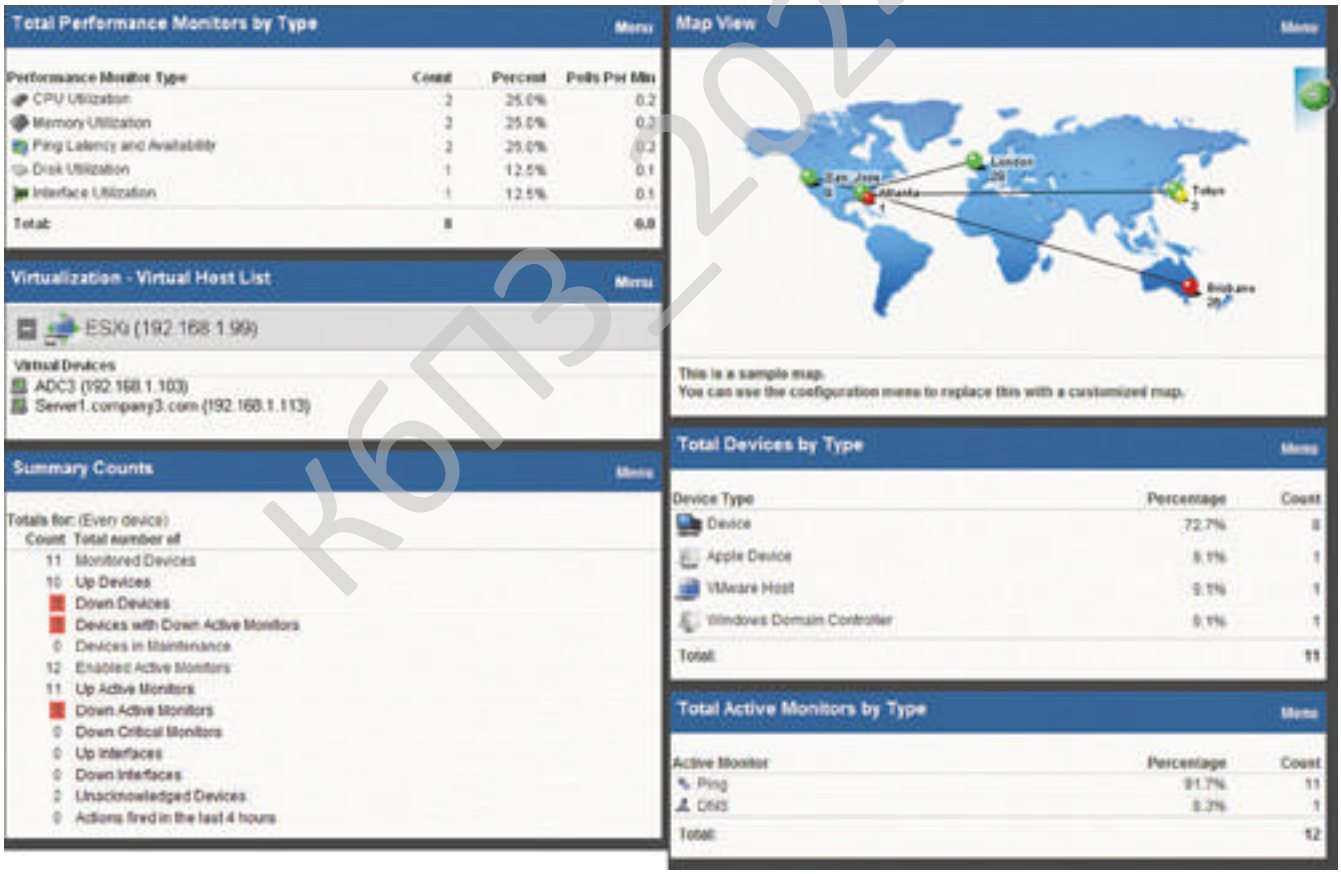

Рисунок 2.3 – Приладова панель WhatsUp Gold

Налаштування приладової панелі WhatsUp Gold дуже гнучкі. Не становить праці додавати, видаляти елементи й змінювати приладову панель відповідно до власних переваг, як показано на рисунку 2.3.

Компанія Ipswitch працює давно й заслужено має авторитет. Новітня версія WhatsUp Gold мене не розчарувала.

Orion Network Performance Monitor (NPM)

Orion Network Performance Monitor (NPM) компанії SolarWinds – ще один зрілий продукт. Його ціна висока в порівнянні з безкоштовними продуктами, але витрати окупаються завдяки широким можливостям.

Я встановив Orion NPM на виділеному сервері Windows Server 2008, члені домена тестової мережі. Установка на контролері домена (DC) не підтримується. У ході установки забезпечуються всі необхідні умови, у тому числі встановлюється. NET Framework 3.5 з пакетом відновлення SP1 і IIS. Зібрані дані зберігаються в обов'язковій базі даних на сервері. У невеликих і тестових мережах можна використовувати SQL Server Express 2005, але рекомендується SQL Server 2005 або більше нова версія.

При першому звертанні до адміністративного веб-консолі майстер допоможе задати облікові дані для SNMP, VMware і Windows, і вказати IP-адреси для перевірки. Потім результати імпортуються в базу даних.

Всі пристрої в моїй мережі були виявлені без проблем, у тому числі сервер ESXi. Коли я розгорнув значок ESXi, на екрані з'явився повний список розміщених на сервері віртуальних машин. Провівши курсором миші над будь-якою віртуальною машиною, можна одержати зведення її стану. Імена неактивних віртуальних машин показані сірим кольором.

Замість установки агентів для контролю пристроїв використовується протокол Internet Control Message Protocol (ICMP – або команди ping), SNMP, WMI, системний журнал або процедура реєстрації в пристрої. Відповідно до керівництва адміністратора Orion NPM, методи без агентів використовуються по наступних причинах:

– NPM не задіє служби, що відбирають ресурси у важливих прикладних програм;

– NPM не встановлює ніяких програм на контрольованих мережних пристроях; некеровані або застарілі програми можуть стати уразливим місцем у мережі.

На мій погляд, це ясна відповідь на питання про переваги моніторингу з використанням агентів або без них.

Структура приладової панелі Orion NPM добре продумана й зручна для переміщень. Наприклад, на рисунку 2.4 показані зручні вкладки у верхній частині панелі, такі як Top10, Alerts (попередження), Syslog (системний журнал) і Events (події). Клацаючи кожний контрольований пристрій, подібно серверу ESXi і його віртуальним машинам, можна перейти на екран з більше докладною інформацією. На таких екранах розміщені численні «спідометри», на яких динамічно відображаються статистичні дані, у тому числі середній час відгуку, втрати пакетів, середня швидкість процесора й використана пам'ять. Клацнувши на спідометрі, можна одержати графік зі значеннями раніше зроблених вимірів. Є графік, що налаштовується також, на якому показаний стан пристрою за кілька тижнів або місяців.

В Orion NPM можна імпортувати інструмент Network Atlas. З його допомогою ви можете створити карту мережі в чотири етапи: виберіть тло, помістіть об'єкти на карту, з'єднаєте об'єкти з об'єктами внутрішньої бази даних і зробіть необхідні налаштування. Компанія SolarWinds надає більше 30 карт, у тому числі карти миру й окремих континентів. Менш чим за хвилину я підготував карту США, додав піктограму й зв'язав піктограму на карті з комутатором своєї тестової мережі.

SolarWinds застосовує цікаву модель ліцензування. Замість єдиної ціни за один пристрій загальна вартість розраховується по найбільшому числу інтерфейсів, вузлів або томів. Наприклад, якщо є два 48-портових комутатори, 100 серверів/вузлів і 20 томів на жорстких дисках, споживач платить тільки за

найбільше число, у даному прикладі – 100 серверів/вузлів. Я ввів цю інформацію в калькулятор ліцензій, доступний через Інтернет, і одержав ціну 3595 дол. Інші контрольовані пристрої виявляються «безкоштовними». Але перш ніж оформити покупку в Інтернеті, подзвоните в компанію й переконаєтеся, що не придбаєте зайві ліцензії.

Рисунок 2.4 – Приладова панель Orion NPM

Трапляється, що, призначивши сигнал тривоги для певного пристрою, адміністратор згодом не одержує попередження про аварію, тому що сигнал був настроєний невірно. До складу Orion NPM входить інструмент Test Fire Alerts, за допомогою якого можна перевірити правильність спрацьовування тривоги у випадку відмови пристрою. Одна з особливостей Orion NPM – спосіб реалізації окремих інструментів. Замість того щоб помістити всі інструменти в один виконується файл, що, багато хто з них представлені окремими програмами, які потрібно встановлювати. Я нарахував 15 таких програм.

Orion Network – перевірена програма керування мережею, потужна й надійна. Але розібратися в незвичайній моделі ліцензування буває нелегко, а ціна може виявитися занадто високої для деяких компаній.

NEC ProgrammableFlow

NEC розробила комерційно доступне рішення SDN для корпоративних мереж на базі OpenFlow. Воно називається ProgrammableFlow і, як заявляють у компанії, забезпечує ефективне використання всіх IT-ресурсів за рахунок віртуалізації мережі й надає прості механізми створення, керування, моніторингу й експлуатації багатокористувальницьких мереж. Лінійка устаткування NEC ProgrammableFlow містить у собі контролер OpenFlow, віртуальні й фізичні комутатори.

До переваг ProgrammableFlow в NEC відносять просте створення логічних мереж необхідної топології, візуалізацію мережі й трафіку, спрощений моніторинг і забезпечення SLA, простоту реалізації додаткових функцій (захист від DDo, VPN і т.п.) за допомогою зовнішніх програмних додатків, а також можливість створення власних додатків.

ProgrammableFlow дозволяє формувати на базі єдиної фізичної інфраструктури безліч логічних ізольованих мереж IP (Virtual Tenant Network, VTN) з різними політиками й топологією й поєднувати територіально розподілені фрагменти SDN у єдину хмарну мережну інфраструктуру з можливістю міграції IT-ресурсів. Створювати й налаштовувати VTN можна централізовано, а також делегувати ці права користувачам. Кожна VTN має власну топологію й мережних політиків, включаючи сценарії обробки трафіку за допомогою мережних компонентів, розміщених у довільній крапці мережі. Щоб обслуговувати фізичну мережу (незалежно від логічних мереж), не потрібний висококваліфікований персонал – досить установити устаткування й підключити кабелі.

Мережі від L2/L3 до L4/L7 можна створювати із графічного інтерфейсу. При цьому є можливість підключення спеціалізованих фізичних мережних пристроїв (appliance), а платформа віртуалізації на базі серверів x86 і гіпервізора

KVM дозволяє запускати віртуальні пристрої (додатка NEC і інших компаній) і має всі ознаки хмарної платформи – зокрема, її можна динамічно масштабувати. Таким чином, завдяки масштабованості мережа ProgrammableFlow можна проектувати з обліком поточних, а не майбутніх вимог, що веде до скорочення початкових інвестицій. Пізніше її можна поступово нарощувати. VTN можуть поєднуватися в домени більше високого рівня.

Ключовий компонент – система оркестрації. Фактично вона містить у собі два оркестратора – сервіси й сама платформа. За допомогою першого користувач може формувати запити до платформи й одержувати необхідну мережну функціональність. Візуалізація мережної інфраструктури дозволяє бачити в інтерфейсі контролера SDN віртуальні й фізичні мережі.

Накладені мережі

Лінійка продуктів SDN від NEC містить у собі стандартний контролер OpenFlow і лінійку комутаторів з підтримкою SDN. Раніше впровадження даних рішень вимагало заміни мережного устаткування – IP-Комутаторів і маршрутизаторів, а інтегрувати їх з існуючими рішеннями було непросто. Така заміна не тільки дорогий процес – впровадження нової технології завжди пов'язане з додатковими ризиками для операторів.

Цього року зроблений значний крок до «комерціалізації» інфраструктури SDN. Недавно випущений «оверлейний» контролер допомагає організувати SDN поверх існуючих мереж IP, а нова версія ПЗ відкриває можливості для побудови ієрархічних систем. Контролер дозволяє адмініструвати всю фізичну мережу без участі людини, підтримує довільну топологію, забезпечує відказостійкість за рахунок самовідновлення й автоматичного розподілу потоків трафіку в мережі, а також 100-процентне завантаження каналів».

Оверлейний контролер NEC PF7600 дозволяє створювати накладені мережі SDN поверх мережі IP і інтегрувати наявні мережі із сегментами OpenFlow. На рівні логічних мереж він надає інтерфейс для простого створення VTN, візуалізує схему мережі й трафік у логічній і фізичній інфраструктурі. У

новому рішенні NEC спрощується реалізація додатків SDN, що функціонують у рамках логічної мережі. Користувач кожної такої мережі може сам задавати в графічному інтерфейсі її топологію й налаштовувати політики для свого віртуального сегмента. Контролер автоматично застосовує їх до фізичної інфраструктури. Можна створювати нові додатки й використовувати додатки сторонніх постачальників, масштабувати мережа (додаючи або забираючи вузли) без переривання обслуговування.

ProgrammableFlow спрощує конфігурування мережі в порівнянні з тим, як це робиться при традиційному підході, коли IP-адреси й VLAN призначаються відповідно до топології фізичної мережі, а конфігурація кожного окремого комутатора створюється з урахуванням мережного оточення й потім завантажується в нього. Користувач задає тільки логічні з'єднання мережі й прописує їх у контролері. Цей контролер самостійно генерує таблицю потоків трафіку, що реалізує задану конфігурацію віртуальної логічної мережі, і автоматично застосовує її до фізичної мережі. Інакше кажучи, контролер самостійно створює конфігурацію логічної мережі поверх фізичної. Крім того, він відслідковує перенапрямок трафіку при збоях у мережі й перерозподіляє його при додаванні додаткових комутаторів.

Віртуалізація мережі дозволяє створювати віртуальні комутатори, маршрутизатори, термінальні пристрої й т.д. За допомогою віртуальних компонентів можна побудувати безліч логічних мереж, за динамікою їхнього розміщення можна спостерігати на карті фізичної мережі.

При створенні ВМ або підключенні фізичної хост-системи до мережі SDN установлюється відповідність між фізичним портом комутатора OpenFlow і хостом. Серверна віртуалізація дозволяє ефективно вирішувати завдання резервного копіювання й міграції ВМ. Віртуальні мережі спрощують міграцію ВМ, тому що не вимагають внесення змін у мережні налаштування. При переносі ВМ засобами гіпервізора або переміщенні хоста у віртуальній мережі ніяких переналаштувань не знадобиться. Будь-які зміни автоматично фіксуються, і всі

потоки трафіку відповідним чином маршрутизуються. Ця можливість реалізується й у рамках розподілених ЦОД. Якщо ВМ, зареєстрована у віртуальній мережі, мігрує, то ніяких операцій з мережним оточенням робити не треба.

Контролер використовує набір програмних компонентів, включаючи Overlay Agent (Open vSwitch), Overlay Gateway і Overlay Reflector для зв'язку із зовнішньою мережею (IP або OpenFlow) і роботи із трафіком BGP і Multicast. Функції накладеної мережі реалізуються за допомогою VXLAN поверх мережі IP. Протокол VXLAN підтримується провідними виробниками комутаторів і маршрутизаторів.

Гібридна інфраструктура

Перевагою гібридної інфраструктури NEC є можливість використання існуючої фізичної мережної інфраструктури й мережі SDN на базі ProgrammableFlow. Останній здатний забезпечити високу доступність і новітні засоби керування трафіком, наприклад QoS для тунельного трафіку. На рівні логічної мережі функціонують VLAN і накладені тунелі VXLAN: їхня комбінація розширює можливості створення віртуальних користувальницьких мереж.

На основі такої схеми можна створити мережу SDN у рамках декількох відособлених фізичних доменів – наприклад, двох ЦОД або ЦОД і філіальних мереж. Оркестратор Unified Network Coordinator (UNC) дозволяє об'єднати контролери й побудувати ієрархічну інфраструктуру SDN. Зв’язність компонентів і фізичних доменів забезпечується ним за допомогою L2 VPN – тунелів GRE або VXLAN через існуючу транспортну мережу. Завдяки цьому якщо буде потреба IT-ресурси будуть автоматично (через UNC) задіяні в іншому ЦОД. Користувач цього навіть не помітить. Дана схема дозволяє здійснювати й резервування ресурсів.

Сегментом SDN управляє контролер PFC (контролер OpenFlow), а за накладену мережу VXLAN, побудовану на базі існуючої мережі IP, відповідає контролер OVC. Всім цим управляє UNC. В 2014 році він був визнаний кращим

рішенням з погляду сумісності компонентів при побудові мереж SDN. OVC управляє накладеними тунелями VXLAN або GRE, що зв'язують фізичні домени поверх транспортної мережі. Це дає можливість вирішити більшість завдань побудови L2 VPN. Для тунелів можна налаштовувати SLA: контролер здатний ураховувати задані параметри. Раніше контролер OpenFlow «бачив» тільки параметри фізичного інтерфейсу, що утрудняло інтеграцію мережі SDN з IP-сегментами, що характеризуються змінною пропускною здатністю.

Серед переваг ЦОД на базі ProgrammableFlow (VXLAN Overlay + OpenFlow SDN) називаються наступні: зниження капітальних витрат завдяки віртуальній мережній інфраструктурі; візуалізація основної інформації, що допомагає в усуненні неполадок; підвищення ефективності використання й надійності за рахунок контролю й керування трафіком; керування з'єднаннями між ЦОД, а також підтримка сервісів NFV.

Наскільки зрілими є сьогодні рішення SDN? Чи можна будувати масштабні мережі? Оверлейні контролери дозволили переглянути масштаб цієї мережної інфраструктури. NEC UNC підтримує до 10 контролерів OpenFlow, кожний з яких обслуговує до 250 комутаторів. Інакше кажучи, один сегмент SDN може містити два з половиною комутуючих вузла й півмільйона хостів. Така мережа здатна обслуговувати мільйони потоків даних. На базі єдиної фізичної інфраструктури створюється до 30 тис. незалежних фізичних мереж зі своїми мережними політиками й SLA, причому користувач самостійно управляє логічною мережею. Локальні сегменти можна будувати з доменів OpenFlow. Кожна VTN містить до 4 тис. VLAN, а контролер OpenFlow підтримує до 16 млн VLAN. Таким чином, на рівні VXLAN будуть функціонувати 16 млн глобальних VTN.

Хмарний ЦОД

Поряд із уже згаданими продуктами, NEC пропонує компоненти для побудови хмарного ЦОД, всі елементи якого віртуалізовані. Такий ЦОД підтримує традиційну IP-інфраструктуру й інфраструктуру SDN на базі

OpenFlow. Інфраструктура SDN взаємодіє із платформою OpenStack (NEC розробила й власний оркестратор) і забезпечує спільну роботу систем серверної віртуалізації, СХД для організації пулів ресурсів і автоматизації розгортання сервісів.

Процес створення приватної хмари в поданні NEC містить у собі кілька етапів. Перші два передбачають віртуалізацію частини системи (звичайно серверів), а потім всієї інфраструктури IT (з використанням мереж SDN і фабрик Ethernet). На цьому в більшості випадків і зупиняються. Однак істотного економічного ефекту подібний перехід не дає, а від фахівців буде потрібно ще більш висока кваліфікація. Зокрема, вони повинні вміти не просто адмініструвати мережу, а програмувати її.

NEC звертає особливу увагу на наступні етапи, що дозволяють домогтися оптимізації витрат керування: стандартизацію керування (створення пула ресурсів IT), його автоматизацію (у тому числі адміністрування) і самообслуговування, що дозволяє делегувати службові завдання. Два останніх приносять реальну вигоду від впровадження хмарної системи. У ній автоматизовані всі процеси, пов'язані з використанням ресурсів і створенням серверів, керуванням і системними додатками.

MasterScope Cloud Manager і MasterScope vDC Automation

У хмарних структурах на додаток до керування мережею OpenFlow необхідно централізоване керування компонентами традиційної мережі, такими як міжмережні екрани, балансувальники навантаження й т.п. У гібридних інфраструктурах, включаючи традиційну мережу й мережу SDN/OpenFlow, сервери й СХД, такий підхід дозволяє знизити витрати при автоматизації хмарної системи в цілому.

Для рішення даного завдання використовуються інфраструктурний і сервісний оркестратори. В NEC це ПЗ MasterScope Cloud Manager і MasterScope vDC Automation. Перше підтримує керування хмарними послугами, друге забезпечує гнучкість, безпеку й автоматизацію керування ЦОД, а також

керування життєвим циклом ресурсів IT. Адміністратор бачить у графічному інтерфейсі пул ресурсів і може створити сценарії для кінцевих користувачів, за допомогою яких вони можуть формувати віртуальний ЦОД для виконання своїх завдань, дотримуючи простої послідовності кроків. Система сама виділяє ресурси IT і розгортає мережу із заданими параметрами на базі наявної фізичної інфраструктури.

Такий ЦОД уже функціонує в Японії. За допомогою SDN у ньому віртуалізована вся інфраструктура й забезпечена повна автоматизація всіх процесів ініціалізації/налаштування, включаючи мережу, сервери, СХД і ПЗ послуг («віртуальний ЦОД як послуга»). По даним NEC, це забезпечило 50-процентне зниження вартості як мережної інфраструктури й експлуатації.

SDN допомагає вирішити безліч завдань, однак у її реалізаціях можуть застосовуватися різні механізми, і згодом їхній спектр, імовірно, стане ще ширше. Швидше за все, будуть створюватися гібридні, комбіновані мережі, де сполучаються нові й традиційні підходи. Компанії розуміють, що, розгорнувши SDN уже зараз, вони одержують конкурентну перевагу й зможуть випередити інших гравців ринку при масовому впровадженні нової технології, тому багато операторів розгортають тестові конфігурації SDN. Завдяки вдосконалюванню підходів, не потребуючі заміни існуючої мережної інфраструктури, і появі нових продуктів SDN знижуються ризики впровадження програмно конфігуруємих мереж, а замовники можуть оптимізувати витрати на їхнє розгортання.

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 віртуалізацій мережних функцій NFV.

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

Програмне конфігурування мереж (Software-Defined Networking, SDN) і віртуалізація мережних функцій (Network Functions Virtualization, NFV) входять до числа найбільш обговорюваних тим сучасної мережної галузі. В NFV віртуалізація застосовується до функціональності мереж. Відповідно до концепції віртуалізації мережних функцій, пропрієтарне мережне устаткування й спеціалізовані мережні пристрої заміняються на програмне забезпечення, що виконується на серверах загального призначення – комерційних стандартних платформах.

Модель розгортання NFV може бути централізованої, коли сервери розміщаються в ЦОД оператора, або розподіленої (Distributed NFV, D-NFV), при якій віртуалізація мережних функцій здійснюється в будь-якому місці мережі – у крапках присутності оператора й навіть на стороні користувача, у клієнтському мережному устаткуванні. Остання концепція, поза тим що може виявитися більше ефективної й економічно виправданої, знижує ризики переходу операторів на нову модель.

Рішення RAD для D-NFV була відзначено як інновація року в області NFV. У ньому використовується устаткування ETX – мережний інтерфейсний пристрій другого/третього рівня, оснащене модулем D-NFV і інтегрованим сервером x86 для віртуалізації сервісів на площадці клієнта. Компанія, що активно бере участь у діяльності відповідних галузевих органів стандартизації NFV, не дуже давно відкрила новий центр розробки й досліджень із фокусом на NFV і SDN.

Ще в 2004 році BT запропонувала ідею мережної інфраструктури 21 Century Network з метою зниження операційних витрат, прискореного

впровадження нових сервісів і надання користувачам більше гнучких можливостей. Вона припускала об'єднання ядра мережі й мережі доступу в рамках єдиної мережної платформи, але без віртуалізації дана концепція не одержала успішного просування.

Сьогодні швидкий розвиток NFV визначається декількома факторами: стрімким зростанням популярності віртуалізації в IT, потребою поширити ідеї віртуалізації на область мережних функцій і можливістю реалізації останніх за рахунок удосконалювання елементної бази й інструментальних засобів. Нарешті, для реалізації NFV доступна інфраструктура хмарних обчислень і ЦОД.

По суті, NFV – це спроба операторів позбутися від твердої прив'язки до конкретного виробника мережного устаткування, оскільки вони не хочуть залежати від одного постачальника. У традиційних мережах на кожній площадці встановлюється фрагментоване спеціалізоване устаткування. Необхідність у дорогій розробці нових апаратних рішень утрудняє вихід на даний ринок нових гравців, перешкоджає інноваціям і конкуренції. NFV припускає застосування стандартних серверів і комутаторів, віртуальних пристроїв незалежних постачальників і програмних інструментів керування/оркестрації. Таким чином, цього разу ініціаторами стандартизації виступають не вендори, а оператори.

По суті, тільки-но з'явившись, нова концепція NFV уже впливає на стратегію великих операторів, що свідчить про гостру необхідність у ній. Менш чим за рік вийшло п'ять специфікацій NFV, і зараз готується нове видання. У випущеному в середині жовтня документі «NFV White Paper 3.0» підсумується вміст підготовлюваних документів NFV Industry Specification Group (ISG), розповідається про перспективи стандартизації NFV, підкреслюється важливість розвитку систем керування мережами відповідно до динаміки й гнучкістю NFV, створення навчальних курсів і проведення досліджень (http://portal/etsi/org/NFV/NFV_White_Parer3.pdf).

Проект документа «NFV ISG Second Release» (його остаточний варіант повинен бути опублікований у січні 2015 року) містить у собі опис загальної

мережної архітектури («Architectural Framework Rev 2»), інфраструктури (огляд використання обчислювального, мережного доменів і домена гіпервізора із прикладами), інтерфейсів і рівнів абстрагування, метрик якості сервісу, переносимості й тиражуємості, керування й оркестрації. Крім того, у ньому розглядаються питання продуктивності й надійності, безпеки й захисту, а також архітектура віртуальних мережних функцій (http://docbox.etsi.org./ISG/NFV/Open/).

До потенційних переваг NFV ставляться наступні можливості: швидке розгортання, модернізація й відключення мережних функцій; гнучке комбінування декількох мережних функцій на одній серверній платформі (що особливо важливо в багатоарендному середовищу); розміщення мережних функцій там, де це ефективніше або економічніше; скорочення витрат завдяки використанню стандартних серверів замість пропрієтарного мережного устаткування.

Крім того, NFV забезпечує швидке масштабування сервісів, можливість використання ПЗ для одержання прибутку, застосування недорогих стандартних серверів замість мережного устаткування, підвищення операційної ефективності, оптимізацію мережі в реальному часі, економію електроенергії за рахунок консолідації навантаження. Нові мережні функції (Virtual Network Function, VNF) можуть розробляти невеликі компанії, а це сприяє інноваціям.

NFV допомагає скоротити операційні витрати, але сама по собі не спрощує операції – це досягається лише в сполученні з автоматизацією й SDN. Проте її основне достоїнство – не економія витрат, а гнучкість. В NFV функціональність мережі реалізується за допомогою стандартних технологій ЦОД і переноситься на рівень ПЗ, що дозволяє позбутися від інерції апаратних процедур. Крім того, спрощується реалізація відказостійких схем і прискорюється впровадження інновацій у мережах і сервісах.

По суті, NFV – це методологія, новий підхід, причому не тільки технологічний, але й організаційний. Він припускає не просто заміну існуючого

мережного устаткування на стандартні сервери, а звертання до набагато більше складних процедур і процесів. Щоб зважитися на впровадження даної методології, оператор повинен по-новому вибудувати свою діяльність, та й сам перехід від її лабораторного тестування до реального використання й одержання прибутку від впровадження залишається нетривіальним завданням. При міграції перехід від існуючої мережної інфраструктури до NFV може представляти значні труднощі, як і зміни в системах керування, OSS/BSS.

Тим часом для традиційних операторів впровадження NFV – питання виживання. Велика кількість вузькоспеціалізованого й дорогого в обслуговуванні устаткування заважає їм швидко впроваджувати й монетизувати інноваційні послуги із прийнятними капітальними й експлуатаційними витратами, утрудняючи конкуренцію із провайдерами послуг. Реалізація NFV могла б запобігти перетіканню доходів до нових сервісів-провайдерів, що розгортають свої послуги «поверх» операторської інфраструктури. Останні зацікавлені в співробітництві із традиційними операторами, оскільки публічний Інтернет не гарантує тої якості сервісу, яких необхідно для надання деяких видів послуг корпоративного рівня.

Один із самих більших бар'єрів на шляху до NFV – організаційна структура. Всі питання, завдання й проблеми повинні розглядатися й вирішуватися в тісній взаємодії з операторами.

Керування й оркестрація

Модель архітектури NFV поки тільки обговорюється. Галузева організація OPNFV (http://www.opnfv.org) має намір взаємодіяти з Європейським інститутом телекомунікаційних стандартів (European Telecommunications Standards Institute, ETSI) і використовувати елементи Open Source. Споконвічно області інтересу OPNFV охоплювали апаратні платформи (обчислювальні ресурси, СХД і мережне устаткування), шар віртуалізації, віртуальні обчислення, мережі й ресурси зберігання даних, а також засобу керування віртуалізованою інфраструктурою. Однак більше важливе значення можуть мати оркестрація й керування більше високого рівня (Management and Orchestration, MANO). Багато

операторів не квапляться із впровадженням NFV саме тому, що ряд питань, пов'язаних з MANO, усе ще залишаються відкритими.

Експерти підтверджують, що дотепер незрозуміло, яка оркестрація буде потрібно для VNF у мережах і як будуть розділятися функції керування між різними рівнями архітектури NFV. Без цього оператори, що розгортають NFV, будуть змушені використовувати пропоновані вендорами пропрієтарні рішення.

Класична мережна модель передбачає три «площини»: дані, контроль і керування. Вся ідея SDN полягає в тому, щоб змінити пропорції між двома останніми – площинами керування більше низького й високого рівня. Усе більше функцій будуть переходити на рівень контролю й ставати програмувальними й автоматизируемими. Ціль – не в конфігуруванні або програмуванні мереж, а в програмуванні сервісів із залученням автоматизації.

При цьому існують два основних підходи. Відповідно до першого, протоколи площини контролю заміняються на API у центральному контролері SDN, а механізми форвардинга в площині даних зводяться до простих операцій у комутаторі SDN. Це припускає заміну існуючого устаткування. При другому підході SDN доповнює існуючі технології: для оптимізації продуктивності вводяться додаткові операції керування, а для створення нових сервісів використовуються програмні засоби, у результаті існуюча мережа стає більше програмувальною. Більшість операторів і вендорів устаткування підтримують перший підхід, а другий застосовується в дослідницьких організаціях.

Розподілена модель і vCPE

Найчастіше функції NFV розміщаються централізовано – у ЦОД або на мережних вузлах (у крапках присутності оператора – POP), що, відповідно до документа «NFV Whitepaper», зовсім не обов'язково. Розподілена модель (Distributed NFV, D-NFV) дозволяє розміщати їх у будь-якому місці, контрольованому сервісом-провайдером. У тому числі й на площадці клієнта – у мережних пристроях, службовців крапкою демаркації мережі оператора й користувача послуг.

У традиційній моделі розгортання нових сервісів нерідко вимагає установки додаткового устаткування в клієнта (Customer Premises Equipment, CPE), а це – «головний біль» для оператора. Віртуалізація мережних функцій припускає перенос функціональності в крапку присутності (Point of Presence, POP) або в ЦОД оператора. Специфічні пристрої різних вендорів з їхніми функціями (оптимізація WAN, міжмережне екранування й ін.) заміняються на пристрої CPE наступного покоління – vCPE, що працюють у взаємодії з віртуалізованими функціями в мережі. vCPE поєднує дві сутності – клієнтський пристрій і функціональність у мережі оператора.

Індивідуальні пристрої CPE заміняються на «віртуальне» мережне рішення, використовуване декількома клієнтами (multi-tenant), причому для реалізації vCPE необов'язково навіть впроваджувати віртуалізацію (саме такими були перші версії) – віртуальність у цьому випадку означає поділ ресурсів між декількома клієнтами.

У сучасному розумінні NVF віртуалізація означає перенос функцій у мережу. NFV значно спрощує реалізацію клієнтського устаткування. На площадці клієнта залишаються лише деякі функції, «не стерпні» у мережу провайдеру, – наприклад, шифрування трафіку, оптимізація WAN, контроль QoS. Крім того, мережні функції розділяються на дві більші категорії: функції, відповідальні за роботу мережі, зокрема моніторинг продуктивності, і додаткові сервіси (Value-Added Services, VAS), такі як Vol, FW, маршрутизація в клієнта та ін. Останні нерідко має сенс реалізувати на стороні клієнта. При розподілі функцій між CPE і устаткуванням у крапці присутності або в ЦОД операторам доводиться враховувати вимоги регуляторів і економічні міркування. Інакше кажучи, функції VNF розміщаються там, де це найбільш вигідно й оптимально.

Основні драйвери впровадження vCPE – перспектива прискореного розгортання нових сервісів, реалізація нових можливостей без заміни устаткування в клієнта, що випливає із цього скорочення капітальних і операційних витрат, більше просте керування CPE і об'єднання в одному

пристрої відразу декількох функцій. А SDN повинна дозволити гнучко поєднувати кілька елементарних функцій у групи.

vCPE – підхід, що розвивається, що передбачає різні варіанти реалізації. Тим часом дотепер немає єдиного визначення vCPE, а ETSI і Broadband Forum продовжують роботу над відповідними стандартами. Можливі три моделі: просте фізичне устаткування CPE у клієнта (pCPE) і централізована віртуальна частина в провайдеру (vCPE); розподіл функцій між CPE і ЦОД провайдеру; перенос до клієнта й фізичної, і віртуальної складової (pvCPE). CPE може бути реалізоване по модульному принципі з вибором необхідної функціональності або як єдиний інтегрований пристрій із заданою функціональністю. pCPE і vCPE можуть бути слабко або тісно зв'язаними – наприклад, взаємодіяти по якімсь протоколі керування. Це відкриває різні можливості для «програмуємості» і автоматизації.

Якщо деякі мережні функції має сенс віртуалізовать на площадці клієнта, то логічно встановити там стандартний сервер і запустити на ньому відповідні віртуальні машини (ВМ). Однак, оскільки через такий сервер буде проходити весь трафік, він повинен бути досить потужним. Для цієї мети RAD розробила програмно-апаратне рішення. Ряд функцій (Forwarding, Traffic Shaping і ін.) їсти зміст вивести на апаратний рівень, оскільки їхня програмна реалізація виявляється дорожче або веде до зниження енергоефективності. На програмний рівень (ВМ) можна винести функції маршрутизації, діагностики й тестування, а у вигляді додатків реалізувати міжмережне екранування, шифрування, оптимізацію WAN, IP-телефонію. Архітектурний підхід vCPE дозволяє легко модернізувати клієнтські системи. Програмна функціональність VNF може наращуватися за допомогою додатків з магазина D-NFV App Store, розроблювальних різними вендорами.

Архітектура мережного інтерфейсного пристрою в моделі D-NFV може містити в собі устаткування NTU (як частина мережної інфраструктури), гіпервізор KVM і додатка у ВМ, такі як міжмережний екран, причому ВМ можуть обробляти лише частина трафіку, наприклад окремі VLAN. Керування такою

системою здійснюється з мережі провайдеру – оркестрація охоплює не тільки функціональність клієнтського устаткування, але й конфігурування мережі. VNF у ВМ можуть бути постійно працюючими або запускатися якщо буде потреба – при настанні певних подій або для тестування й виявлення проблем у мережі. Такий варіант називається децентралізованим (Centerless D-NFV), але частина функцій VNF може бути перенесена в мережні вузли або в ЦОД провайдеру.

Ключові переваги D-NFV – гнучке розгортання сервісів і контрольовані витрати. Експериментальна експлуатація дає операторові можливість випробувати новий підхід і проаналізувати його придатність. NFV і SDN досить важко обґрунтувати економічно, тому розгортання NFV на площадці клієнта дозволяє почати з «низького старту», впровадити нові рішення з мінімальними початковими інвестиціями й відмовитися від них у випадку невдачі без особливого збитку. Нарешті, у цьому випадку не потрібна заміна OSS/BSS.

Тема NFV не випадково стала однією з найбільш обговорюваних на найбільших галузевих форумах, адже завдяки віртуалізації мережних функцій значно міняються традиційні методи проектування й керування корпоративними мережами, а також мережною інфраструктурою телекомунікаційних компаній.

3.2 Розробка структурної схеми

Програмно конфігуруємі мережі можуть стати основою сучасних хмарних ЦОД і корпоративних мереж, причому віртуалізація мережної інфраструктури дозволяє одержати динамічне мережне середовище, що поєднує в собі існуюче мережне устаткування й нові компоненти. Вендори вже пропонують для цього комерційні продукти.

Поширення концепцій програмно конфігуруємих мереж (Software-Defined Networking, SDN) і віртуалізації мережних функцій (Network Functions Virtualization, NFV) веде до трансформації мережних інфраструктур і принципів створення, розвитку й експлуатації мереж. Вендори вже пропонують архітектури,

рішення й інструменти, які дозволяють наділити мережі необхідними властивостями для відповідності новим вимогам. Як очікується, перехід до програмно конфігуруємим мереж дозволить мережній інфраструктурі ефективніше підтримувати виконання завдань, що коштують перед IT. По даним Infonetics Research, торік ринок рішень SDN для ЦОД і корпоративних мереж (включаючи контролери й комутатори) виріс більш ніж на 190%. Згідно із прогнозами, до 2025 року його обсяг досягне 18 млрд доларів.

Перехід до архітектури програмно обумовлених мереж нерідко розглядається як ефективний спосіб рятування від існуючих у мережній галузі проблем.

SDN і NFV: динамічна мережна інфраструктура

SDN і NFV – стратегічні напрямки NEC. При розробці відповідних рішень компанія концентрує зусилля в чотирьох областях. Це інфраструктура SDN для ЦОД і корпоративних мереж, віртуалізація мережних функцій, SDN для транспортних мереж, а також керування й оркестрація SDN. Поділ телекомунікаційної інфраструктури на фізичну й логічну (віртуалізовану) складові пред'являє безліч вимог до їх правильного оркестрації: вона повинна дозволяти досить легко й динамічно інтегрувати додатка сторонніх постачальників у мережі підприємств. Тому в області додатків SDN компанія NEC не тільки концентрується на інфраструктурних завданнях, але й співробітничає з іншими вендорами.

SDN припускає перегляд мережної архітектури з поділом керування, комутації й передачі даних – перехід від автономних розподілених систем, де кожний мережний пристрій має свій «інтелект» і конфігурацію й здійснює передачу даних відповідно до закладеного в ньому логікою й протоколами, до мереж з поділом передачі даних (Data Plane) і керування (Control Plane). У таких мережах всі завдання, пов'язані з конфігуруванням устаткування, маршрутизацією трафіку й мережними сервісами, винесені на рівень керування – рівень контролера SDN і мережної операційної системи.

Контролер забезпечує автоматичне адміністрування мережних пристроїв, при цьому оператор зв'язку має можливість створювати власне ПЗ керування мережею, а відкриті програмні інтерфейси (API) дозволяють впроваджувати мережні додатки сторонніх вендорів. Крім того, контролер забезпечує зв'язок між рівнем додатків і мережею. Наприклад, він може передати додатку інформацію про трафік, після чого додаток за допомогою контролера змінює конфігурацію мережі. Таким чином, мережна інфраструктура стає динамічною.

При впровадженні SDN необхідно правильно спланувати мережа, з урахуванням її архітектури й функціональності мережних елементів. Реалізація SDN кожним виробником має свої особливості. Існують складності інтеграції з наявними системами керування, розширення/зміни мережної інфраструктури й реалізації нових сценаріїв.

Рисунок 3.1 – Структурна схема системи

З SDN тісно зв'язане інший напрямок – віртуалізація мережних функцій. NFV означає перегляд архітектури компонентів мережі і їх віртуалізацію й припускає заміну традиційного спеціалізованого мережного устаткування на віртуальні пристрої – віртуальні машини (ВМ) з відповідною функціональністю,

що працюють у середовищі віртуалізації на стандартних серверах (NEC застосовує для цього гіпервізор «операторського класу» на базі KVM).

Оператор може створювати такі ВМ на обчислювальних потужностях у будь-якій крапці мережі: у своєму ЦОД, на вузлі зв'язку або на площадці користувача. Таким чином, NFV скасовує прив'язку мережних елементів до певного устаткування й місця, а крім того, дозволяє розподіляти функціональність (Distributed NFV), тим самим забезпечуючи більше ефективне використання ресурсів і підвищуючи надійність.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

При побудові моделі (без апріорної прив'язки до «організації») дало можливість зв'язати її блоки на різних рівнях декомпозиції з об'єктами організаційно-технічної структури, що виступають як механізми. У цьому випадку, і це методично надто важливо, організаційно-технічна структура стає результатом функціонального моделювання. Саме цього вдалося досягти при декомпозиції механізмів впливу на систему керування навантаженням на вузли сервісу віртуалізацій мережних функцій NFV.

Створена функціональна модель мережі сервісу віртуалізацій мережних функцій NFV дозволяє вирішити наступні завдання:

– сформувати наочне візуальне подання взаємодії основних процесів, що відбуваються в мережі сервісу віртуалізацій мережних функцій NFV;

– зробити чітке визначення ролі системи поліпшення функціонування в загальній системі сервісу віртуалізацій мережних функцій NFV;

– точно визначити вхідні, вихідні й керуючі впливи на підсистему балансування навантаження, що надалі дозволяє провести аналітичне й імітаційне моделювання інформаційного потоку;

– сценарій балансування навантаження, розроблений у ході виконання магістерської роботи, є основою алгоритму імітаційного моделювання.

Функціональна схема розробленої системи зображена на рисунку 3.2.

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

– Блок розрахунку вектору значень інтегрального критерію оптимізації.

– Блок ініціалізації граничних значень параметрів моделювання.

– Блок формування матриці балансування навантаження.

– Блок формування матриці транзитних вузлів.

– Блок формування сигналів для елементів мережі.

– Згенеровані звіти по поточному завантаженню елементів мережі.

Рисунок 3.2 – Функціональна схема системи

Інтернет-технології прийняли широке поширення, і використовуються як основа побудови корпоративних і критично важливих додатків, таких як WEB-вузли, вузли потокового мультимедіа-віщання й сервери віртуальних приватних мереж. Служба розподілу навантаження мережі є оптимальним і ефективним рішенням, що забезпечує масштабованість і високу відказостійкість таких додатків як в Інтернеті, так і в інтрамережах.

Служба балансування навантаження мережі (БНМ) дозволяє системним адміністраторам створювати кластери, що включають до 32 вузлів, між якими будуть розподілятися вступні від клієнтів запити. При цьому з погляду клієнтів кластер нічим не відрізняється від звичайного сервера; серверні додатки також не вимагають адаптації для роботі в кластері.

Служба БНМ постачена всіма необхідним адміністраторам способами керування, у тому числі можливістю (після уведення пароля) віддалено управляти кластером з будь-якого комп'ютера в мережі. Крім того, адміністратори мають можливість набудовувати кластер під спеціальні завдання, управляючи потоком даних на рівні портів. Вузли додаються й виключаються із кластера без припинення обслуговування. Крім того, програмне забезпечення на вузлах кластера можна обновляти без припинення обробки клієнтських запитів.

Служба БНМ використовує для розподілу навантаження між вузлами повністю розподілений алгоритм. На відміну від рішень на основі диспетчеризації така архітектура забезпечує високу продуктивність і низькі накладні витрати ресурсів на розподіл потоку запитів від клієнтів. Крім того, для цієї архітектури характерна висока відказостійкість (N-1) при числі вузлів N. Всі ці характеристики досягаються без необхідності використовувати спеціальні апаратні або програмні рішення.

Служба БНМ періодично розсилає ритмічні повідомлення, призначені для інформування кожного члена кластера про наявність інших вузлів. Збій будь-якого вузла виявляється протягом п'яти секунд, а відновлення обслуговування клієнтів виконується протягом десяти секунд. Як при відключенні працюючого

вузла, так і при додаванні нового вузла в кластер навантаження автоматично й прозоро перерозподіляється між членами кластера.

Тести продуктивності демонструють, що використання програмної служби БНМ дає низькі накладні витрати на обробку потоку даних і чудові можливості масштабування продуктивності, обмежені тільки пропускною здатністю підмережі. Служба БНМ демонструє пропускну здатність більше 200 Мбіт/с у реальних рішеннях по обслуговуванню електронної торгівлі з більш ніж 800 млн. запитів протягом дня.

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. 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

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

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

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

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції.

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

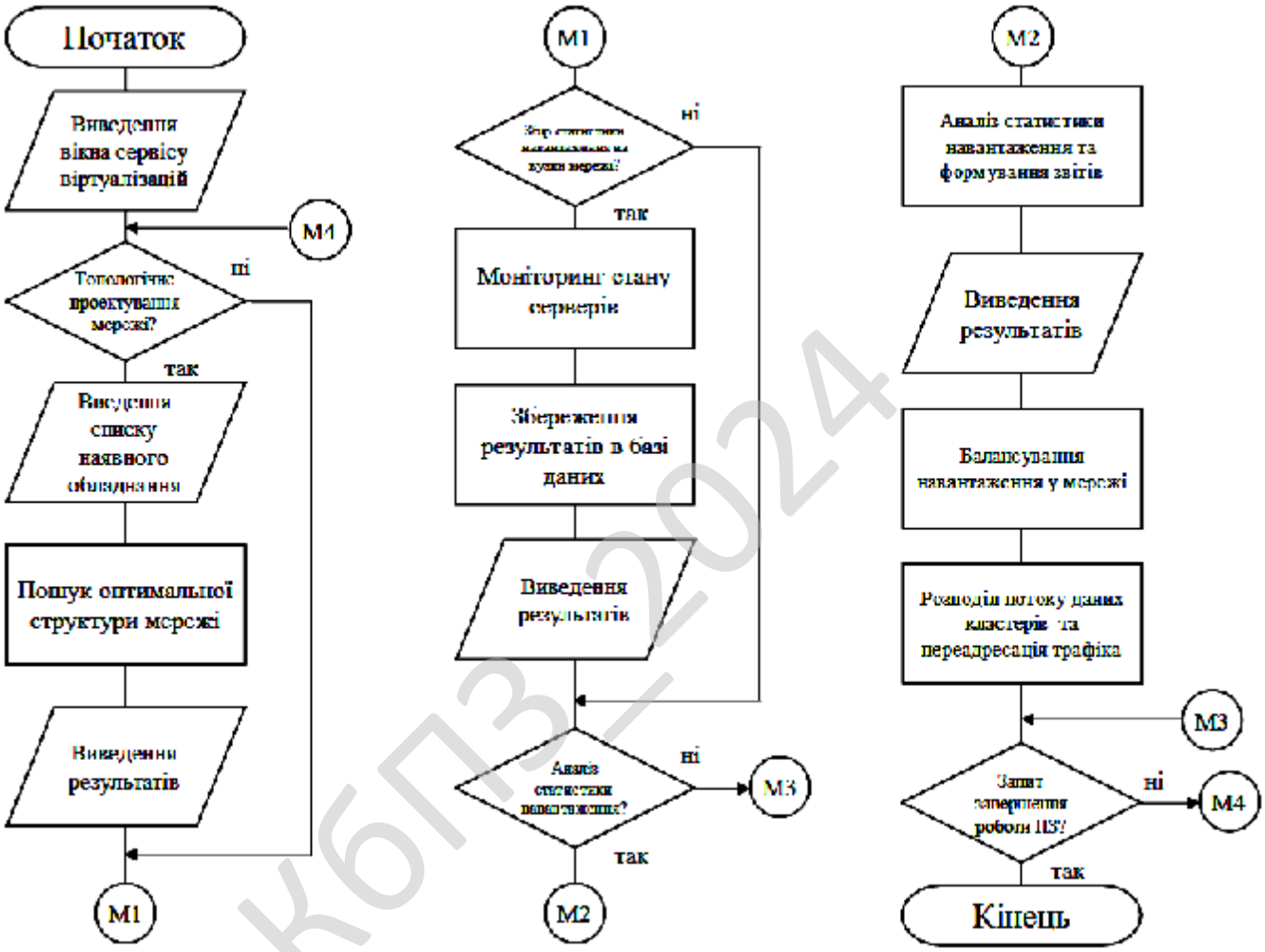

Рисунок 4.1 – Блок-схема основної програми

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

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

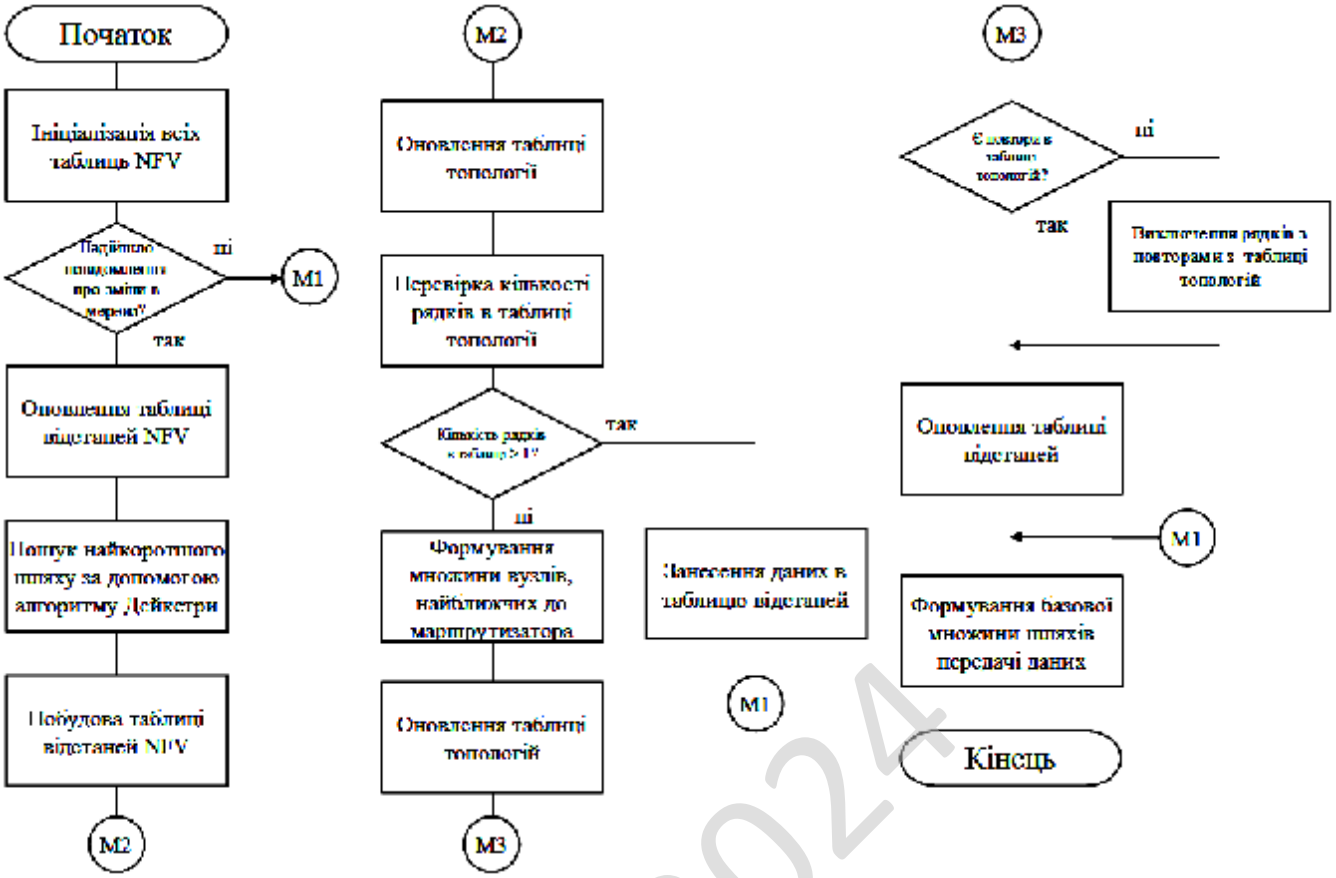

Рисунок 4.2 – Блок-схема роботи підпрограми

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

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу).

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій.Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії

мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Наведемо частину коду, який реалізує вище перераховані дії.

```
procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
 WorkgroupNode, ComputerNode: TTreeNode;
begin
 Screen.Cursor:=crHourGlass;
 try
// Ініціалізація об’єктів та сканування мережі
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп’ ютерів в мережі
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);
// Отримання списку всіх робочих груп і комп’ютерів
// в мережі, відсортованих в алфавітному порядку
   List:=TStringList.Create;
   ListView1.Items.Clear;
   try
```

```
    N.ListNetwork(List);
    for i:=0 to List.Count - 1 do begin
     ListViewItem:=ListView1.Items.Add;
     ListViewItem.Caption:=List[i];
     ListViewItem.ImageIndex:=Integer(List.Objects[i]);
    end;
   finally
    List.Free;
   end;
// Побудова дерева робочих груп і комп' ютерів в мережі
   TreeView1.Items.Clear;
   for i:=0 to N.Count - 1 do begin
    WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
    WorkgroupNode.ImageIndex:=1;
    WorkgroupNode.SelectedIndex:=1;
    for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
     ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode,
                  (N.Objects[i] as TStrings).Strings[j]);
     ComputerNode.ImageIndex:=0;
    end;
   end;
// Отримання IP адрес комп'ютерів
  GetIPAddresses(N, Memo2.Lines);
  finally
   N.Free;
  end;
  TreeView1.FullExpand;
 finally
  Screen.Cursor:=crDefault;
end;
end;
```

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

```
const
  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
    ( Prt: 0; Srv:   ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:   ' ECHO   '  ),     {Пінгування    }
    ( Prt: 9; Srv:   ' DISCARD'  ),
    ( Prt: 13; Srv:  ' DAYTIME'  ),
    ( Prt: 17; Srv:  ' QOTD   '  ),     {Показчик на день}
```

```
    ( Prt: 19; Srv:  ' CHARGEN'  ),{Генератор символів}
    ( Prt: 20; Srv:  ' FTPDATA'  ),{ Протокол File Transfer Protocol – дані}
    ( Prt: 21; Srv:  ' FTPCTRL'  ),{ Протокол File Transfer Protocol – управління}
    ( Prt: 22; Srv:  ' SSH    '  ),
    ( Prt: 23; Srv:  ' TELNET '  ),
    ( Prt: 25; Srv:  ' SMTP   '  ),     { Протокол Simple Mail Transfer Protocol}
    ( Prt: 37; Srv:  ' TIME   '  ),     { Протокол Time Protocol }
    ( Prt: 43; Srv:  ' WHOIS  '  ),     { WHO IS сервіс  }
    ( Prt: 53; Srv:  ' DNS    '  ),     { Domain Name Service }
    ( Prt: 67; Srv:  ' BOOTPS '  ),     { BOOTP сервер }
    ( Prt: 68; Srv:  ' BOOTPC '  ),     { BOOTP клієнт }
    ( Prt: 69; Srv:  ' TFTP   '  ),     { Стандартний  FTP  }
    ( Prt: 70; Srv:  ' GOPHER '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:  ' FINGER '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:  ' HTTP   '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:  ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv: ' POP2   '  ),{ Протокол Post Office Protocol Version 2 }
    ( Prt: 110; Srv: ' POP3   '  ),{ Протокол Post Office Protocol Version 3 }
    ( Prt: 111; Srv: ' SUN_RPC'  ),     { SUN віддалений виклик функцій }
    ( Prt: 119; Srv: ' NNTP   '  ),{ Протокол Network News Transfer Protocol }
    ( Prt: 123; Srv: ' NTP    '  ),{ Протокол Network Time protocol          }
    ( Prt: 135; Srv: ' DCOMRPC'  ),     { Локальний сервіce              }
    ( Prt: 137; Srv: ' NBNAME '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv: ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv: ' NBSESS '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv: ' IMAP   '  ),{ Протокол Internet Message Access Protocol }
    ( Prt: 161; Srv: ' SNMP   '  ),{ Протокол Simple Netw. Management Protocol }
    ( Prt: 169; Srv: ' SEND   '  )
   );
  const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
(' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , ' IP_DEST_HOST_UNREACHABLE'
,   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,   ' IP_BAD_OPTION' ,' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );
  ARPEntryType  : array[1..4] of string = ( 'Інший' , 'Несправний' ,
```

```
    ' Динамічний' , ' Статичний'
    );
  TCPConnState  :
    array[1..12] of string =
 ( ' CLOSED' , ' READ' , ' SYN_SENT' ,' SYN_RCVD' , ' ESTABLISHED' , ' FIN_WAIT1',
' FIN_WAIT2' , ' WAIT' , ' CLOSING' , ' LAST_ACK' , ' WAIT' , ' DELETE_TCB'    );
  TCPToAlgo     : array[1..4] of string =
( ' Const.Timeout' , ' MIL-STD-1778' , '  ' , ' Other'  );
  IPForwTypes   : array[1..4] of string =
    ( ' other' , ' invalid' , ' local' , ' remote'  );
  IPForwProtos  : array[1..18] of string =
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );
type
// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;
  TIfRows = array of TMibIfRow ; // динамічний масив колонок
```


4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою IDEA – симетричний блоковий алгоритм шифрування даних, запатентований швейцарською фірмою Ascom. Відомий тим, що застосовувався в пакеті програм шифрування PGP. У листопаді 2000 року IDEA був представлений

як кандидат у проекті NESSIE в рамках програми Європейської комісії IST (англ. Information Societes Technology, інформаційні громадські технології).

Першу версію алгоритму розробили в 1990 році Лай Сюецзя (Xuejia Lai) і Джеймс Мессі (James Massey) зі Швейцарського інституту ETH Zürich (за контрактом з Hasler Foundation, яка пізніше влилася в Ascom-Tech AG) як заміна DES (англ. Data Encryption Standard, стандарт шифрування даних) і назвали її PES (англ. Proposed Encryption Standard, запропонований стандарт шифрування). Потім, після публікації робіт Біхамом і Шаміра по диференціальному криптоаналізу PES, алгоритм був поліпшений з метою посилення криптостійкості і названий IPES (англ. Improved Proposed Encryption Standard, покращений запропонований стандарт шифрування). Через рік його перейменували в IDEA (англ. International Data Encryption Algorythm).

Так як IDEA використовує 128-бітний ключ і 64-бітний розмір блоку, відкритий текст розбивається на блоки по 64 біт. Якщо таке розбиття неможливо, останній блок доповнюється різними способами певною послідовністю біт. Для уникнення витоку інформації про кожному окремому блоці використовуються різні режими шифрування. Кожен вихідний незашифрований 64 – біт ний блок ділиться на чотири підблока по 16 біт кожен, так як всі алгебраїчні операції, що використовуються в процесі шифрування, відбуваються над 16-бітними числами. Для шифрування і розшифрування IDEA використовує один і той же алгоритм.

Позначення операцій:

- $\boxplus$ Додавання за модулем 2^{16}.
- $\odot$ Множення за модулем $2^{16}+1$.
- $\oplus$ Побітова виключна диз'юнкція.

Фундаментальним нововведенням в алгоритмі є використання операцій з різних алгебраїчних груп, а саме:

Додавання за модулем 2^{16}.

Множення за модулем $2^{16}+1$.

Побітова виключна диз'юнкція (XOR).

Ці три операції несумісні в тому сенсі, що ніякі дві з них не задовольняють дистрибутивному закону, тобто:

$$a \odot (b \oplus c) \neq (a \odot b) \oplus (a \odot c).$$

Застосування цих трьох операцій ускладнює криптоаналіз IDEA в порівнянні з DES, який базується виключно на операції виключає АБО, а також дозволяє відмовитися від використання S-блоків і таблиць заміни. IDEA є модифікацією мережі Фейстеля.

Генерація ключів

З 128-бітного ключа для кожного з восьми раундів шифрування генерується по шість 16-бітних подключів, а для вихідного перетворення генерується чотири 16-бітних підключа. Всього буде потрібно 52 = 8 x 6 + 4 різних подключів по 16 біт кожен. Процес генерації п'ятдесяти двох 16-бітних ключів полягає в наступному:

Насамперед, 128-бітний ключ розбивається на вісім 16-бітних блоків. Це будуть перші вісім подключів по 16 біт кожен – $(K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)})$

Потім цей 128-бітний ключ циклічно зсувається вліво на 25 позицій, після чого новий 128-бітний блок знову розбивається на вісім 16-бітних блоків. Це вже наступні вісім подключів по 16 біт кожен – $(K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)})$

Процедура циклічного зсуву і розбивки на блоки триває до тих пір, поки не будуть згенеровані всі 52 16-бітних підключа.

Шифрування

Структура алгоритму IDEA показана на рисунку 4.3.

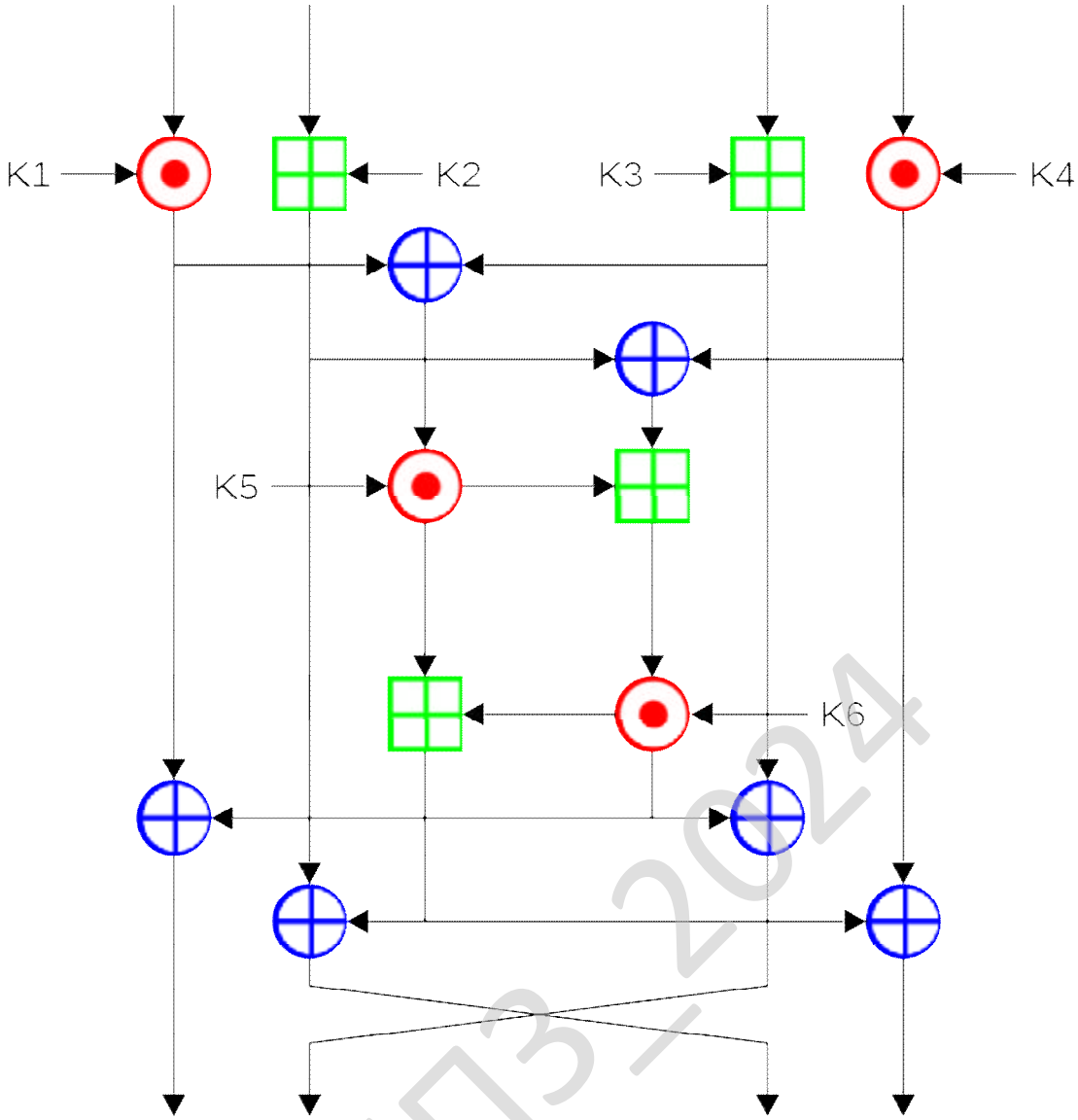

Рисунок 4.3 – Структура алгоритму IDEA

Процес шифрування складається з восьми однакових раундів шифрування і одного вихідного перетворення. Вихідний незашифрований текст ділиться на блоки по 64 біта. Кожен такий блок ділиться на чотири підблока по 16 біт кожен. На рисунку ці підблоки позначені D_1, D_2, D_3, D_4. У кожному раунді використовуються свої підключі згідно з таблицею підключів. Над 16-бітними підключами і підблока незашифрованого тексту проводяться наступні операції:

– Множення за модулем $2^{16}+1 = 65537$, причому замість нуля використовується 2^{16}.

– Додавання за модулем 2^{16}.

– Побітове виключне АБО.

В кінці кожного раунду шифрування є чотири 16-бітних підблоки, які потім використовуються як вхідні підблоки для наступного раунду шифрування. Вихідна перетворення являє собою скорочений раунд, а саме, чотири 16-бітних підблоки на виході восьмого раунду і чотири відповідних підключа піддаються операціям:

– Множення за модулем $2^{16}+1$.

– Додавання за модулем 2^{16}.

Після виконання вихідного перетворення конкатенація підблоків D_1', D_2', D_3' і D_4' являє собою зашифрований текст. Потім береться наступний 64-бітний блок незашифрованого тексту і алгоритм шифрування повторюється. Так продовжується до тих пір, поки не зашифрують всі 64-бітові блоки вихідного тексту.

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 сервісу віртуалізацій мережних функцій NFV складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Кластер; Хост; Параметри; Довідка.

– Розділ виведення сервісу віртуалізацій мережних функцій NFV.

– Розділу поточної інформації.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

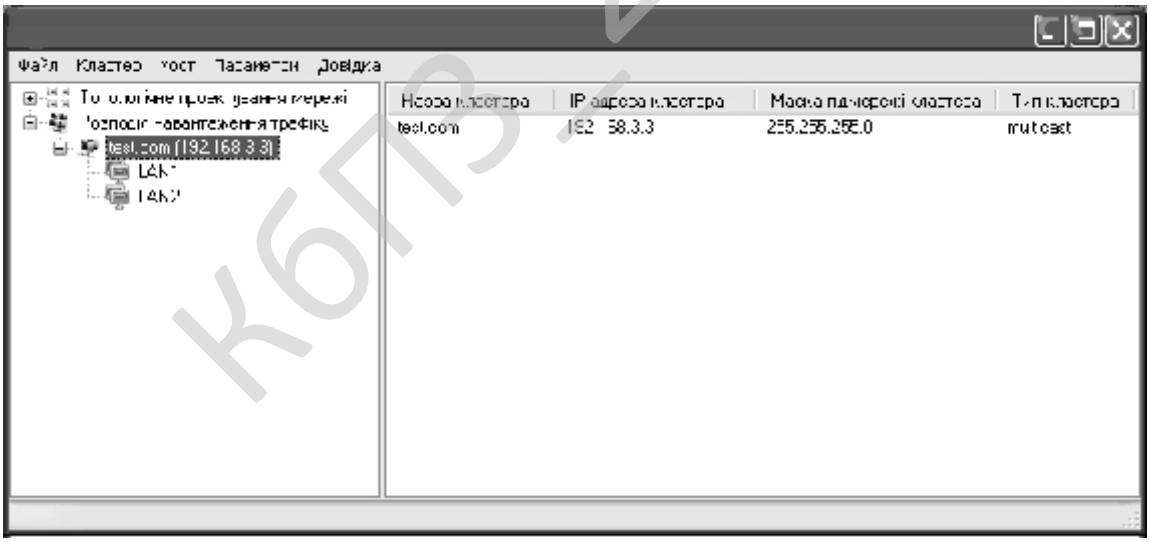

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

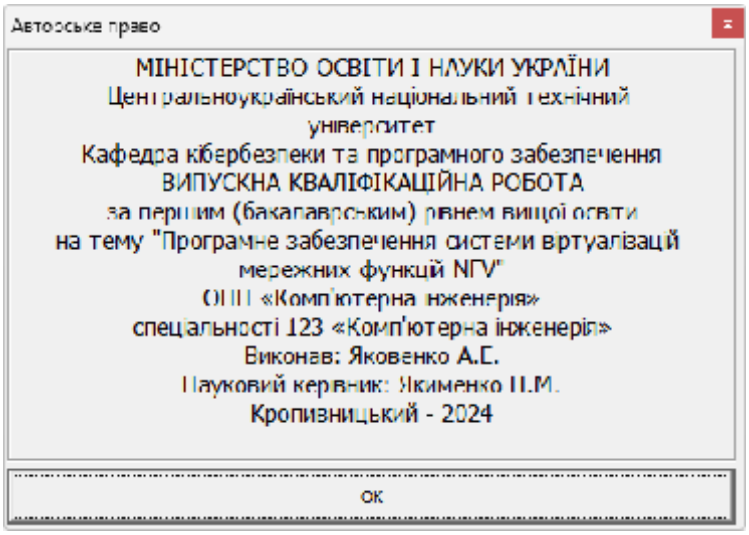

Рисунок 5.2 – Вікно розробника ПЗ

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 чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ІТс;

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій;

– Помилок інтерфейсу;

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних;

– Помилок характеристик (необхідна ємність пам'яті і т.д.);

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів.

Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення.

Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються.

Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи віртуалізацій мережних функцій NFV.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем віртуалізацій мережних функцій NFV.

– Досліджена система віртуалізацій мережних функцій NFV.

– На основі отриманих результатів досліджень створена програмна реалізація системи віртуалізацій мережних функцій NFV.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання віртуалізацій мережних функцій NFV.

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

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

Програма реалізована на мові високого рівня Delphi 10.4 Sydney. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи віртуалізацій мережних функцій NFV. 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 IDEA.

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

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

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

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

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

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

7. 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.

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

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

14. 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.

15. 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.

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

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

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

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

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

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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

37. 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.

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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).

44. 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

45. 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.

46. 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.

47. 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.

48. 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.

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

50. 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.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи віртуалізацій мережних функцій NFV.

2 Підстава для розробки

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

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи віртуалізацій мережних функцій NFV.

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 віртуалізацій мережних функцій NFV;

– 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 Sydney.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 76 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 віртуалізацій мережних функцій NFV

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 39

Літера: РП

Кропивницький – 2024 року

Основна програма

Файл Main.pas основної програми

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Dim, Networks, StdCtrls, ComCtrls, ImgList, ShellAPI, ShlObj, IpHlpApi,
IPtraff,
  ExtCtrls, About, Buttons;

type
  TForm1 = class(TForm)
    Memo1: TMemo;
    ClickMe: TButton;
    TreeView1: TTreeView;
    ImageList1: TImageList;
    ListView1: TListView;
    Memo2: TMemo;
    Button1: TButton;
    Label1: TLabel;
    Edit1: TEdit;
    Memo3: TMemo;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    MemoTR: TMemo;
    Timer1: TTimer;
    Label7: TLabel;
    Button2: TButton;
    Button3: TButton;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
    SpeedButton3: TSpeedButton;
    SpeedButton4: TSpeedButton;
    SpeedButton5: TSpeedButton;
    SpeedButton6: TSpeedButton;
    MemoX: TMemo;
    Label8: TLabel;
    procedure ClickMeClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure SpeedButton2Click(Sender: TObject);
    procedure SpeedButton4Click(Sender: TObject);

  private
    { Private declarations }
     procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation
```

```
{$R *.DFM}

function GetShellFolder(CompName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=CompName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo3.Lines.Clear;
 Screen.Cursor:=crHourGlass;
 try
  if not EnumSharedResources(Edit1.Text, Memo3.Lines)
   then raise Exception.Create(‘ Невірне ім"я комп"ютера’ );
 finally
  Screen.Cursor:=crDefault;
 end;
end;



procedure TForm1.ClickMeClick(Sender: TObject);
var
 N: TNetworkNeighborhood;
 List: TStringList;
 i, j: Integer;
 ListViewItem: TListItem;
 WorkgroupNode, ComputerNode: TTreeNode;
begin
 Screen.Cursor:=crHourGlass;
 try
  // Ініціалізація об’ єктів та сканування мережі
  N:=TNetworkNeighborhood.Create;
  try
   // Отримання списку всіх комп’ ютерів в мережі
   Memo1.Lines.Clear;
   N.ListComputers(Memo1.Lines);

   // Отримання списку всіх робочих груп і комп’ ютерів в мережі, відсортованих
в алфавітному порядку
   List:=TStringList.Create;
   ListView1.Items.Clear;
   try
    N.ListNetwork(List);
    for i:=0 to List.Count - 1 do begin
     ListViewItem:=ListView1.Items.Add;
     ListViewItem.Caption:=List[i];
     ListViewItem.ImageIndex:=Integer(List.Objects[i]);
    end;
   finally
    List.Free;
   end;
```

```
   // Побудова дерева робочих груп і комп’ ютерів в мережі
   TreeView1.Items.Clear;
   for i:=0 to N.Count - 1 do begin
    WorkgroupNode:=TreeView1.Items.Add(nil, N[i]);
    WorkgroupNode.ImageIndex:=1;
    WorkgroupNode.SelectedIndex:=1;
    for j:=0 to (N.Objects[i] as TStrings).Count - 1 do begin
     ComputerNode:=TreeView1.Items.AddChild(WorkgroupNode, (N.Objects[i] as
TStrings).Strings[j]);
     ComputerNode.ImageIndex:=0;
    end;
   end;

  // Отримання IP адрес комп’ ютерів
  GetIPAddresses(N, Memo2.Lines);

  finally
   N.Free;
  end;
  TreeView1.FullExpand;

 finally
  Screen.Cursor:=crDefault;
 end;
end;


procedure TForm1.FormCreate(Sender: TObject);
begin
 Edit1.Text:=GetComputerName;

  if LoadIpHlp then
  begin
      DOIpStuff;
      Timer1.Enabled := true;
  end;

end;


procedure TForm1.Timer1Timer(Sender: TObject);
begin

   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;

end;


procedure TForm1.DOIpStuff;
begin

 Get_TCPTable( MemoX.Lines );
 Get_UDPTable( MemoTR.Lines );

end;


procedure TForm1.Button2Click(Sender: TObject);
begin

Form2.Show;

end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
begin

Form1.Close;

end;


procedure TForm1.SpeedButton2Click(Sender: TObject);
begin

Form1.Close;

end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin

Form2.Show;

end;


end.
```

Файл IPtraff.pas – відслідковування та контроль трафіку для сервісу віртуалізацій мережних функцій NFV

```
unit IPtraff;

interface

uses
  Windows, Messages, SysUtils, Classes, Dialogs, IpHlpApi;

const
  NULL_IP       = '   0.  0.  0.  0' ;


type
  TWellKnownPort = record
    Prt: DWORD;
    Srv: string[20];
  end;


const

  WellKnownPorts: array[1..32] of TWellKnownPort
  = (
//    ( Prt: 0; Srv:    ' RESRVED'  ),     {Зарезервовано}
    ( Prt: 7; Srv:    ' ECHO    '  ),     {Пінгування    }
    ( Prt: 9; Srv:    ' DISCARD'  ),
    ( Prt: 13; Srv:   ' DAYTIME'  ),
    ( Prt: 17; Srv:   ' QOTD    '  ),     {Показчик на день}
    ( Prt: 19; Srv:   ' CHARGEN'  ),     {Генератор символів}
    ( Prt: 20; Srv:   ' FTPDATA'  ),     { Протокол File Transfer Protocol -
дані}
    ( Prt: 21; Srv:   ' FTPCTRL'  ),     { Протокол File Transfer Protocol -
управління}
    ( Prt: 22; Srv:   ' SSH     '  ),
    ( Prt: 23; Srv:   ' TELNET  '  ),
    ( Prt: 25; Srv:   ' SMTP    '  ),     { Протокол Simple Mail Transfer
Protocol}
    ( Prt: 37; Srv:   ' TIME    '  ),     { Протокол Time Protocol }
    ( Prt: 43; Srv:   ' WHOIS   '  ),     { WHO IS сервіс  }
    ( Prt: 53; Srv:   ' DNS     '  ),     { Domain Name Service }
    ( Prt: 67; Srv:   ' BOOTPS  '  ),     { BOOTP сервер }
    ( Prt: 68; Srv:   ' BOOTPC  '  ),     { BOOTP клієнт }
    ( Prt: 69; Srv:   ' TFTP    '  ),     { Стандартний  FTP  }
    ( Prt: 70; Srv:   ' GOPHER  '  ),     { Протокол Gopher       }
    ( Prt: 79; Srv:   ' FINGER  '  ),     { Протокол Finger       }
    ( Prt: 80; Srv:   ' HTTP    '  ),     { Протокол HTTP         }
    ( Prt: 88; Srv:   ' KERBROS'  ),     { Протокол Kerberos     }
    ( Prt: 109; Srv:  ' POP2    '  ),     { Протокол Post Office Protocol Version
2 }
    ( Prt: 110; Srv:  ' POP3    '  ),     { Протокол Post Office Protocol Version
3 }
    ( Prt: 111; Srv:  ' SUN_RPC'  ),     { SUN віддалений виклик функцій }
    ( Prt: 119; Srv:  ' NNTP    '  ),     { Протокол Network News Transfer
Protocol }
    ( Prt: 123; Srv:  ' NTP     '  ),     { Протокол Network Time protocol
}
    ( Prt: 135; Srv:  ' DCOMRPC'  ),     { Локальний сервіce             }
    ( Prt: 137; Srv:  ' NBNAME  '  ),     { NETBIOS сервіс імен          }
    ( Prt: 138; Srv:  ' NBDGRAM'  ),     { NETBIOS сервіс датаграм     }
    ( Prt: 139; Srv:  ' NBSESS  '  ),     { NETBIOS сервіс сесій        }
    ( Prt: 143; Srv:  ' IMAP    '  ),     { Протокол Internet Message Access
Protocol }
    ( Prt: 161; Srv:  ' SNMP    '  ),     { Протокол Simple Netw. Management
Protocol }
    ( Prt: 169; Srv:  ' SEND    '  )
    );
```

```
  const
  ICMP_ERROR_BASE = 11000;
  IcmpErr : array[1..22] of string =
  (
   ' IP_BUFFER_TOO_SMALL' ,' IP_DEST_NET_UNREACHABLE' , '
IP_DEST_HOST_UNREACHABLE' ,
   ' IP_PROTOCOL_UNREACHABLE' , ' IP_DEST_PORT_UNREACHABLE' , ' IP_NO_RESOURCES'
,
   ' IP_BAD_OPTION' ,' IP_HARDWARE_ERROR' , ' IP_PACKET_TOO_BIG' , '
IP_REQUEST_TIMED_OUT' ,
   ' IP_BAD_REQUEST' ,' IP_BAD_ROUTE' , ' IP_TTL_EXPIRED_TRANSIT' ,
   ' IP_TTL_EXPIRED_REASSEM' ,' IP_PARAMETER_PROBLEM' , ' IP_SOURCE_QUENCH' ,
   ' IP_OPTION_TOO_BIG' , ' IP_BAD_DESTINATION' ,' IP_ADDRESS_DELETED' ,
   ' IP_SPEC_MTU_CHANGE' , ' IP_MTU_CHANGE' , ' IP_UNLOAD'
  );


  ARPEntryType  : array[1..4] of string = ( ' Інший' , ' Несправний' ,
    ' Динамічний' , ' Статичний'
    );
  TCPConnState  :
    array[1..12] of string =
 ( ' CLOSED' , ' READ' , ' SYN_SENT' ,
    ' SYN_RCVD' , ' ESTABLISHED' , ' FIN_WAIT1' ,
    ' FIN_WAIT2' , ' WAIT' , ' CLOSING' ,
    ' LAST_ACK' , ' WAIT' , ' DELETE_TCB'    );

  TCPToAlgo     : array[1..4] of string =
    ( ' Const.Timeout' , ' MIL-STD-1778' ,
    ' Van Jacobson' , ' Other'  );

  IPForwTypes   : array[1..4] of string =
    ( ' other' , ' invalid' , ' local' , ' remote'  );

  IPForwProtos  : array[1..18] of string =
    ( ' OTHER' , ' LOCAL' , ' NETMGMT' , ' ICMP' , ' EGP' ,
    ' GGP' , ' HELO' , ' RIP' , ' IS_IS' , ' ES_IS' ,
    ' CISCO' , ' BBN' , ' OSPF' , ' BGP' , ' BOOTP' ,
    ' AUTO_STAT' , ' STATIC' , ' NOT_DOD'  );

type

// для IpHlpNetworkParams
  TNetworkParams = record
    HostName: string ;
    DomainName: string ;
    CurrentDnsServer: string ;
    DnsServerTot: integer ;
    DnsServerNames: array [0..9] of string ;
    NodeType: UINT;
    ScopeID: string ;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
```

```
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ;
    LeaseExpires: LongInt;
  end ;

  TAdaptorRows = array of TAdaptorInfo ;



function IpHlpAdaptersInfo(var AdpTot: integer;var AdpRows: TAdaptorRows):
integer ;
procedure Get_AdaptersInfo( List: TStrings );
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
procedure Get_IfTable( List: TStrings );
function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
procedure Get_RecentDestIPs( List: TStrings );

// утіліти перетворення
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
function IpAddr2Str( IPAddr: DWORD ): string;
function Str2IpAddr( IPStr: string ): DWORD;
function Port2Str( nwoPort: DWORD ): string;
function Port2Wrd( nwoPort: DWORD ): DWORD;
function Port2Svc( Port: DWORD ): string;
function ICMPErr2Str( ICMPErrCode: DWORD) : string;

implementation

var
  RecentIPs     : TStringList;


{ перетворення точена у рядок, потім рядок знищується }
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
```

```
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;


{ пертворення MAC-адреси до ww-xx-yy-zz рядка }
function MacAddr2Str( MacAddr: TMacAddress; size: integer ): string;
var
  i             : integer;
begin
  if Size = 0 then
  begin
    Result := ' 00-00-00-00-00-00' ;
    EXIT;
  end
  else Result := ' ' ;
  //
  for i := 1 to Size do
    Result := Result + IntToHex( MacAddr[i], 2 ) + '-' ;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  IP-адреси  в мережний байт типу DWORD }
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
    Result := Result + Format( ' %3d.', [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;


{ перетворення  крапкову десяткову IP-адресу  в мережний байт типу  DWORD}
function Str2IpAddr( IPStr: string ): DWORD;
var
  i             : integer;
  Num           : DWORD;
begin
  Result := 0;
  for i := 1 to 4 do
  try
    Num := ( StrToInt( NextToken( IPStr, ' .'  ) ) ) shl 24;
    Result := ( Result shr 8 ) or Num;
  except
    Result := 0;
  end;

end;


{ перетворення  номер порту в мережний байт типу  DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;
```

```
{ перетворення  номера порту в мережний байт типу  string }
function Port2Str( nwoPort: DWORD ): string;
begin
  Result := IntToStr( Port2Wrd( nwoPort ) );
end;


{ перетворення  відомого номера порту у сервіс ID }
function Port2Svc( Port: DWORD ): string;
var
  i             : integer;
begin
  Result := Format( ' %4d' , [Port] ); // у випадку якщо порт не знайдено
  for i := Low( WellKnownPorts ) to High( WellKnownPorts ) do
    if Port = WellKnownPorts[i].Prt then
    begin
      Result := WellKnownPorts[i].Srv;
      BREAK;
    end;
end;


{ фіксуємо параметри мережі }

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' HOSTNAME          : '  + HostName );
        List.Add( ' DOMAIN            : '  + DomainName );
        List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP SCOPE        : '  + ScopeID );
        List.Add( ' ROUTING ENABLED   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   ENABLED   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     ENABLED   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS SERVER ADDR   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         //
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   //
begin
    InfoSize := 0 ;   //
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetNetworkParams( Nil, @InfoSize );  //
    if result <> ERROR_BUFFER_OVERFLOW then exit ; //
    GetMem (FixedInfo, InfoSize) ;                    //
    try
    result := GetNetworkParams( FixedInfo, @InfoSize );   //
```

```
    if result <> ERROR_SUCCESS then exit ;
    NetworkParams.DnsServerTot := 0 ;
    with FixedInfo^ do
    begin
        NetworkParams.HostName := trim (HostName) ;
        NetworkParams.DomainName := trim (DomainName) ;
        NetworkParams.ScopeId := trim (ScopeID) ;
        NetworkParams.NodeType := NodeType ;
        NetworkParams.EnableRouting := EnableRouting ;
        NetworkParams.EnableProxy := EnableProxy ;
        NetworkParams.EnableDNS := EnableDNS ;
        NetworkParams.DnsServerNames [0] := DNSServerList.IPAddress ;  //
        if NetworkParams.DnsServerNames [0] <> ' '  then
                                        NetworkParams.DnsServerTot := 1 ;
        PDnsServer := DnsServerList.Next;
        while PDnsServer <> Nil do
        begin
            NetworkParams.DnsServerNames [NetworkParams.DnsServerTot] :=
                                        PDnsServer^.IPAddress ;  //
            inc (NetworkParams.DnsServerTot) ;
            if NetworkParams.DnsServerTot >=
                            Length (NetworkParams.DnsServerNames) then exit ;
            PDnsServer := PDnsServer.Next ;
        end;
    end ;
    finally
       FreeMem (FixedInfo) ;                //
    end ;
end;

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

function ICMPErr2Str( ICMPErrCode: DWORD) : string;
begin
   Result := ' UnknownError : '  + IntToStr( ICMPErrCode );
   dec( ICMPErrCode, ICMP_ERROR_BASE );
   if ICMPErrCode in [Low(ICMpErr)..High(ICMPErr)] then
     Result := ICMPErr[ ICMPErrCode];
end;


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

// включаємо байти вводу/виводу для кожного адаптеру

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; //
  TableSize := 0;
   //  перший виклик: беремо необхідний розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  //
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
  try
      FillChar (pBuf^, TableSize, #0);  // очищуємо буфер, з W98 не потрібно

   // беремо показчик на таблицю
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
```

```
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // перетворюємо
Unicode у string
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
               ' %0.8x - %3d - %16s - %8d - %12d - %2d - %2d - %10d - %10d  -
%-s - %-s' ,
              [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
              dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
              dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
counters are 32-bit
              sIfName, sDescr] )  //  , додаємо введення/вивід
              );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // звільняємо пам' ять
end ;

function IpHlpIfEntry(Index: integer; var IfRow: TMibIfRow): integer ;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  FillChar (IfRow, SizeOf (TMibIfRow), #0);  // очищуємо буфер, з W98 не
потрібно
  IfRow.dwIndex := Index ;
  result := GetIfEntry (@IfRow) ;
end ;

//---------------------------------------------------------------------------
{ інформація про мережні адаптери }
```

```
function IpHlpAdaptersInfo(var AdpTot: integer; var AdpRows: TAdaptorRows):
integer ;
var
  BufLen        : DWORD;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  PIpAddr       : PTIP_ADDR_STRING;
  PBuf          : PCHAR ;
  I             : integer ;
begin
  SetLength (AdpRows, 4) ;
  AdpTot := 0 ;
  BufLen := 0 ;
  result := GetAdaptersInfo( Nil, @BufLen );
  if (result <> ERROR_INSUFFICIENT_BUFFER) and (result = NO_ERROR) then exit ;
  GetMem( pBuf, BufLen );
  try
      FillChar (pBuf^, BufLen, #0);  // очищуємо буфер
      result := GetAdaptersInfo( PTIP_ADAPTER_INFO (PBuf), @BufLen );
      if result = NO_ERROR then
      begin
         AdapterInfo := PTIP_ADAPTER_INFO (PBuf) ;
         while ( AdapterInfo <> nil ) do
         begin
            AdpRows [AdpTot].IPAddressTot := 0 ;
            SetLength (AdpRows [AdpTot].IPAddressList, 2) ;
            SetLength (AdpRows [AdpTot].IPMaskList, 2) ;
            AdpRows [AdpTot].GatewayTot := 0 ;
            SetLength (AdpRows [AdpTot].GatewayList, 2) ;
            AdpRows [AdpTot].DHCPTot := 0 ;
            SetLength (AdpRows [AdpTot].DHCPServer, 2) ;
            AdpRows [AdpTot].PrimWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].PrimWINSServer, 2) ;
            AdpRows [AdpTot].SecWINSTot := 0 ;
            SetLength (AdpRows [AdpTot].SecWINSServer, 2) ;
            AdpRows [AdpTot].CurrIPAddress := NULL_IP;
            AdpRows [AdpTot].CurrIPMask := NULL_IP;
            AdpRows [AdpTot].AdapterName := Trim( string(
AdapterInfo^.AdapterName ) );
            AdpRows [AdpTot].Description := Trim( string(
AdapterInfo^.Description ) );
            AdpRows [AdpTot].MacAddress := MacAddr2Str( TMacAddress(
                              AdapterInfo^.Address ),
AdapterInfo^.AddressLength ) ;
            AdpRows [AdpTot].Index := AdapterInfo^.Index ;
            AdpRows [AdpTot].aType := AdapterInfo^.aType ;
            AdpRows [AdpTot].DHCPEnabled := AdapterInfo^.DHCPEnabled ;
            if AdapterInfo^.CurrentIPAddress <> Nil then
            begin
                AdpRows [AdpTot].CurrIPAddress :=
AdapterInfo^.CurrentIPAddress.IpAddress ;
                AdpRows [AdpTot].CurrIPMask :=
AdapterInfo^.CurrentIPAddress.IpMask ;
            end ;

        // беремо список IP адрес та масок для IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].IPAddressList) <= I then
                begin
                     SetLength (AdpRows [AdpTot].IPAddressList, I * 2) ;
                     SetLength (AdpRows [AdpTot].IPMaskList, I * 2) ;
                end ;
            end ;
```

```
            AdpRows [AdpTot].IPAddressTot := I ;

         // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.GatewayList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].GatewayList [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].GatewayList) <= I then
                            SetLength (AdpRows [AdpTot].GatewayList, I * 2) ;
            end ;
            AdpRows [AdpTot].GatewayTot := I ;

         // беремо список IP адрес для GatewayList
            I := 0 ;
            PIpAddr := @AdapterInfo^.DHCPServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].DHCPServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].DHCPServer) <= I then
                            SetLength (AdpRows [AdpTot].DHCPServer, I * 2) ;
            end ;
            AdpRows [AdpTot].DHCPTot := I ;

         // беремо список IP адрес для PrimaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.PrimaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].PrimWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].PrimWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].PrimWINSServer, I * 2)
;
            end ;
            AdpRows [AdpTot].PrimWINSTot := I ;

        // беремо список IP адрес для SecondaryWINSServer
            I := 0 ;
            PIpAddr := @AdapterInfo^.SecondaryWINSServer ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].SecWINSServer [I] := PIpAddr.IpAddress ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
                if Length (AdpRows [AdpTot].SecWINSServer) <= I then
                            SetLength (AdpRows [AdpTot].SecWINSServer, I * 2) ;
            end ;
            AdpRows [AdpTot].SecWINSTot := I ;

            AdpRows [AdpTot].LeaseObtained := AdapterInfo^.LeaseObtained ;
            AdpRows [AdpTot].LeaseExpires := AdapterInfo^.LeaseExpires ;

            inc (AdpTot) ;
            if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot * 2) ;  // more memory
            AdapterInfo := AdapterInfo^.Next;
          end ;
          SetLength (AdpRows, AdpTot) ;
        end ;
    finally
        FreeMem( pBuf );
    end ;
end ;
```

```
procedure Get_AdaptersInfo( List: TStrings );
var
  AdpTot: integer;
  AdpRows: TAdaptorRows ;
  Error: DWORD ;
  I: integer ;
  //J: integer ;  jpt
  //S: string ;        id.
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (AdpRows, 0) ;
  AdpTot := 0 ;
  Error := IpHlpAdaptersInfo(AdpTot, AdpRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if AdpTot = 0 then
      List.Add( ' Даних немає.'  )
  else
  begin
      for I := 0 to Pred (AdpTot) do
      begin
        with AdpRows [I] do
        begin
            //List.Add(AdapterName + ' -'  + Description ); // jpt : не корисне
            List.Add( Format(' %8.8x - %6s - %16s - %2d - %16s - %16s - %16s' ,
                [Index, AdaptTypes[aType], MacAddress, DHCPEnabled,
                GatewayList [0], DHCPServer [0], PrimWINSServer [0]] ) );
            {if IPAddressTot <> 0 then    // jpt : not useful
            begin
                S := ' '  ;
                for J := 0 to Pred (IPAddressTot) do
                        S := S + IPAddressList [J] + ' /'  + IPMaskList [J] + '
- '  ;
                List.Add(IntToStr (IPAddressTot) + '  IP Addresse(s): '  + S);
            end ;
            List.Add( '   ' ); }
        end ;
      end ;
  end ;
  SetLength (AdpRows, 0) ;
end ;

//---------------------------------------------------------------------------
{ зчитуємо кількість раундів, та час звертання до IP }
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT := -1; //
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;

//---------------------------------------------------------------------------
{ ARP-таблиця – список відношень між віддаленими IP та віддаленими MAC-адресами.

}
procedure Get_ARPTable( List: TStrings );
var
  IPNetRow      : TMibIPNetRow;
  TableSize     : DWORD;
  NumEntries    : DWORD;
  ErrorCode     : DWORD;
```

```
  i             : integer;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  //  перший виклик: беремо довжину таблиці
  TableSize := 0;
  ErrorCode := GetIPNetTable( Nil, @TableSize, false );   //
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-кеш пустий.'  );
    EXIT;
  end;
  // беремо таблицю
  GetMem( pBuf, TableSize );
  NumEntries := 0 ;
  try
  ErrorCode := GetIpNetTable( PTMIBIPNetTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBIPNetTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then //.
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        IPNetRow := PTMIBIPNetRow( PBuf )^;
        with IPNetRow do
          List.Add( Format( ' %8x - %12s - %16s - %10s' ,
                            [dwIndex, MacAddr2Str( bPhysAddr, dwPhysAddrLen ),
                            IPAddr2Str( dwAddr ), ARPEntryType[dwType]
                            ]));
        inc( pBuf, SizeOf( IPNetRow ) );
      end;
    end
    else
      List.Add( '  ARP-кеш пустий.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we _must_ restore pointer!
  finally
      dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
      FreeMem( pBuf );
  end ;
end;


//-----------------------------------------------------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  RecentIPs.Clear;
  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
```

```
    EXIT;

  // беремо розмір пам' яти, який потрібно та здійснюємо виклик знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

    NumEntries := PTMIBTCPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        TCPRow := PTMIBTCPRow( pBuf )^; // беремо наступний запис
        with TCPRow do
        begin
          if dwRemoteAddr = 0 then
            dwRemotePort := 0;
          DestIP := IPAddr2Str( dwRemoteAddr );
          List.Add(
            Format( ' %15s : %-7s -> %15s : %-7s -> %-16s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) ),
              DestIP,
              Port2Svc( Port2Wrd( dwRemotePort ) ),
              TCPConnState[dwState]
              ] ) );
         //
            if (not ( dwRemoteAddr = 0 ))
            and ( RecentIps.IndexOf(DestIP) = -1 ) then
               RecentIPs.Add( DestIP );
        end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;

//---------------------------------------------------------------------------
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі: '  + TCPToAlgo[dwRTOAlgorithm] );
      List.Add( ' Мінімальний час виведення         : '  + IntToStr( dwRTOMin )
+ '  ms'  );
      List.Add( ' Максимальний час виведення         : '  + IntToStr( dwRTOMax )
+ '  ms'  );
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
```

```
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Відновлений зв'язок   : '  + IntToStr( dwEstabResets ) );
      List.Add( ' Поточний зв'язок .: '  + IntToStr( dwCurrEstab ) );
      List.Add( ' Отримано сегментів        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлено сегментів            : '  + IntToStr( dwOutSegs )
);
      List.Add( ' Ретрансльовано сегментів   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилки входження          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виведення          : '  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки   : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

function IpHlpTCPStatistics (var TCPStats: TMibTCPStats): integer ;
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetTCPStatistics( @TCPStats );
end;

//-------------------------------------------------------------------------------
procedure Get_UDPTable( List: TStrings );
var
  UDPRow        : TMIBUDPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;

  //  перший виклик : беремо розмір таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetUDPTable(Nil, @TableSize, false );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо потрібний розмір пам'яті, викликаємо знову
  GetMem( pBuf, TableSize );

  // беремо таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMIBUDPTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) ); // беремо розмір останьої таблиці
      for i := 1 to NumEntries do
      begin
        UDPRow := PTMIBUDPRow( pBuf )^; // беремо наступний запис
        with UDPRow do
          List.Add( Format( ' %15s : %-6s' ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
```

```
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0; ;
  NumEntries := 0 ;
  //  перший виклик: беремо довжину таблиці
  ErrorCode := GetIpAddrTable(Nil, @TableSize, true );  //
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrRow := PTMIBIPAddrRow( pBuf )^;
        with IPAddrRow do
          List.Add( Format( ' %8.8x | %15s | %15s | %15s | %8.8d' ,
            [dwIndex,
            IPAddr2Str( dwAddr ),
              IPAddr2Str( dwMask ),
              IPAddr2Str( dwBCastAddr ),
              dwReasmSize
              ] ) );
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // we must restore pointer!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
{ беремо дані з таблиці маршрутизатора Cisco}
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
```

```
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  //  перший виклик: беремо довжину таблиці
  NumEntries := 0 ;
  ErrorCode := GetIpForwardTable(Nil, @TableSize, true);
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо таблицю
  GetMem( pBuf, TableSize );
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize,
true);
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPForwardTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPForwRow := PTMibIPForwardRow( pBuf )^;
        with IPForwRow do
        begin
          if (dwForwardType < 1)
          or (dwForwardType > 4) then
                   dwForwardType := 1 ;   //  , враховуємо погані значення
          List.Add( Format(
            ' %15s | %15s | %15s | %8.8x | %7s |   %5.5d |  %7s |   %2.2d' ,
            [IPAddr2Str( dwForwardDest ),
            IPAddr2Str( dwForwardMask ),
              IPAddr2Str( dwForwardNextHop ),
              dwForwardIFIndex,
              IPForwTypes[dwForwardType],
              dwForwardNextHopAS,
              IPForwProtos[dwForwardProto],
              dwForwardMetric1
              ] ) );
        end ;
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' Даних немає.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблоковане пересилання      : '  + ' Так'  )
```

```
      else
        List.add( ' Розблоковане пересилання      : '  + ' Hi'  );
      List.add( ' Вбудований TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримані датаграми       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку     (в)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Адреса помилкова    (в)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
//
      List.add( ' Невідомий протокол (в)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Датаграма відвергнута     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Запит виведення            : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрутизація в маршрутизаторі Cisco відвергнута      : '  +
inttostr( dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (з)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Час виведення перебору      : '  + inttostr( dwReasmTimeOut )
);
      List.add( ' Запити перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Перебори здійснено : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Фрагментація мережі успішна: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграми фрагментовано    : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' маршрут в таблиці маршрутизації Cisco : '  + inttostr(
dwNumRoutes ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

function IpHlpIPStatistics (var IPStats: TMibIPStats): integer ;      //
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetIPStatistics( @IPStats );
end ;

//---------------------------------------------------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Datagrams (в)    : '  + inttostr( dwInDatagrams ) );
      List.add( ' Datagrams (з)   : '  + inttostr( dwOutDatagrams ) );
      List.add( ' No Ports          : '  + inttostr( dwNoPorts ) );
      List.add( ' Errors    (в)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP Listen Ports  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;
```

```
//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *//
function IpHlpUdpStatistics (UdpStats: TMibUDPStats): integer ;     //
begin
    result := ERROR_NOT_SUPPORTED ;
    if NOT LoadIpHlp then exit ;
    result := GetUDPStatistics (@UdpStats) ;
end ;



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

procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );

var
  ErrorCode     : DWORD;
  ICMPStats     : PTMibICMPInfo;

begin
  if ( ICMPIn = nil ) or ( ICMPOut = nil ) then EXIT;
  ICMPIn.Clear;
  ICMPOut.Clear;
  New( ICMPStats );
  ErrorCode := GetICMPStatistics( ICMPStats );
  if ErrorCode = NO_ERROR then
  begin
    with ICMPStats.InStats do
    begin
      ICMPIn.Add( ' Отримано повідослень    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилки ICMP            : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування    : '  + IntToStr( dwDestUnreachs ) );
      ICMPIn.Add( ' Час перевищено      : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми параметрів     : '  + IntToStr( dwParmProbs ) );
      ICMPIn.Add( ' Джерело відключено       : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перенаправлення          : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Exo запит      : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Exo повтор       : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу   : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPIn.Add( ' Запит маски адреси : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор маски адреси   : '  + IntToStr( dwAddrReps ) );
    end;
     //


    with ICMPStats.OutStats do


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

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл NetConst.pas – ініціалізація констант

```
unit NetConst;



interface

resourcestring

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

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

 SFileError002 = '  - файл не знайдено' ;
  SFileError003 = '  - шлях не знайдено' ;
  SFileError004 = '  - не можу відкрити файл' ;
  SFileError005 = '  - немає доступу' ;
  SFileError014 = '  - не достатньо пам"яті' ;
  SFileError015 = '  - не можу знайти драйвер' ;
  SFileError017 = '  - не можу перемістити файл' ;
  SFileError019 = '  - носій захищений від запису' ;
  SFileError020 = '  - не можу знайти пристрій' ;
  SFileError021 = '  - пристрій не відкривається для читання' ;
  SFileError022 = '  - пристрій не може розпізнати команду' ;
  SFileError025 = '  - вказана область не знайдена' ;
  SFileError026 = '  - пристрій недоступний' ;
  SFileError027 = '  - сектор не знайдено' ;
  SFileError029 = '  - помилка запису на пристрій' ;
  SFileError030 = '  - помилка читання з пристрою' ;
  SFileError032 = '  - файл використовується іншою програмою' ;
  SFileError036 = '  - занадто багато відкритих файлів' ;
  SFileError038 = '  - досягнутий кінець файлу' ;
  SFileError039 = '  - диск переповнений' ;
  SFileError050 = '  - запит не підтримується' ;
  SFileError051 = '  - віддалений комп' ютер недоступний' ;
  SFileError052 = '  - у мережі знайдені ідентичні імена' ;
  SFileError053 = '  - мережний шлях не знайдено' ;
  SFileError054 = '  - мережа зайнята' ;
  SFileError055 = '  - ресурс мережі або пристрій недоступний' ;
  SFileError057 = '  - апаратна помилка в мережному адаптері' ;
  SFileError058 = '  - сервер не в змозі виконувати дану дію' ;
  SFileError059 = '  - помилка у мережі' ;
  SFileError064 = '  - недоступне мережне ім"я' ;
  SFileError065 = '  - немає доступу до мережі' ;
  SFileError066 = '  - невірно вказатий тип мережного ресурсу' ;
  SFileError067 = '  - не знайдене вказане мережне ім"я' ;
  SFileError070 = '  - відключейний сервер мережі' ;
  SFileError082 = '  - не можу створити файл чи каталог' ;
  SFileError112 = '  - не достатньо вільного місця на диску' ;
  SFileError123 = '  - в імені файлу вказано недопустимий символ' ;
  SFileError161 = '  - неправильно вказано шдях' ;
```

```
  SFileError183 = '  - файл не існує' ;

  SCannotSetSize = ' Не можу змінити розмір файлу' ;

  SUnableToCompress = ' Не можу заархівувати дані' ;
  SUnableToDecompress = ' Не можу розархівувати дані' ;

  SCannotFindNetwork = ' Не можу знайти  мережу' ;

implementation

end.
```

Файл Networks.pas – робота з мережею для сервісу віртуалізацій мережних функцій NFV

```
unit Networks;

interface

uses Windows, ShellAPI, ShlObj, ActiveX, ComObj, Dim, Classes, SysUtils,
WinSock;

type
  ComputerFound = class (Exception);
  ECannotFindNetwork = class (Exception);

  TStringObject = class (TObject)
  private
    FValue: TString;
    FTag: Integer;
    FData: Pointer;
    FRefObj: TObject;
    procedure SetValue(const Value: TString);
    procedure SetData(const Value: Pointer);
    procedure SetRefObj(const Value: TObject);
    procedure SetTag(const Value: Integer);
  public
    property Value: TString read FValue write SetValue;
    property RefObj: TObject read FRefObj write SetRefObj;
    property Tag: Integer read FTag write SetTag;
    property Data: Pointer read FData write SetData;
  end;

  TStringObjectArray = class (TDynamicArray)
  private
    function GetObject(Index: Integer): TStringObject;
    function GetData(Index: Integer): Pointer;
    function GetRefObj(Index: Integer): TObject;
    function GetTag(Index: Integer): Integer;
    function GetValue(Index: Integer): TString;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetRefObj(Index: Integer; const Value: TObject);
    procedure SetTag(Index: Integer; const Value: Integer);
    procedure SetValue(Index: Integer; const Value: TString);

    function FreeObject(Index: Integer; var Obj: TStringObject): Integer;

    procedure FreeItem(Index: Integer);
    procedure CreateItem(Index: Integer);

  protected
    procedure SetCount(const NewCount: Cardinal); override;
  public
    function Add: Integer; override;
    procedure Insert(Index: Integer); override;
    procedure Delete(Index: Integer); override;
    function AddItem(const Item): Integer; override;
    procedure InsertItem(Index: Integer; const Item); override;
    procedure DeleteItem(Index: Integer; out Item); override;
    property Value[Index: Integer]: TString read GetValue write SetValue;
default;
    property RefObj[Index: Integer]: TObject read GetRefObj write SetRefObj;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;
    property Data[Index: Integer]: Pointer read GetData write SetData;
    constructor Create;
    destructor Destroy; override;
  end;

  TStringObjectList = class (TStrings)
```

```
  private
    FArray: TStringObjectArray;
    function GetData(Index: Integer): Pointer;
    function GetTag(Index: Integer): Integer;
    procedure SetData(Index: Integer; const Value: Pointer);
    procedure SetTag(Index: Integer; const Value: Integer);
  protected
    function Get(Index: Integer): string; override;
    function GetCount: Integer; override;
    function GetObject(Index: Integer): TObject; override;
    procedure Put(Index: Integer; const S: string); override;
    procedure PutObject(Index: Integer; AObject: TObject); override;

  public
    property Data[Index: Integer]: Pointer read GetData write SetData;
    property Tag[Index: Integer]: Integer read GetTag write SetTag;

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Exchange(Index1, Index2: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;


    constructor Create;
    destructor Destroy; override;
  end;

  {TNetworkWorkgroup - клас, який створює список всіх комп' ютерів в робочій
групі. Цей клас є нащадком класу TStrings і повністю сумісний з іншим нащадками
цього класу.  Об' єкти цього класу записуються у властивості об' єктів класу
TNetworkNeiborhood. Клас TNetworkNeiborhood містить об' єкти і властивості
робочих груп. }

  TNetworkWorkgroup = class (TStringObjectList);


  {TNetworkNeiborhood - клас, який створює список всіх робочих груп в мережі}
  TNetworkNeighborhood = class (TStringObjectList)
  private
    function CreatePIDL(Size: Integer): PItemIDList;
    procedure DisposePIDL(ID: PItemIDList);
    function NextPIDL(IDList: PItemIDList): PItemIDList;
    function GetPIDLSize(IDList: PItemIDList): Integer;
    function CopyPIDL(IDList: PItemIDList): PItemIDList;
    procedure StripLastID(IDList: PItemIDList);
    function GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
    class function GetDisplayName(ShellFolder: IShellFolder; PIDL: PItemIDList):
TString;
    function OriginFolder:  IShellFolder;
    function OriginFolderNT: IShellFolder;
    class function EnumObjects(ShellFolder: IShellFolder): IEnumIDList;
    class procedure ParseFolder(Folder: IShellFolder; Items: TStringObjectList;
StorePIDLs: Boolean = False);
    class procedure ParseFolderEx(Folder: IShellFolder; Items: TStrings);

    function FreeRefObj(Index: Integer; var Obj: TStringObject): Integer;
    function GetWorkgroup(Name: TString): TNetworkWorkgroup;

  public
    { Процедура Оновлення шукає всі доступні робочі групи в мережі }

    procedure Refresh;

    { містить списки всіх комп' ютерів в  мережі}

    property Workgroup[Name: TString]: TNetworkWorkgroup read GetWorkgroup;
```

```
   { Функція FindComputer шукає комп’ ютер зі вказаним ім’ ям і повертає ім’ я робочої групи де його знайдено, або порожню строку
     якщо комп’ ютера з таким іменем у мережі немає }

    function FindComputer(Name: TString): TString;

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

    { Процедура ListNetwork копіює відсортований в алфавітному порядку список всіх робочих груп і комп’ ютерів в мережі.
     Робочі групи мають ‘ TObject(1)’  у відповідному елементі властивості об’ єктів, а комп’ ютери - ‘ nil’ }

    procedure ListNetwork(Strings: TStrings);

    function Add(const S: string): Integer; override;
    procedure Clear; override;
    procedure Delete(Index: Integer); override;
    procedure Insert(Index: Integer; const S: string); override;
    constructor Create;
  end;

{ Функція GetIPAddress отримує IP адресу комп’ ютера або сервера.
  Параметр NetworkName конкретизує ім’ я комп’ ютера або  сервера.
  Ця функція повертає адреси IP у форматі XXX.XXX.XXX.XXX у разі успішного виконання, ‘ Error’  - коли неможливо ініціалізувати, ‘ Unknown’  – коли параметр NetworkName посилається на неіснуючий комп’ ютер або на комп’ ютер  без встановленого протоколу TCP/IP }

  function GetIPAddress(NetworkName: TString): TString;


{ GetIPAddresses отримує IP адреси всіх доступних комп’ ютерів мережі }

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



{ Функція EnumSharedResources перераховує загальні ресурси мережі. Параметр ComputerName конкретизує
  ім’ я комп’ ютера. }

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;


implementation

uses NetConst;

{ TStringObject }

procedure TStringObject.SetData(const Value: Pointer);
begin
  FData := Value;
end;

procedure TStringObject.SetRefObj(const Value: TObject);
begin
  FRefObj := Value;
end;

procedure TStringObject.SetTag(const Value: Integer);
begin
  FTag := Value;
end;

procedure TStringObject.SetValue(const Value: TString);
```

```
begin
  FValue := Value;
end;

{ TStringObjectArray }

function TStringObjectArray.Add: Integer;
begin
 Result:=inherited Add;
 CreateItem(Result);
end;

function TStringObjectArray.AddItem(const Item): Integer;
begin
 Result:=Add;
end;

constructor TStringObjectArray.Create;
begin
 inherited Create(0, SizeOf(TStringObject));
end;

procedure TStringObjectArray.CreateItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 P^:=TStringObject.Create;
end;

procedure TStringObjectArray.Delete(Index: Integer);
begin
 FreeItem(Index);
 inherited;
end;

procedure TStringObjectArray.DeleteItem(Index: Integer; out Item);
begin
 Delete(Index);
end;

destructor TStringObjectArray.Destroy;
begin
 ForEach(Integer(Self), @TStringObjectArray.FreeObject);
 inherited;
end;

procedure TStringObjectArray.FreeItem(Index: Integer);
var
 P: ^TStringObject;
begin
 P:=GetItemPtr(Index);
 FreeAndNil(P^);
end;

function TStringObjectArray.FreeObject(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj);
 Result:=0;
end;

function TStringObjectArray.GetData(Index: Integer): Pointer;
begin
 Result:=GetObject(Index).Data;
end;

function TStringObjectArray.GetObject(Index: Integer): TStringObject;
begin
```

```
 GetItem(Index, Result);
end;

function TStringObjectArray.GetRefObj(Index: Integer): TObject;
begin
 Result:=GetObject(Index).RefObj;
end;

function TStringObjectArray.GetTag(Index: Integer): Integer;
begin
 Result:=GetObject(Index).Tag;
end;

function TStringObjectArray.GetValue(Index: Integer): TString;
begin
 Result:=GetObject(Index).Value;
end;

procedure TStringObjectArray.Insert(Index: Integer);
begin
 inherited;
 CreateItem(Index);
end;

procedure TStringObjectArray.InsertItem(Index: Integer; const Item);
begin
 Insert(Index);
end;

procedure TStringObjectArray.SetCount(const NewCount: Cardinal);
var
 i, OldCount: Integer;
begin
 OldCount:=Count;
 if NewCount > Count then begin
  inherited SetCount(NewCount);
  for i:=OldCount to NewCount - 1 do CreateItem(i);
 end else if NewCount < Count then begin
  for i:=NewCount to OldCount - 1 do FreeItem(i);
  inherited SetCount(NewCount);
 end;
end;

procedure TStringObjectArray.SetData(Index: Integer; const Value: Pointer);
begin
 GetObject(Index).Data:=Value;
end;

procedure TStringObjectArray.SetRefObj(Index: Integer;
  const Value: TObject);
begin
 GetObject(Index).RefObj:=Value;
end;

procedure TStringObjectArray.SetTag(Index: Integer; const Value: Integer);
begin
 GetObject(Index).Tag:=Value;
end;

procedure TStringObjectArray.SetValue(Index: Integer; const Value: TString);
begin
 GetObject(Index).Value:=Value;
end;

{ TStringObjectList }

function TStringObjectList.Add(const S: string): Integer;
begin
 Result:=FArray.Add;
```

```
 FArray.Value[Result]:=S;
end;

procedure TStringObjectList.Clear;
begin
 FArray.Count:=0;
end;

constructor TStringObjectList.Create;
begin
 inherited Create;
 FArray:=TStringObjectArray.Create;
end;

procedure TStringObjectList.Delete(Index: Integer);
begin
 FArray.Delete(Index);
end;

destructor TStringObjectList.Destroy;
begin
 FArray.Free;
 inherited;
end;

procedure TStringObjectList.Exchange(Index1, Index2: Integer);
begin
 FArray.Swap(Index1, Index2);
end;

function TStringObjectList.Get(Index: Integer): string;
begin
 Result:=FArray.Value[Index];
end;

function TStringObjectList.GetCount: Integer;
begin
 Result:=FArray.Count;
end;

function TStringObjectList.GetData(Index: Integer): Pointer;
begin
 Result:=FArray.Data[Index];
end;

function TStringObjectList.GetObject(Index: Integer): TObject;
begin
 Result:=FArray.RefObj[Index];
end;

function TStringObjectList.GetTag(Index: Integer): Integer;
begin
 Result:=FArray.Tag[Index];
end;

procedure TStringObjectList.Insert(Index: Integer; const S: string);
begin
 FArray.Insert(Index);
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.Put(Index: Integer; const S: string);
begin
 FArray.Value[Index]:=S;
end;

procedure TStringObjectList.PutObject(Index: Integer; AObject: TObject);
begin
 FArray.RefObj[Index]:=AObject;
```

```
end;

procedure TStringObjectList.SetData(Index: Integer; const Value: Pointer);
begin
 FArray.Data[Index]:=Value;
end;

procedure TStringObjectList.SetTag(Index: Integer; const Value: Integer);
begin
 FArray.Tag[Index]:=Value;
end;

{ TNetworkNeighborhood }

function TNetworkNeighborhood.Add(const S: string): Integer;
begin
 Result:=inherited Add(S);
 Objects[Result]:=TNetworkWorkgroup.Create;
end;

procedure TNetworkNeighborhood.Clear;
begin
 FArray.ForEach(Integer(Self), @TNetworkNeighborhood.FreeRefObj);
 inherited;
end;

function TNetworkNeighborhood.CopyPIDL(IDList: PItemIDList): PItemIDList;
var
 Size: Integer;
begin
 Size := GetPIDLSize(IDList);
 Result := CreatePIDL(Size);
 if Assigned(Result) then CopyMemory(Result, IDList, Size);
end;

constructor TNetworkNeighborhood.Create;
begin
 inherited Create;
 Refresh;
end;

function TNetworkNeighborhood.CreatePIDL(Size: Integer): PItemIDList;
var
 Malloc: IMalloc;
 HR: HResult;
begin
 Result := nil;
 HR := SHGetMalloc(Malloc);
 if Failed(HR) then  Exit;
 try
  Result := Malloc.Alloc(Size);
  if Assigned(Result) then FillChar(Result^, Size, 0);
 finally
 end;
end;

procedure TNetworkNeighborhood.Delete(Index: Integer);
begin
end;

procedure TNetworkNeighborhood.DisposePIDL(ID: PItemIDList);
var
 Malloc: IMalloc;
begin
 if ID = nil then Exit;
 OLECheck(SHGetMalloc(Malloc));
 Malloc.Free(ID);
end;
```

```
class function TNetworkNeighborhood.EnumObjects(
  ShellFolder: IShellFolder): IEnumIDList;
const
 Flags = SHCONTF_FOLDERS or SHCONTF_NONFOLDERS or SHCONTF_INCLUDEHIDDEN;
begin
 ShellFolder.EnumObjects(0, Flags, Result);
end;

function TNetworkNeighborhood.FindComputer(Name: TString): TString;
var
 i, j: Integer;
 List: TNetworkWorkgroup;
 S: TString;
begin
 Result:=' ' ;
 try
  for i:=0 to Count - 1 do begin
   List:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to List.Count - 1 do begin
    S:=List[j];
    CleanUp(S);
    if EqualText(Name, S) then begin
     Result:=Strings[i];
     raise ComputerFound.Create(' ' );
    end;
   end;
  end;
 except
  if not (ExceptObject is ComputerFound) then raise;
 end;
end;

function TNetworkNeighborhood.FreeRefObj(Index: Integer;
  var Obj: TStringObject): Integer;
begin
 FreeAndNil(Obj.FRefObj);
 Result:=0;
end;

class function TNetworkNeighborhood.GetDisplayName(ShellFolder: IShellFolder;
  PIDL: PItemIDList): TString;
var
 StrRet: TStrRet;
 P: PChar;
begin
 Result := ' ' ;
 ShellFolder.GetDisplayNameOf(PIDL, SHGDN_NORMAL, StrRet);
 case StrRet.uType of
  STRRET_CSTR: SetString(Result, StrRet.cStr, lStrLen(StrRet.cStr));
  STRRET_OFFSET: begin
   P := @PIDL.mkid.abID[StrRet.uOffset - SizeOf(PIDL.mkid.cb)];
   SetString(Result, P, PIDL.mkid.cb - StrRet.uOffset);
  end;
  STRRET_WSTR: Result := StrRet.pOleStr;
 end;
 CleanUp(Result, True);
end;

function TNetworkNeighborhood.GetPIDLSize(IDList: PItemIDList): Integer;
begin
 Result := 0;
 if Assigned(IDList) then begin
  Result := SizeOf(IDList^.mkid.cb);
  while IDList^.mkid.cb <> 0 do begin
   Result := Result + IDList^.mkid.cb;
   IDList := NextPIDL(IDList);
  end;
 end;
end;
```

```
function TNetworkNeighborhood.GetPrevPIDL(PIDL: PItemIDList): PItemIDList;
var
 Temp: PItemIDList;
begin
 Temp := CopyPIDL(PIDL);
 if Assigned(Temp) then StripLastID(Temp);
 if Temp.mkid.cb <> 0 then Result:=Temp else Result:=nil;
end;

function TNetworkNeighborhood.GetWorkgroup(Name: TString): TNetworkWorkgroup;
var
 Index: Integer;
begin
 Index:=IndexOf(Name);
 if Index<>-1 then Result:=Objects[Index] as TNetworkWorkgroup else Result:=nil;
end;

procedure TNetworkNeighborhood.Insert(Index: Integer; const S: string);
begin
end;

procedure TNetworkNeighborhood.ListComputers(Strings: TStrings);
var
 i, j: integer;
 L: TNetworkWorkgroup;
 S: TString;
begin
 Strings.BeginUpdate;
 try
  Strings.Clear;
  for i:=0 to Count - 1 do begin
   L:=Objects[i] as TNetworkWorkgroup;
   for j:=0 to L.Count - 1 do begin
    S:=L[j];
    CleanUp(S);
    Strings.Add(S);
   end;
  end;
 finally
  Strings.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.ListNetwork(Strings: TStrings);
var
 List: TStringList;
 i: Integer;
begin
 List:=TStringList.Create;
 try
  List.AddStrings(Self);
  for i:=0 to List.Count - 1 do List.Objects[i]:=TObject(1);
  for i:=0 to Count - 1 do begin
   List.AddStrings(Objects[i] as TStrings);
  end;
  for i:=Count to List.Count - 1 do List.Objects[i]:=nil;
  List.Sort;
  Strings.Assign(List);
 finally
  List.Free;
 end;
end;

function TNetworkNeighborhood.NextPIDL(IDList: PItemIDList): PItemIDList;
begin
 Result := IDList;
 Inc(PChar(Result), IDList^.mkid.cb);
end;
```

```
function TNetworkNeighborhood.OriginFolder: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString;
 P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 P:=StringToOleStr(S);
 Flags:=0;
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 try
   Network:=GetPrevPIDL(Workgroup);
  try
   Desktop.BindToObject(Network, nil, IShellFolder, Pointer(Result));
  finally
   DisposePIDL(Network);
  end;
 finally
  DisposePIDL(Workgroup);
 end;
end;

function TNetworkNeighborhood.OriginFolderNT: IShellFolder;
var
 Desktop: IShellFolder;
 S: TString; W: WideString; P: PWideChar;
 Len, Flags: LongWord;
 Machine, Workgroup, Network: PItemIDList;
 NetShell: IShellFolder;
 Enum: IEnumIDList;
 ID: PItemIDList;
begin
 S:=' \\' +GetComputerName;
 Len:=Length(S);
 W:=S; P:=PWideChar(W);
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Workgroup:=GetPrevPIDL(Machine);
 Network:=GetPrevPIDL(Workgroup);
 Desktop.BindToObject(Network, nil, IShellFolder, NetShell);
 Enum:=EnumObjects(NetShell);
 Enum.Next(1, ID, Flags);
 NetShell.BindToObject(ID, nil, IShellFolder, Pointer(Result));
 DisposePIDL(Network);
 DisposePIDL(Workgroup);
end;

class procedure TNetworkNeighborhood.ParseFolder(Folder: IShellFolder;
  Items: TStringObjectList; StorePIDLs: Boolean);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
 Index: Integer;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Index:=Items.Add(S);
```

```
   if StorePIDLs then Items.Data[Index]:=ID;
  end;
 finally
  Items.EndUpdate;
 end;
end;

class procedure TNetworkNeighborhood.ParseFolderEx(Folder: IShellFolder;
  Items: TStrings);
var
 ID: PItemiDList;
 EnumList: IEnumIDList;
 NumIDs: LongWord;
 S: TString;
begin
 Items.BeginUpdate;
 try
  Items.Clear;
  EnumList:=EnumObjects(Folder);
  if Assigned(EnumList) then while EnumList.Next(1, ID, NumIDs) = S_OK do begin
   S:=GetDisplayName(Folder, ID);
   Items.Add(S);
  end;
 finally
  Items.EndUpdate;
 end;
end;

procedure TNetworkNeighborhood.Refresh;
var
 Network: IShellFolder;
 Workgroup: IShellFolder;
 i: Integer;
begin
 try
  if WinNT and (not Win2K) then Network:=OriginFolderNT else
   Network:=OriginFolder;
  ParseFolder(Network, Self, True);
  for i:=0 to Count - 1 do begin
   Network.BindToObject(PItemIDList(Data[i]), nil, IShellFolder, Workgroup);
   ParseFolder(Workgroup, Objects[i] as TStringObjectList, False);
   Workgroup:=nil;
  end;
 except
  raise ECannotFindNetwork.Create(SCannotFindNetwork);
 end;
end;

procedure TNetworkNeighborhood.StripLastID(IDList: PItemIDList);
var
 MarkerID: PItemIDList;
begin
 MarkerID := IDList;
 if Assigned(IDList) then begin
  while IDList.mkid.cb <> 0 do begin
   MarkerID := IDList;
   IDList := NextPIDL(IDList);
  end;
  MarkerID.mkid.cb := 0;
 end;
end;


procedure GetIPAddresses(Network: TNetworkNeighborhood; List: TStrings);
var
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
```

```
 i: Integer;
 TmpList: TStringList;
 S: TString;
begin
{ List.BeginUpdate;
 try}
  List.Clear;
  Error:=WSAStartup(MakeWord(1, 1), Data);
  if Error = 0 then begin
   TmpList:=TStringList.Create;
   try
    Network.ListComputers(TmpList);
    for i:=0 to TmpList.Count - 1 do begin
     HostEntry:=gethostbyname(PChar(TmpList[i]));
     Error:=GetLastError;
     if Error <> 0 then S:=‘ Unknown’  else begin
      Address:=PInAddr(HostEntry^.h_addr_list^)^;
      S:=inet_ntoa(Address);
     end;
     List.Add(Format(‘ %s [%s]’ , [TmpList[i], S]));
    end;
   finally
    TmpList.Free;
   end;
  end else begin
   List.Add(‘ Error’ );
  end;
{ finally
  List.EndUpdate;
 end;}
end;


function GetShellFolder(ComputerName: TString): IShellFolder;
var
 S: TString;
 W: WideString;
 P: PWideChar;
 Desktop: IShellFolder;
 Len, Flags: LongWord;
 Machine: PItemIDList;
begin
 S:=ComputerName;
 if Pos(‘ \\’ , S) <> 1 then S:=‘ \\’ +S;
 Len:=Length(S);
 W:=S;
 P:=@W[1];
 SHGetDesktopFolder(Desktop);
 Desktop.ParseDisplayName(0, nil, P, Len, Machine, Flags);
 Desktop.BindToObject(Machine, nil, IShellFolder, Pointer(Result));
end;

function EnumSharedResources(ComputerName: TString; List: TStrings): Boolean;
var
 ShellFolder: IShellFolder;
begin
 ShellFolder:=GetShellFolder(ComputerName);
 Result:=Assigned(ShellFolder);
 if Result then TNetworkNeighborhood.ParseFolderEx(ShellFolder, List);
end;



function GetIPAddress(NetworkName: TString): TString;
var
 Error: DWORD;
 HostEntry: PHostEnt;
 Data: WSAData;
 Address: In_Addr;
```

```
begin
 Error:=WSAStartup(MakeWord(1, 1), Data);
 if Error = 0 then begin
  HostEntry:=gethostbyname(PChar(NetworkName));
  Error:=GetLastError();
  if Error = 0 then begin
   Address:=PInAddr(HostEntry^.h_addr_list^)^;
   Result:=inet_ntoa(Address);
  end else begin
   Result:=‘ Unknown’ ;
  end;
 end else begin
  Result:=‘ Error’ ;
 end;
 WSACleanup();
end;

end.
```

Файл About.pas - довідка

```
unit About;

interface

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

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

end.
```