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 розгортання централізовано програмованих мереж на основі технології SDN”

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

Керівник проекту
кандидат технічних наук, доцент
_____ Коваленко А.С.
«____» _____ 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

Буханенко Лілії Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN*

2. Керівник роботи *Коваленко Анна Степанівна, канд. техн. 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 розгортання централізовано програмованих мереж на основі технології SDN*

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
_____ Коваленко А.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Буханенко Л.А.
(прізвище та ініціали)

АНОТАЦІЯ

Буханенко Л.А. Програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN. 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 розгортання централізовано програмованих мереж на основі технології SDN.

Метою розробки є програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN.

Результат роботи – програмна реалізація системи розгортання централізовано програмованих мереж на основі технології SDN.

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.

Ключові слова: комп'ютерна інженерія, SDN

ABSTRACT

Bukhanenko L.A. System software for deploying centrally programmable networks based on SDN technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software is developed, which is intended for the deployment system of centrally programmed networks based on SDN technology.

The purpose of the development is software for the deployment of centrally programmable networks based on SDN technology.

The result of the work is a software implementation of a system for deploying centrally programmable networks based on SDN technology.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

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

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the Delphi 10.4 environment.

Keywords: computer engineering, SDN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
УЛММР	–	узагальнена лінійна модель множинної регресії
МНК	–	метод найменших квадратів
СУБД	–	система управління базою даних
УМНК	–	узагальнений метод найменших квадратів
ОПР	–	особа яка приймає рішення

ВСТУП

Актуальність теми. Зі швидким ростом обсягів мережного трафіку й кількості підключених до мережі пристроїв конфігурування великомасштабних мереж перетворюється в дуже складне завдання. Щоб його спростити, потрібні серйозні зміни в підходах до побудови, експлуатації мереж і керуванню ними. SDN означає перегляд мережної архітектури, відділення керування від передачі даних і автоматизацію процесу адміністрування мережного встаткування. Однак у цей час SDN використовують лише великі компанії й інтернет-гіганти.

«Програмно визначаємі» (або «програмно конфігуруємі») мережі (Software-Defined Networking, SDN) і віртуалізація мережних функцій (Network Function Virtualization, NFV) стають фокусними темами найбільших галузевих форумів. І це не випадково, адже завдяки їхньому використанню значно міняються традиційні методи проектування й керування корпоративними мережами, мережною інфраструктурою телекомунікаційних компаній і ЦОД.

За прогнозом аналітиків, до 2025 року обсяг світового ринку SDN виросте до 35 млрд доларів. А 40% всіх витрат на мережі передачі даних будуть так чи інакше пов'язані з SDN. У першу чергу SDN будуть затребувані сервісами-провайдерами, хмарними комерційними центрами обробки даних, великими корпоративними ЦОД.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN.

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

– Огляд існуючих систем розгортання централізовано програмованих мереж на основі технології SDN.

– Дослідження системи розгортання централізовано програмованих мереж на основі технології SDN.

– Програмна реалізація системи розгортання централізовано програмованих мереж на основі технології SDN.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі розгортання централізовано програмованих мереж на основі технології SDN.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN, 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

Ми живемо в епоху кардинальної перебудови мережних інфраструктур, здійснюваної під прапором SDN і NFV. Її необхідність викликається багатьма факторами, у першу чергу швидким ростом числа підключених до Інтернету пристроїв і обсягів мережного трафіку, причому якщо в традиційних мережах він передається в основному між клієнтом і сервером (північ – південь), те у віртуалізованих і хмарних середовищах домінуючої стає трафік між серверами й віртуальними машинами (захід – схід). У результаті конфігурування великомасштабних мереж перетворюється в дуже складне завдання. Серйозні зміни в підходах до побудови, експлуатації мереж і керуванню ними потрібні також у зв'язку з поширенням мобільних технологій, концепції BYOD, Великих Даних і інших тенденцій.

Технологія SDN і протокол OpenFlow були розроблені в Стенфордському університеті у відповідь на потребу виділення мережних ресурсів для тестування нових сервісів. Ідею швидко підхопили великі інтернет-провайдери, і незабаром з'явилася організація Open Networking Forum. Одним з її починань стала стандартизація протоколу OpenFlow для взаємодії контролера з комутаторами в інфраструктурі SDN. Як очікується, SDN дозволить використовувати недорогі комутатори й керовані контролери.

Ідея NFV, що споконвічно просувалася великими європейськими операторами, припускала перенос мережних сервісів зі спеціалізованих пристроїв на стандартні комп'ютерні платформи у віртуалізовані середовища. Робота зі стандартизації NFV ведеться європейським інститутом ETSI. SDN і NFV – різні, хоча й частково пересічні технології.

1.2 Область застосування

Суть SDN складається у відділенні «площини керування» від «площини передачі даних», централізації керування й програмування мережі, зміні архітектури мережі в цілому. NFV же припускає перенос мережних функцій зі спеціалізованих пристроїв (appliance) на типові сервери (віртуальні середовища) і зміна функціональних елементів мережі. Стандартні сервери тепер здатні успішно заміняти колишні системи, що створювалися для вузьких областей (у тому числі в рішеннях для інфраструктури зв'язку).

SDN означає перегляд мережної архітектури, відділення керування від передачі даних і автоматизацію процесу адміністрування мережного встаткування, а NFV – перегляд компонентів мережі, що реалізують ті або інші послуги. SDN у зв'язуванні з NFV дозволяє абстрагувати послуги не тільки від устаткування, але й від його місцезнаходження. Більше того, послугу можна розподілити по мережі, замість того щоб локалізувати в конкретному вузлі».

Традиційна мережа IP являє собою набір функціональних блоків, причому в кожному її вузлі виконується обробка досить більших пакетів даних. Така архітектура складна, далеко не оптимальна й неминуче вносить істотні затримки. У великих мережах це серйознішає проблемою. У випадку SDN і NFV при побудові мережі використовується стандартне недороге встаткування, а процедури керування й необхідні в кожному мережному вузлі сервіси реалізуються за допомогою ПЗ.

Основні типи мереж, де може бути затребувана (і вже задіється) технологія SDN – це кампусні мережі, мережі ЦОД, хмарні платформи, а області застосування – мережі для хмар, оркестрація й автоматизація. У цей час основними об'єктами впровадження SDN є великі хмарні мережі й платформи. Поки лише деякі замовники вирішуються побудувати ЦОД на базі SDN, хоча такі приклади є.

Технологія NFV використовується в маршрутизаторах, міжмережних екранах і шлюзах, пристроях CDN, акселераторах WAN, контролерах доставки додатків (ADC) у мережах операторів і сервісів-провайдерів. SDN і NFV не зв'язані між собою, але добре доповнюють один одного, кажуть експерти.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN, 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

Ринок послуг комерційних дата-центрів в Україні – один з деяких сегментів економіки, що розвивається цілком передбачуване й згідно зі світовими тенденціями. Звичайно, відповідність тут неповне, але загалом напрямку розвитку в нас і у світі збігаються – відзначається з темпів росту попиту на традиційні послуги комерційних ЦОД (таких як хостинг, колокейшн і т.д.) з одночасним збільшенням ринку «хмарних» сервісів. При цьому останні вже формують близько 20% загального обсягу вітчизняного ринку. Слід зазначити, що хоча загальні показники сегмента комерційних ЦОД в Україні в 2023 році як мінімум не скоротилися (якщо вважати в національній валюті), відбулося це не від гарного життя – складна ситуація в країні підштовхнула багато компаній, у т.ч. зі сходу України, до пошуку альтернативних площадок для розміщення своїх IT-ресурсів. Разом з тим торік нові дата-центри продовжували вводитися в експлуатацію. З'явився, принаймні, один великий корпоративний ЦОД (створений компанією RIM2000 для одного з банків) і ще декілька комерційних. Але от як буде виглядати ринок за підсумками 2024 року, сказати поки складно. Швидше за все, це буде самий непростий рік в історії вітчизняного ринку, і багато операторів відчують падіння попиту, крім хіба що тих, які пропонують «хмарні» сервіси.

Відносний ріст

За підсумками 2023 року можна відзначити, що український ринок комерційних дата-центрів показав ріст всупереч всім складностям. Більше того, цифри дослідження говорять про те, що торік темпи приросту навіть

прискорилися в порівнянні з 2022-м. В 2023 році обсяг ринку в умовних юнітах склав 69,2 тис., збільшившись на 9% (у той час як 2022 рік додав ледве більше 5% у порівнянні з 2021-м).

Звичайно, такий ріст не можна назвати значним, і викликаний він головним чином вищезгаданими факторами, пов'язаними з війною й прагненням компаній скоротити витрати на підтримку своїх IT. Проте позитивна динаміка в сегменті в наявності. Виграли в першу чергу великі оператори з відносно більшими й надійними датами-центрами. У той же час середні й невеликі площадки в більшості випадків випробовували відтік клієнтів. Загальний потенційний обсяг українського ринку комерційних ЦОД сьогодні становить 105–110 тис. умовних юнітів (2500–2600 шаф або непохитно-місць). З огляду на нинішній рівень наповненості, можна зробити вивід про те, що «загальноукраїнська» ємність обрана приблизно на 65%.

Але насправді цей показник може виявитися ще вище, оскільки шафи досить рідко заповнюються IT-устаткуванням на 100%. І в деяких стійках можуть пустувати від декількох одиниць до декількох десятків юнітів, які вже ніхто не буде заповнювати (скажемо, через високе теплове навантаження). Також нерідкі випадки, коли клієнт бере під оренду ціла шафа або виділена зона дати-центра. Але, для зручності розрахунків будемо вважати, що він відразу займає всі юніти, узяті в оренду, хоча фізично це може бути й не так.

Підготовлена площа комерційних ЦОД в Україні оцінюється «СІБ» в 8,5 тис. кв. м (приблизно 0,02% загальносвітового ринку).

Якщо говорити про ринок у грошовому вираженні, то ситуація тут украй неоднозначна. У доларовому еквіваленті обсяг значно скоротився – приблизно на 20%, але в гривневому, навпроти, виріс майже на ту ж величину. Природно, свою роль тут зіграли катастрофічні валютні коливання, що терзали Україну весь 2023 рік.

Майже ніхто з операторів в умовах високої волатильності економіки не зважився відразу підняти ціни на свої послуги або переглянути умови укладених

контрактів, які найчастіше підписуються із клієнтом на півроку, рік і навіть більше (згодом тарифи, звичайно ж, були переглянуті). При цьому оплата вироблялася завжди в національній валюті. Навіть у випадку прив'язки до долара для розрахунку, як правило, ураховувався курс, що існував на момент укладання договору (у багатьох випадках близько 40 грн. за $1). Тому навіть в умовах припливу клієнтів і збільшення гривневого доходу його доларовий еквівалент скоротився. Як очікується, така тенденція продовжиться й в 2024 році, притім що більшість операторів уже встигли підняти ціни на послуги – у середньому в півтора-два рази. Але далеко не всі оператори прив'язують вартість сервісів до курсу долара США.

Як відзначає Олександр Кариченский, директор по розвитку бізнесу компанії «Анті Медіам» («Парковий»): «Компанія прийняла рішення не переглядати ціни по укладених контрактах. Вартість традиційних послуг дата-центра не змінилася. У той же час нові договори на «хмарні» послуги полягають за індексованими цінами, але без прив'язки до змін валютного курсу».

Зате «хмарний» сегмент переживає явний розквіт. В 2023 році обсяг продажів українськими комерційними дата-центрами «хмарних» сервісів, за даними дослідження «СІБ», склав $4,6 млн., або 55 млн. грн. У відносних показниках ріст просто феноменальний – 65% у доларовому еквіваленті й 150% у національній валюті (останнє, звичайно, також результат істотної девальвації гривні). Варто підкреслити, що тут мова йде тільки про доходи, отриманих українськими операторами, у той час як багато клієнтів користуються послугами закордонних сервісів-провайдерів, таких як Microsoft, Amazon, Google і багатьох інших. Скільки грошей іде, таким чином, за рубіж, підрахувати не представляється можливим, але, мабуть, не менше, ніж дістається українським «хмарним» компаніям. Дані отримані шляхом анкетування, безпосереднього опитування учасників ринку, у ході відвідування об'єктів (ЦОД), збору відомостей з відкритих джерел, обробки конфіденційної інформації, спілкування

з експертами й т.д. Для обробки даних і одержання результатів використовувалися методи статистичного аналізу.

Хто в лідерах?

Цього року при дослідженні українського сегмента комерційних дата-центрів ми вперше виділили в окрему категорію сегмент «хмарних» послуг. Від цього структура часток учасників ринку трохи змінилася в порівнянні з торішніми дослідженнями.

За станом на середину 2024 року дослідникам «СІБ» удалося нарахувати в Україні тридцяти шість фізичних дата-центрів, що надають комерційні послуги. Більше двадцяти площадок (у т.ч. всі самі великі) розташовані в Києві, п'ять у Харкові, по двох в Одесі й Дніпропетровську, по одній у Вінниці, Миколаєві й Львові.

Крім того, у нашій країні працюють більше десяти віртуальних операторів, що не мають своєї інженерної інфраструктури. Для зручності класифікації «СІБ» розділяє всі вітчизняні комерційні дата-центри на три категорії відповідно до їх максимальної корисної ємності: «великі» (можна розмістити понад 150 шаф), «середні» (50-150 шаф) і «малі» (до 50 стійок). Таким чином, у першу групу за станом на середину 2024 року попадають шість площадок: BeMobile, «Парковий», Gigacenter, De Novo, United DC і «Воля». Ще в Україні працюють біля десятка «середніх» комерційних дата-центрів і приблизно два десятки «малих». Критерієм відбору послужила сукупність різних факторів: технічні параметри комплексу, у т.ч. показники енергоефективності, перелік фактично надаваних послуг, відкликання клієнтів, відсутність аварій протягом тривалого часу й т.д.

Якщо розглядати український ринок у розрізі оренди фізичної інфраструктури, то однозначним лідером в 2023 році став комерційний ЦОД BeMobile, що займає майже третину всього сегмента. Це не дивно, адже дата-центр компанії є самим більшим в Україні як по корисній площі (1200 кв. м), так і по числу встановлених шаф – за підсумками 2023 року їх було понад 400, у

середині поточного року – уже 420 при максимальній ємності 450. Таким чином, очевидно, що можливості першої черги комплексу практично вичерпані, тому із другої половини 2024 року планується почати будівництво другої черги аналогічної місткості й потужності. У більше довгострокових планах оператора – створення ще й третьої подібної площадки. Для цього є й місце, і електрична потужність.

Оскільки єдиним напрямком бізнесу BeMobile є здача в оренду виділених зон, компанія споконвічно орієнтувалася на великих замовників. До середини 2024 року на площадці оператора розмістили свої фізичні ресурси вже сорок клієнтів, серед яких банки, роздрібні мережі, логістичні, промислові й страхові компанії, системні інтегратори, провайдери та ін. Тільки за останній рік сюди прийшли такі замовники, як «Приватбанк», «Метінвест», IBM і інші.

Друге місце в сегменті оренди фізичної інфраструктури ЦОД, як треба з діаграми на мал. 5, займає компанія De Novo, що в останні кілька років розвивається дуже динамічно по всіх своїх основних напрямках. У минулому послугами комерційного ЦОД компанії почали користуватися більше десяти нових великих клієнтів, у тому числі «ПУМБ», «Український Процесинговий Центр», «Ощадбанк», «ДТЕК», « Апк-інвест».

У зв'язку з ростом обсягу енергоспоживання в 2023 році наращивалась ємність ИБП і акумуляторних батарей, розширювалися можливості електророзподільної системи, також на об'єкті був інстальований третій дизель-генератор Cummins потужністю 1,6 Мвт. Крім того, були уведені в експлуатацію два нових модулі усередині дата-центра загальною ємністю 50 стійок.

Дата-Центр «Парковий» в 2023-м активно нарощував клієнтську базу. По даним, наданим «СІБ», число клієнтів виросло на 150%. У числі замовників послуг з'явилися такі відомі компанії, як Ciklum, «Метінвест», «Космо». У той же час комплекс ще не заповнений і на 15%. Цей фактор може зіграти на руку операторові, адже інші великі українські площадки заповнені майже повністю, а

виходить, їм буде потрібно час, щоб розширити вільний простір, а в «Паркового» воно є вже зараз.

Дата-Центр компанії «Воля» в 2023 році продемонстрував ріст клієнтської бази в межах 10%, що є відмінним показником, з огляду на, що в дата-центрі оператора встановлені тисячі клієнтських серверів. Наприкінці 2023-го на площадці була почата процедура по заміні й відновленню активного мережного встаткування й організації додаткових резервних каналів зв'язку. У рамках модернізації, зокрема, був установлений новий стек мережного встаткування Cisco.

Дата-Центр United DC був відкритий компанією «Об'єднані мережі України» у середині 2022 року, але вже до кінця 2023-го в компанії налічувалося більше п'ятисот клієнтів колокейшн (розміщення серверів у стійках оператора) і понад вісімдесят замовників, що орендують шафи цілком (або по 1/2). В 2023 році кількість клієнтів United DC виросло на 70% у порівнянні з підсумками 2022-го. З'явилися такі орендарі, як «Терабит», віртуальний датацентр «УДЦ» (який надає послуги VPS) і ін. Усього площадка містить кілька окремих блоків, які планується вводити в експлуатацію поетапно. У квітні 2024-го був запущений третій такий модуль і підготовлений електрична частина для підключення ще двох.

Важливо відзначити, що незважаючи на всі економічні складності, нові дата-центри в Україні всетаки з'являються, хоча й дуже повільно. Так, в 2023 році в Києві відкрився новий масштабний по українських мірках комплекс – «Гігацентр» (Gigacenter)загальною потужністю 1МВт і ємністю триста шаф. На даний момент послугами дата-центра користуються більше 60 клієнтів. Основні параметри комплексу наведені в таблиці. У переліку послуг оператора – колокейшн, оренда фізичних серверів, VDS і інші, допоміжні сервіси, у числі яких резервне копіювання даних, CDN (Content Delivery Network – територіально-розподілена мережна інфраструктура, що дозволяє оптимізувати

доставку й розподіл контента кінцевим користувачам в Інтернеті). Найближчим часом буде запущена послуга захисту від DDoS атак.

У сегменті середніх по розмірі комерційних дата-центрів варто відзначити ЦОД «Укртелекома», що торік істотно активізував діяльність і, як наслідок, залучив ряд великих замовників. Восени 2023 року ядро своєї інформаційної інфраструктури, а це більше 20 стійок з устаткуванням, розмістив тут «ДТЕК». Орендована площадка була приєднана до розгалуженої корпоративної мережі компанії по двох незалежних маршрутах на швидкості 20 Гбіт/с. Узимку 2024 року послугами ЦОД «Укртелекома» скористався телекомунікаційний оператор Vega, що розмістив у дата-центрі свої сервери, СЗД, мережне встаткування. Усього було встановлено сім шаф IT-устаткування (50 одиниць, які зайняли 140 юнітів). Фахівці «Укртелекома» також забезпечили зв'язок з корпоративною мережею Vega за допомогою двох волоконно-оптичних ліній передачі даних.

Ще один комерційний ЦОД – FREEhost, якому можна віднести до категорії середніх, – в 2023 році ввів в експлуатацію новий модуль дата-центра ємністю 900 юнітів і на 50% збільшив пропускну здатність зовнішніх каналів зв'язку – до 90 Гбіт/с. Цього року оператор планує зосередитися на заміні наявного встаткування більше енергоефективним, у тому числі будуть застосовані нові методи охолодження приміщень машинних залів.

До речі, цікавою тенденцією минулого року, що перекочувала в рік нинішній, стала поява «фантомних» комерційних дата-центрів. Це досить своєрідне явище, коли компанії, у спробах хоч якось із користю застосувати незадіяні IT-ресурси, що вивільнилися через різкий спад економічної діяльності, починають здавати в оренду фізичні або віртуальні сервери у своїх корпоративних датацентрах. Як правило, відбувається це не цілком легально, і свої послуги вони рекламують на тематичних форумах, дошках оголошень (таких як OLX), у соціальних мережах і інших методах.

Сьогодні вітчизняні компанії (потенційні замовники) поступово переборюють побоювання, пов'язані з «хмарами», серед яких – питання безпеки,

продуктивності, інтеграції, зміни провайдеру й т.д. До того ж багато хто навіть не цілком розуміють принципи роботи й оплати сервісів, не впевнені в їхній надійності. Однак уже чимало й таких, у тому числі великих, компаній, які успішно використовують «хмари» у своїй діяльності.

Наприклад, якщо взяти найбільшого на сьогоднішній день українського провайдеру подібних сервісів – компанію De Novo, те за станом на середину 2024 року в її «хмарі» працює більше п'ятдесятьох замовників. При цьому в 2023-м загальна кількість споживачів виросла на 68%. Тільки за минулий рік і першу половину 2024-го серед клієнтів з'явилися такі організації, як «ДТЕК», «Ощадбанк», «Укргазбанк», «Український Ритейл», «Тріумф Медиа груп», « Апк-Инвест», «СК АХА», Львівська міська рада. Як видно на мал. 6, De Novo, за даними дослідження «СІБ», стала лідером сегмента в 2023 році.

«Хмара» оператора надає послуги класу Infrastructure as a Service, виконано відповідно до концепції Trusted Cloud (рішення корпоративного класу з підвищеним рівнем надійності й захищеності) і розраховано в основному на потребі великих і середньо-великих інфраструктур, що обслуговують «важкі» додатки.

В 2023 році компанією були запущені нові сервіси, такі як можливість розмістити інфраструктуру на резервній площадці у Вільнюсі, Disaster Recovery as a Service, InterCloud Disaster Recovery (катастрофостійка пара на базі київської й вільнюської площадок). Крім того, з'явилися класи обслуговування, оптимізовані для різних завдань: Turbo (для додатків 1С) і Express (для невеликих інфраструктур). Цього року планується запуск віртуальних мережних пристроїв Cisco ASA і CSR (повний функціональний аналог апаратної реалізації). Також розглядається можливість створення цілої серії нових сервісів, орієнтованих на невеликих клієнтів. З технічної точки зору в 2023 році можливості «хмари» De Novo були збільшені в 2,4 рази.

Друге місце по обсязі продажів серед українських «хмарних» компаній займає «Парковий». Але в цьому випадку таку високу позицію операторові

забезпечив, головним чином, один клієнт – «Укрпошта», що в 2023 році, якщо вірити відкритим джерелам, замовив «хмарних» послуг на 17,1 млн. грн. (в 2022 році – майже на 8 млн. грн. за півроку). В 2024 році співробітництво було продовжено, цього разу, як очікується, «Укрпошта» заплатить 6,7 млн. грн. за користуванням ресурсами «хмари» у період з березня по грудень. Інші клієнти «Паркового», яких сьогодні чимало – Краматорський феросплавний завод, «Укрпапір», «Нерухомість столиці», «ЕланТВ» та інші, по даним «СІБ» використовують значно менший обсяг «хмарних» ресурсів. Торік «Парковий» запустив новий сервіс – Disaster Recovery (аварійне відновлення IT-інфраструктури клієнта з «хмари»).

У компанії «Воля», «бронзового» призера українського ринку «хмарних» сервісів, за станом на середину 2024 року налічувалося більше сімдесятьох клієнтів VoliaCloud. Серед них як невеликі замовники, так і великі компанії, які розміщають всю свою IT-інфраструктуру. Також «Воля» готується до запуску нового технологічного кластера для розширення можливостей свого «хмари», що говорить про попит і зростаючу зацікавленість у послузі з боку замовників. У числі останніх – розроблювачі ПЗ, рекламні агентства, студії дизайну, системні інтегратори, сервісні IT-компанії. В основному це представники СМБ. Останнім часом до числа клієнтів «хмарної» інфраструктури «Воля» приєдналися промислові підприємства, великі компанії, що займаються роздрібною торгівлею, і ін. У числі основних сервісів дата-центра «Воля» – інфраструктура як сервіс (IaaS) і послуга керованого сховища даних (Storage as a Service). Крім того, оператор розвиває й SaaS-рішення – сьогодні вже пропонуються такі сервіси, як бухгалтерія в «хмарі» (1С-online), CRM, відеоконференцзв'язок, ПЗ Microsoft. У липні 2024 року планується завершення процесу розширення технічних можливостей «хмари» VoliaCloud.

В 2024 році розвивати «хмарні» сервіси планує також «Укртелеком». Так, оператор уже запустив нову послугу за назвою «Хмарна АТС», що покликана не тільки повністю замінити офісну телефонну станцію, але й надати нові

можливості для взаємодії співробітників. Наприклад – використовувати смартфони/комп'ютери як офісний телефон, у тому числі поза офісом. Далі, за словами представників компанії, намічений запуск сервісів типу IaaS і SaaS. У тому числі таких послуг, як «віртуальне робоче місце», захист і резервне копіювання даних. Для сегмента СМБ розглядається можливість надання пакета програмних додатків, розгорнутих в «хмарі» для рішення різних бізнес-завдань. Правда, ніяких конкретних сервісів, так само як і строків їхнього виходу на ринок, поки названо не було.

Як показує практика, клієнти комерційних дата-центрів – це найбільш передові представники своїх галузей, а серед них виділяються ті, хто воліє «хмарні» послуги. Адже для того щоб їх використовувати, треба детально проаналізувати власні потреби й зрозуміти, що саме потрібно. Адже, як відомо, «хмари» – це не панацея, а лише один з інструментів досягнення мети організації. Так, він відкриває нові блискучі можливості, але лише тим, хто вміє ними користуватися. Наприклад, вартість використання «хмарної» IT-інфраструктури, як правило, нижче, ніж власної фізичної, але ціна міграції (особливо у випадку середніх і великих компаній) може виявитися занадто високою.

Але в цьому контексті слід зазначити, що усе більше компаній в Україні переходять від традиційної схеми оренди стійок і серверів до «хмарного» сервісам, хоча процес цей іде повільніше, ніж в економічно розвинених країнах. Згідно даним дослідження «СІБ», в 2023 році на традиційні послуги дата-центрів доводилося 80% усього ринку, але, скажемо, в 2021-м це було майже 95%!

Проте класична оренда місця в стійках формує майже половину ринку послуг комерційних ЦОД, ще близько 17% доводиться на оренду фізичних серверів, надаваних оператором. Різні «віртуальні» сервери, представлені в тому числі «еластичними» до обчислювальних навантажень VDS- і VPS-системами, дають ще 8% ринку. Тут у плані типів послуг всі більш-менш стабільно. Інша справа – «хмари». Ця сфера відносно нова для нашого ринку, тому варто зробити деякі уточнення. На сьогоднішній день у нашій країні популярні три базових

типи сервісів – IaaS, SaaS, PaaS, на які доводиться 93–95% продажів усередині країни. Що стосується внеску кожної із цих послуг у загальну структуру «хмарного» ринку, те якщо брати сервіси, надавані місцевими операторами, то по даним «СІБ», 76% доводяться на IaaS, 11% – на SaaS і 8% – на PaaS.

Інша частина ринку доводиться на додаткові й що розвиваються (на нашім ринку) сервіси, такі як Storage as a Service («СЗД як послуга»), Backup as a Service (створення й зберігання резервних копій в «хмарі»), Recovery as a Service («аварійне відновлення як послуга»), Database as a Service (підтримка СУБД організації в «хмарі оператора») і деякі інші.

«СІБ» не має точні відомості про те, які «хмарні» сервіси воліють використовувати українські користувачі на закордонних площадках, але очевидно, що в цьому випадку переважають моделі SaaS і IaaS.

У цілому, за прогнозами «СІБ», у наступні роки на українському ринку буде рости частка саме SaaS і тих сервісів, які сьогодні віднесені в категорію «інші», у той час як частка IaaS і PaaS буде знижуватися.

Як відзначає, наприклад, Максим Агєєв, генеральний директор компанії De Novo: «Український ринок «хмарних» сервісів перебуває в початковій стадії розвитку. Тому перелік перспективних послуг може містити в собі як завгодно багато найменувань. Найбільш затребуваними будуть залишатися сервіси, що ставляться до інфраструктурного шару. Хоча в перспективі 3-4 років цілком може вирости попит на послуги класу SaaS, особливо з боку невеликих клієнтів».

За словами Ірини Стельмах, керівника дата-центра «Воля»: «Українські замовники позитивно ставляться до запропонованих перспектив розвитку «хмарних» сервісів. Незважаючи на ситуацію в країні, ринок таких послуг демонструє позитивну динаміку. Важливо й те, що сьогодні інтерес до «хмарного» інфраструктурі спостерігається не тільки з боку компаній корпоративного сегмента, але й у представників СМБ, а це говорить про поширеність даного типу послуг. Таким чином, розвиток цього напрямку неминуче. Більше того, швидше за все, уже в 2024 році на ринку з'явиться ще

трохи «хмарних» операторів. Це явно говорить про ріст конкуренції, що, мабуть, приведе до якісного росту на ринку».

Про державне регулювання

Всі українські власники комерційних дата-центрів щиро сподіваються, що держава, принаймні в найближчі роки, не почне активно «регулювати» їхня діяльність. Проте від офіційного захисту, у першу чергу від представників тої ж держави, ніхто б не став відмовлятися. Все пам'ятають кількаразові епізоди за участю силових структур, під час яких у дата-центрах вилучалися сервери того або іншого клієнта. При цьому сама методика конфіскації приводила до порушення роботи всього комплексу, у тому числі й устаткування зовсім непричетних користувачів.

Проблема ще й у тім, що донедавна для дата-центра, по суті, не існувало ніякого офіційного визначення, яке б дозволяло однозначно вказати місце такого об'єкта на телекомунікаційному ринку, та й взагалі в українському правовому полі. Тому в 2024 році в складі НКРСІ активно діяла спеціальна робоча група по підготовці закону України «Про електронні комунікації», до складу якої були притягнуті представники найбільших вітчизняних комерційних дата-центрів. Одним з результатів діяльності групи стало те, що було дано перше визначення того, що ж таке ЦОД у термінах українського законодавства.

Але поки що положення власників комерційних дата-центрів украй уразливо. От лише кілька недавніх прикладів: в 2023 році СБУ здійснювала обшук в «Парковому» (конфіскація не проводилася, але було відключено IT-устаткування), в 2021-м податкова призупинила роботу серверів інтернет-магазина Rozetka в одному з українських комерційних дата-центрів, у тому ж році МВС блокувало ресурси файлообмінного сервісу ex.ua.

Нинішній, 2024 рік, не став виключенням. Так, у середині квітня СБУ провела вилучення серверів доменного реєстратора NIC.UA з харківського дата-центра Step Host. Хоча приводом до таких дій послужила «боротьба із сепаратистськими доменами» (які нібито розміщалися на серверах реєстратора).

При цьому конфіскація проводилася таким чином, що була порушена робота й зовсім непричетних компаній, у числі яких (по даним «BBC Україна») – благодійний фонд, що займається допомогою потерпілим від гуманітарної кризи в Україні; фонд допомоги онкобольним; юридична організація, що надає правові консультації учасникам АТО; Госсельхозінспекція України; Керування Госмістопромнадзора в Київській області й багато хто інші.

По даним NIC.UA, непрацездатними стали близько 30 тис. сайтів. При цьому всього було вилучено 139 одиниць техніки, включаючи сервери, мережне встаткування, комп'ютери відеоспостереження й т.д. (тобто брали всі підряд), а також вся документація, що вдалося знайти. Властиво, деталі події буквально по кроках описані й зняті на сайті власника дата-центра (http://steephost.com/).

Як повідомляє «BBC Україна», СБУ хотіла зупинити діяльність тільки п'яти сайтів, які, за даними слідчих, «використовувалися для проведення акцій інформаційної агресії з боку Російської Федерації».

При цьому, за словами виконавчого директора NIC.UA Андрія Хветкевича, ці сайти взагалі не припиняли роботу, оскільки розташовувалися на інші – закордонних площадках. Постраждав тільки один сайт, що використовував переадресацію в оператора, але й він заробив через годину або два – після того, як даний сервіс був відновлений для всіх клієнтів.

Пікантності ситуації додає той факт, що буквально за два тижні до цієї події, 2 квітня, Президент України підписав «Закон України № 191-VIII Про внесення змін до деяких законодавчих актів України щодо спрощення розумів ведення бізнесу (дерегуляція)» (прийнятий Верховною Радою 12 лютого). У ньому згадані, наприклад, що випливають виправлення в Кримінально-процесуальний кодекс України:

1) частину першу статті 159 доповнити абзацом іншим такого змісту: «Тимчасовий доступ до електронних інформаційних систем або їх частин, мобільних терміналів систем зв'язку здійснюється шляхом зняття копії

інформації, що міститься в таких електронних інформаційних системах або їх частинах, мобільних терміналах систем зв'язку, без їх вилучення»;

2) частину другові статті 168 доповнити абзацом іншим такого змісту: «Тимчасове вилучення електронних інформаційних систем або їх частин, мобільних терміналів систем зв'язку для вивчення фізичних властивостей, які мають значення для кримінального провадження, здійснюється лише в разі, якщо смороду безпосередньо зазначені в ухвалі суду».

Як бачимо – цілком здорові виправлення. Вони забороняють вилучення IT-устаткування, що не перераховано в рішенні суду, а крім того, не можна конфісковувати інформаційні системи, якщо можна зняти з них копії даних. Але, на даний момент ні зміна влади в країні, ні прийняття нового закону, ні дії робочої групи НКРСІ фактично не змогли поліпшити правового положення справ у галузі комерційних дата-центрів. Устаткування як вилучалося раніше, так і продовжує вилучатися. Будемо сподіватися, згодом ситуація виправиться…

У пошуках шляхів розвитку

Яка б не була нинішня ситуація на українському ринку комерційних дата-центрів, очевидно одне – сегмент продовжить розвиватися, незважаючи ні на що. Питання тільки в темпах такого розвитку, які прямо залежать від економічного становища країни. Як би те не було, але сьогодні не можна випустити з уваги сучасні тенденції розвитку комерційних дата-центрів, які актуальні не тільки в економічно розвинених державах, але й у нас у країні.

Наприклад, усе більше операторів у світі переходять на використання тільки SSD-дисків для надання послуг хостинга. Справа в тому, що традиційні масиви SATA і навіть SAS-дисків стають «вузьким» місцем у випадку активного використання СЗД безліччю клієнтів. Справа в тому, що сучасні моделі магнітних дисків реально здатні забезпечити не більше 100–200 операцій читання/запису в секунду (IOPS), а в SSD-Накопичувачів цей показник досягає декількох десятків тисяч. Відзначимо, що багато операторів і раніше використовували твердотільні диски, але, як правило, в обмеженому обсязі – тільки для кешування даних.

Другою важливою тенденцією є активний перехід на енергоефективное встаткування, прагнення до поліпшення PUE і застосування альтернативних джерел електроенергії. Даний тренд поступово докочується й до нашої країни у зв'язку зі стрімким зростанням тарифів на електрику. Нові площадки ЦОД будуються вже з використанням більше енергоефективних технологій (наприклад, прямого фрикулінга), а ті, які були створені раніше, прагнуть виконати модернізацію систем електроживлення й охолодження.

Слід також зазначити, що значні перспективи розвитку українського ринку комерційних дата-центрів сховані в «хмарному» сегменті. Складна ситуація в українській економіці ще як мінімум кілька років буде стимулювати компанії до скорочення витрат на IT і перекладу своїх інфраструктур на орендовані, у тому числі «хмарні», площадки. Також цілком відчутною ринковою силою в найближчі роки може стати сегмент СМБ, що набагато легше сприймає нові технології. Більше того, недавня поява технології 3G на мережах українських операторів зв'язку дозволить розширити аудиторію активних інтернет-користувачів за рахунок невеликих населених пунктів, а виходить, збільшиться число потенційних користувачів «хмарних» сервісів компаній СМБ-сегмента. Для них найбільш актуальні такі сервіси, як корпоративна пошта, зберігання даних, корпоративні сайти, віртуальні АТС, бухгалтерські й складські сервіси, веб-конференції й т.ін.

Як відзначає Євгеній Шерман, директор дата-центра FREEhost: «Поява в Україні 3G-мереж повинне вплинути на ринок. Відповідно до статистики, великий відсоток користувачів здійснює доступ до різних сервісів за допомогою мобільного Інтернету. Тому нова технологія доступу зажадає від провайдерів розширення каналів зв'язку, а крім того, більше якісний доступ в Інтернет буде стимулювати ріст числа інтернет-стартапів».

Ще одним великим, але, поки що, потенційним замовником послуг комерційних дата-центрів у нашій країні є державні організації. Адже, наприклад, у США і ЄС держсектор дуже активно використовує «хмари».

У кожному разі українським операторам зараз важливо не здавати свої позиції, оскільки ринок послуг комерційних дата-центрів має більші перспективи розвитку. І як тільки ситуація в українській економіці небагато стабілізується, він почне рости з подвоєною силою. До цього моменту треба бути готовими, оскільки є чимало закордонних операторів, готових забрати вітчизняного замовника. Якщо дозволити їм всерйоз вийти на ринок, то ці компанії зможуть досить швидко відрізати більшу частину ринку, особливо в сегменті «хмарних» рішень. Навряд чи вони легко відступляться від перспективного ринку, так що основні конкурентні баталії, схоже, ще спереду.

Огляд постачальників SDN

Концепція архітектури SDN і протоколу OpenFlow зародилася в Стенфордському університеті, дослідницькій групі якого треба було створити тестове середовище для експериментів з новими мережними протоколами. Будувати окрему мережу було дорого, тому вирішили задіяти наявну університетську мережу, у якій за допомогою прообразу SDN були виділені ресурси для випробувань.

Інтерес до SDN з боку неуніверситетських кіл першими виявили великі постачальники інтернет-сервисів, яким були потрібні високопродуктивні інфраструктури для організації взаємодії між десятками й навіть сотнями серверів у гігантських ЦОД. Традиційна трирівнева архітектура (доступ – агрегація – ядро) і необхідність робити безліч дій при обробці трафіку в кожному вузлі представлялися для них надлишковими й надмірними. Саме шість великих постачальників послуг – компанії Deutsche Telekom, Facebook, Google, Microsoft,Verizon і Yahoo – навесні 2011 року сформували організацію Open Networking Foundation (ONF) з метою розвитку технологій SDN у цілому й протоколу OpenFlow зокрема. Сьогодні членами ONF є практично всі основні постачальники мережного встаткування, включаючи Alcatel-Lucent, Brocade, Ciena, Cisco, Dell, Ericsson, Extreme Networks, HP, Huawei, IBM, Infinera, Intel,

Juniper Networks, Mellanox, Netgear, Nokia Siemens Networks, ZTE, а також лідери ринку систем віртуалізації VMware і Citrix.

Інтерес до SDN з боку великих постачальників інтернет-сервісів, хмарних послуг і власників мегаЦОД зрозумілийі: нові технології дозволять їм вирішувати свої завдання більш ефективно й, головне, за менші гроші. А що ж виробники мережного встаткування?

Найбільше завзято за освоєння нового «поля» узялися стартапи. Один з піонерів в області SDN компанія Nicira розробила контролер і платформу віртуалізації мереж (Network Virtualization Platform, NPV), а влітку 2024 року вона була куплена Vmware. Очевидно, що випливає претендент на поглинання ким-небудь із великих гравців – компанія Big Switch Networks, що випустила контролер Floodlight і засіб тестування OFTest. Згадаємо ще один стартап – компанію Pica8, що пропонує серію недорогих OpenFlow сумісних комутаторів.

Відомі компанії теж не залишаються осторонь. Так, наприклад, NEC розробила контролер і комутатори з підтримкою OpenFlow. IBM випустила лінію OpenFlow-сумісних комутаторів, а також співробітничає з NEC з метою забезпечення їхньої сумісності з її контролером. HP додала підтримку OpenFlow у велику кількість моделей комутаторів і розробляє контролер SDN. Багато провідних виробників комутаторів і маршрутизаторів, включаючи Brocade, Cisco, Extreme Networks і Juniper Networks, або вже оголосили про випуск OpenFlow-сумісних комутаторів/маршрутизаторів, або планують зробити це найближчим часом.

Але жоден із провідних постачальників мережного встаткування не оголосив SDN головною метою свого технологічного розвитку. Цього можна було очікувати, адже SDN відкриває можливість використання простих і дешевих комутаторів у якімсь ступені підриває бізнес цих компаній. Вони багато років удосконалювали функціонала своїх комутаторів і маршрутизаторів, саме він – їх головне «ноу-хау», основне джерело доданої вартості, а виходить, і прибутку. Думаю, одна із причин полягає в тому, що виробники не хочуть втрачати великих

замовників, які «перейнялися» ідеями SDN. Ще раз гляньте на список засновників ONF, додайте до них інших операторів зв'язку й ЦОД, які вже приєдналися до цієї організації (Colt, Deutsche Telekom, France Telecom Orange, Goldman Sachs, KDDI, Korea Telecom, Level 3 Communications, NTT Communications, SK Telecom, Swisscom, Telecom Italia), урахуйте зацікавленість, висловлену безліччю інших гравців – і стане зрозуміло, що вендорам просто не можна дистанціюватися від настільки представницької компанії. Інша, можливо, набагато більше важлива причина – у мережних вендорів просто немає альтернативи. Компанії Broadcom і Marvell, що ведуть виробники мікросхем для комутаторів, – теж члени ONF – уже оголосили про підтримку OpenFlow. А зібрати простенький комутатор, що працює по протоколі OpenFlow, з наявних наборів мікросхем може навіть не занадто спокушена в питаннях мережних технологій компанія. Що, властиво, і продемонструвала Google: мережа, що зв'язує її центри обробки даних, побудована на базі комутаторів SDN власної розробки. Якщо цим шляхом підуть і інші великі гравці, бізнес Cisco і інших традиційних виробників комутаторів може виявитися під погрозою.

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 кросплатформеність

– 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 розгортання централізовано програмованих мереж на основі технології SDN.

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

В 2024 році самою обговорюваною темою в мережній галузі стали централізовано програмувальні мережі (SDN). В 2024 році SDN залишиться «у моді», але справа до конкретних проектів в «звичайних» мережах (Google – не береться до уваги), тим більше в Україні, навряд чи дійде. Справа в тому, що концепція SDN повна протиріч. Винос «інтелекту» у контролер, здавалося б, «відкриває двері» простим (а виходить, недорогим) комутаторам, але це навряд чи вигідно виробникам мережного встаткування. Разом з тим при роботі з «вказівки» контролера буде потрібно виділити спеціальні ресурси для обробки записів у таблицях потоків – у мережах великих центрів обробки даних таких записів буде не просто багато, а дуже багато, а отже, і обчислювальні можливості мережного встаткування повинні бути досить значними. Це не занадто в'яжеться з тезою про «дешеві комутатори». Централізація керування й адміністрування – це прекрасно, але будь-яка централізація вимагає вживання додаткових заходів по забезпеченню відказостійкості, включаючи катастрофостійкість, що звичайно забезпечується децентралізацією «інтелекту». Список подібних протиріч можна продовжувати ще довго...

Масла у вогонь дискусії про SDN підлив один із творців протоколу OpenFlow (використовуваного для комунікації контролерів з комутаторами в мережі SDN) і засновник компанії Nicira Мартін Касадо. Наприкінці 2023 року він заявив, що використання цього протоколу для керування пересиланням трафіку комутаторами – помилкове рішення й що сама по собі технологія OpenFlow не здатна забезпечити ефективну підтримку мережною інфраструктурою віртуалізованих середовищ.

Але ж саме це завдання з усією актуальністю зараз устала перед великими сервісами-провайдерами, що націлилися на надання «інфраструктури як сервісу» (IaaS). На його думку, для віртуалізації мережної інфраструктури найбільш ефективна організація накладеної мережі, у якій контролер буде управляти не фізичними, а віртуальними (програмними) комутаторами на зразок Cisco Nexus 1000v, VMware DVS, IBM 5000v або Open vSwitch. Саме таке рішення реалізоване в продуктах Nicira/VMware.

Пошук ефективних інструментів для віртуалізації мережної інфраструктури стане, на наш погляд, головною тенденцією 2024 року. У цьому зв'язку рекомендуємо постежити за розвитком двох технологій: Virtual eXtensible LAN (VXLAN) і Network Virtualization using Generic Routing Encapsulation (NV-GRE). По суті, обидві вони забезпечують формування накладеної системи тунелів для функціонування великої кількості віртуальних мереж другого рівня (L2), здатних працювати в тому числі й через маршрутизируємі мережі L3. Обидві припускають для ідентифікації логічних мереж використовувати поля довжиною 24 біт, що дозволяє організовувати до 16 млн таких мереж. Це істотно більш того максимуму в 4096 віртуальних мереж, що надає класична технологія VLAN.

Прихильниками технології VXLAN виступають, зокрема, компанії VMware, Cisco, Arista, Citrix і RedHat. У таборі підтримки NV-GRE – такіі великовагові поїзди галузі, як Microsoft і Intel. На даний момент обидві технології перебувають у стадії розвитку, а документи IETF, у яких описані їхні основні алгоритми, носять статус «експериментальний» або «інформаційний». Проте саме із цими технологіями зв'язані надії на появу стандартних методів віртуалізації мережної інфраструктури.

В основі будь-якої віртуальної мережі повинна бути надійна (відказостійка) і високопродуктивна (висока швидкість + низька затримка) фізична мережна інфраструктура. Тут спостерігається перехід від традиційних трирівневих (доступ – агрегація – ядро) мереж з використанням протоколу STP або його різновидів до нових архітектурам комутаційних протоколів типу Fabric.

Їхню основу формують протоколи, що забезпечують одночасне використання всіх наявних ліній зв'язку (без яких-небудь блокувань для виключення зациклення) і загальносистемних крапок відмови. В області «фабрик» поки переважають фірмові (нестандартні) рішення, однак активна робота організацій IETF і IEEE над технологіями Transparent Interconnection of Lots of Links (TRILL) і Shortest Path Bridging (SPB) дає певну надію на те, що вже в 2024 році з'являться інтероперабельні рішення.

Що стосується швидкості окремих каналів, те, згідно із прогнозом Dell'Oro, в 2024 році більше половини нових серверів будуть підключатися за технологією 10Gb, а виходить, варто очікувати різкого підвищення попиту на комутатори з такими портами. Відповідно, на рівні агрегації (або в з'єднаннях між комутаторами в комутаційних «фабриках») будуть затребувані комутатори з магістральними портами 40Gb, число яких у світовому масштабі буде вимірятися вже сотнями тисяч. Однак на ділянці між серверами й комутаторами доступу технологія 40Gb почне застосовуватися тільки з 2024 року. Що стосується рішень 100Gb, те їхнє впровадження істотно стримується дорожнечею, а також більшими розмірами й високим енергоспоживанням існуючих трансіверів CFP. Поява в 2024 році компактних і енергоефективних трансіверів CFP2 (як очікується, їхнє споживання буде приблизно в чотири рази менше, ніж в CFP) зрушить із мертвої крапки використання 100-гігабитного Ethernet.

Масове впровадження 10-гігабитних інтерфейсів вселяє надію й на те, що більш активно буде використовуватися поки явно пробуксовуюча технологія FCoE. На даний момент це єдиний варіант конвергенції мережних інфраструктур локальних мереж і SAN, що чекають багато замовників. Саме необхідність підтримки відразу декількох мережних інфраструктур і технологій викликає стурбованість більшої частини українських фахівців. У цьому випадку вся стандартизаційна база для FCoE уже прийнята, однак навіть ті представники виробників, які у своїх презентаціях ратують за конвергенцію, у кулуарах конференцій не радять замовникам поспішати із впровадженням цієї технології.

Багато років перебуваючи «у тренді», FCoE поки так і не одержала широкого визнання.

Федерація хмар і оркестрація

Поняття «мережа для хмари» у значній мірі залежить від того, який зміст вкладати в терміни «хмара» і «мережа». Нерідка хмару розглядають як якусь інфраструктуру з ефективним масштабуванням сервісів і обмежуються контекстом ЦОД, однак більше правильним варто вважати розширений підхід і орієнтуватися на такі масштабні реалізації, як Amazon або Azure. У цей час для подібних хмар не передбачається єдине планування ресурсів (обчислювальних, мережних і сховищ даних), а також відсутні спеціальні убудовані засоби підвищеної відказостійкості й інструменти, що дозволяють управляти топологією віртуальної мережі. І взагалі через особливості своєї структури вони не призначені для використання NFV.

Для реалізації NFV у масштабному розподіленому середовищі необхідна оркестрація відповідних сервісів у системі хмар. При цьому віртуалізація ресурсів і їхнє абстрагування від конкретного місця розташування стають набагато складніше, тому що потрібне погоджене планування. NFV у хмарі – це не просто віртуальна машина з екземпляром ОС, а значно більше складна конструкція, по суті спеціалізована платформа, на якій зараз фокусують свою увагу багато розроблювачів. Вона припускає створення високошвидкісного тракту між ВМ і мережею. Використання для цієї мети стека протоколів операційної системи не дозволяє домогтися необхідної швидкості обміну даними й реалізувати платформу для мережних сервісів, розташовуваних у розподіленій мережі.

За допомогою сучасних методів можна більш ніж на порядок прискорити даний тракт. Вони працюють в обхід стека протоколів ОС і використовують віртуальні комутатори.

NFV не зажадає відмови від уже розгорнутої мережної інфраструктури при розгортанні нових сервісів, так що «нове» може мирно співіснувати з

«старим». Таким чином, впровадити NFV буде простіше, ніж SDN, оскільки в NFV використовується стандартне апаратне середовище. Крім того, NFV дозволяє повернутися при необхідності до колишньої мережної інфраструктури, а це знижує ризики.

Для стикування NFV з SDN може знадобитися додатковий сервісний оркестратор, що у загальному випадку ускладнює мережа, збільшує її вартість і знижує надійність. Такий зовнішній оркестратор може повідомляти контролер про доступну пропускну здатність: контролери SDN повинні мати інформацію про канал і надавати додаткам відомості про трафіке й вільні ресурси.

Замість спеціалізованого мережного встаткування NFV припускає використання стандартних серверів з гіпервізором віртуалізації й переклад всіх мережних компонентів на рівень ВМ. Як наслідок, важливим завданням стає створення гіпервізора віртуалізації операторського класу, що дозволяє перебороти проблеми продуктивності мережного стека й забезпечити моніторинг ВМ із контролем продуктивності й функціональності. З огляду на обмежену продуктивність ВМ, потрібний максимально ефективний механізм балансування навантаження, і таким є мережа SDN на базі OpenFlow, що дозволяє використовувати фізичні порти, а не прошарок у вигляді Open vSwitch, продуктивність якої обмежена мережним стеком.

Компанія NEC, наприклад, розробила гіпервізор операторського класу на базі KVM (Carrier Grade HyperVisor, CGHV), що підтримує компоненти NFV. Реалізовані програмно на базі гіпервізора мережні компоненти (VNF) с використанням стандартного апаратного забезпечення дозволяють знизити капітальні й експлуатаційні витрати. Розроблювачам NEC удалося мінімізувати затримки в мережі й забезпечити високу доступність. Віртуальній машині, що працює під керуванням цього гіпервізора, на сервері x86 може бути наданий повнодуплексний канал 10Gb. Спеціальні механізми протоколюють що виконуються ВМ інструкції й контролюють активність запущених процесів і її продуктивність. При збої в роботі ВМ оркестратор запускає її копію, створюючи

нову віртуальну машину. Автоматичне усунення несправностей і відновлення системи дозволяють гарантувати високий рівень доступності (по даним NEC – «п'яти дев'яток»), скоротити середнє часу відновлення (MTTR) у порівнянні із традиційними мережами й знизити операційні витрати за рахунок автоматизації.

Оркестратору приділяється особлива увага, але, щоб реалізовувати функції ефективно й дешево, робота гіпервізора, мережі й оркестратора повинна бути погоджена. З огляду на ідеологію відкритих мереж, що лежить в основі SDN, зробити це непросто, що підкреслює важливість оркестратора. У цей час група NFV, що діє в рамках організації ETSI, займається стандартизацією інтерфейсів оркестратора з компонентами NFV і SDN, а також питаннями інтеграції із системами керування OSS/BSS.

Щоб гарантувати виконання вимог SLA, оркестрація повинна здійснюватися як на рівні кожної площадки, за керування якої відповідає окремий контролер, так і на рівні федерації хмар. Остання являє собою розподілену мережу площадок, кожна з яких має пул ресурсів для федеративного використання. Для роботи із цими ресурсами необхідно нове рішення – «метаоркестратор». Контролер SDN визначає набір правил для обробки пакетів і завантажує їх у комутатори. Самі ж контролери взаємодіють по протоколі BGP (як це реалізовано, наприклад, у проекті OpenDaylight і в протоколі BGP-LS з підтримкою OpenFlow). Одна з невирішених проблем SDN пов'язана з тим, що кілька контролерів SDN можуть завантажувати в комутатори логічно суперечливі конфігурації й правила. Такі конфлікти вкрай складно дозволяти в реальному часі (щоб уникнути затримок).

На ринку рішень SDN активно працюють як починаючі компанії, так і всіма визнані ветерани галузі. Поряд з поділом процесів передачі й керування даними, SDN припускає наявність уніфікованого інтерфейсу між рівнями керування й передачі даних і віртуалізацію її фізичних ресурсів.

У відповідності зі своєю концепцією SDN Cisco розробила API для взаємодії з мережними пристроями (віртуальні або фізичними), контролери й

агенти для комунікацій з керованими елементами мережі. Для програмування різних мережних пристроїв можна використовувати єдині інтерфейси Cisco API OnePK.

Через API з мережним пристроєм взаємодіє додаток. Воно може перебувати усередині контейнера віртуалізації ОС самого мережного встаткування або бути зовнішнім. Такі додатки вже пропонуються замовникам. Одне з них – агент OpenFlow для комутаторів, що підтримують контейнерну віртуалізацію, наприклад комутаторів рівня ЦОД Cisco Nexus. Внутрішня взаємодія агента з комутатором відбувається через API, а зовнішнє (з контролером) – по стандартному протоколі OpenFlow. У результаті комутатори Nexus можуть «спілкуватися» з контролером SDN – Cisco XNC. Контролер XNC на базі проекту OpenDaylight з підтримкою OpenFlow і відкритим SDK для створення додатків – власна розробка Cisco, пропонована як комерційний продукт.

За допомогою SDK замовники можуть створювати власні унікальні рішення, і подібні приклади вже є. Оскільки контролер XNC постачений і убудовані додатки від Cisco, замовники можуть застосовувати їх для керування трафіком і конфігураціями, а також моніторингу стану. Наприклад, одна з компаній створила спеціалізовану мережу для збору, моніторингу й аналізу трафіку: функціями передачі даних для наступного аналізу минулого оснащені звичайні комутатори Cisco, що обійшлося дешевше застосування спеціального устаткування.

Ще один приклад – реалізація в корпоративній мережі власних механізмів маршрутизації. У цьому випадку в контейнер установлюється додаток маршрутизації, його взаємодія із системою відбувається через API, а для комунікацій з контролером використовується протокол обміну повідомленнями. Кілька підприємств створили додатка для сегментації мережі й модифікації шляху передачі даних по розширених метриках.

Ряд замовників використовують для керування конфігурацією додатків у великих мережах (десятки тисяч комутаторів рівня доступу) «відкриті» агенти Chef і Puppet. Ці механізми застосовуються для відстеження конфігурації ПЗ на серверах і служать для керування нею. Однак їх удалося розширити для конфігурування мережних пристроїв, і в результаті вийшло добре масштабоване рішення.

Huawei, що працює над власною концепцією SDN і NFV із централізованим керуванням і поділом рівнів керування й передачі даних, цього року планує анонсувати нові технології й продукти. Її рішення SDN припускає використання стандартизованих «північних» (додатки) і «південних» (устаткування) інтерфейсів. Інтерфейси API повинні забезпечити інтеграцію нового обладнання в існуючу мережу й полегшити процес міграції.

Недавно Huawei випустила програмувальний «мережний процесор» (Ethernet Network Processor, ENP), на основі якого вже створені й будуть будуватися її рішення SDN, зокрема нова лінійка продуктів Agile Evolution. ENP – альтернатива ASIC – спрощує реалізацію різних схем обробки трафіку, безпеки і якості обслуговування. Програмуємість даних продуктів дозволяє оперативно аналізувати й змінювати формати кадрів і застосовувати до будь-якому трафіку правила переадресації.

На відміну від протоколу OpenFlow, розрахованого на IP, підтримка Protocol Oblivious Forwarding (POF) дає можливість працювати з будь-якими протоколами й нестандартними пакетами. У той же час POF сполучимо с OpenFlow для забезпечення взаємодії з рішеннями інших виробників. Таким чином, POF у сполученні з ENP гарантує незалежність від застосовуваного протоколу. Процесори ENP дозволяють впроваджувати нові протоколи програмним шляхом. Сторонні розроблювачі можуть задіяти відкритий вихідний код POF у своїх проектах.

Huawei буде пропонувати три види контролерів SDN: для операторських і кампусних мереж, а також хмарних ЦОД. Наприклад, у кампусних мережах SDN

допомагає здійснювати керування користувальницьким доступом, а в хмарних ЦОД – знижувати витрати на впровадження нових сервісів. Передбачається наявність платформи оркестрації Network Orchestration Platform з уніфікованим керуванням провідними й бездротовими мережами.

По запиті користувача оператор може легко змінити топологію віртуальної мережі. Програмувальні процесори й SDN дозволяють впроваджувати зовсім нові протоколи будь-якого рівня й правила переадресації, збирати статистику по окремих потоках трафіку й додаткам і тим самим оцінювати якість надаваних користувачеві послуг.

Компанія NEC, розробляючи рішення SDN для операторів, фокусується на чотирьох напрямках: оркестрації (система NetCracker уже є в портфелі рішень NEC) і підтримці операцій, рішеннях SDN для ЦОД і продуктах SDN для транспортних мереж. Останні дозволять віртуалізувати транспортні мережі й управляти їхнім конфігуруванням і виділенням ресурсів клієнтам за допомогою контролера SDN.

Одне із самих масштабних впроваджень NEC SDN – проект, реалізований NT&T і охоплює біля десятка ЦОД цього оператора. У вигляді комерційних продуктів уже доступний віртуальний шлюз для операторів стільникового зв'язка й рішення для віртуалізації абонентських пристроїв (CPE), що дозволяє, подібно VDI, перенести функції з такого пристрою CPE у ЦОД і тим самим здешевити і спростити впровадження нових сервісів.

З погляду філософії й технології в SDN нічого нового немає – це прояв того ж принципу декомпозиції, що використовувався при розробці ОКС-7 і інтелектуальних мереж. Поділ керування й комутації – відома ідея. Але чи буде вона вдалої у випадку SDN? Однозначної відповіді немає. Поки має місце протистояння великих компаній, і в кожної з них є прагнення як до закритості (своя SDN і пропрієтарний контролер), так і до відкритого підходу. Успіх SDN залежить від того, чи буде досягнутий баланс». Можливі кілька сценаріїв

подальшого розвитку SDN, наприклад: галузеве співробітництво у виборі кращих рішень або розробка інтерфейсів для взаємодії систем різних вендорів.

Тим часом технологія SDN уже готова до впровадження. Її використовують NT&T, Deutsche Telecom, Telecom Italia і інші великі оператори. Серйозні надії покладають на SDN і виробники комутаторів, адже впровадження будь-якого нового протоколу третього мережного рівня припускає величезний обсяг робіт по програмуванню – створенню систем оркестрації й керування IT- і мережною інфраструктурою. Провідні вендори відкривають «центри рішень» SDN, надають розроблювачам API і SDK.

Вітчизняні ж фахівці шукають можливості використання технології SDN для створення власних рішень в області комутації, що дозволяють перебороти залежність від закордонних постачальників. SDN – гарний варіант для реалізації в Україні власних розробок і рішень. Цей підхід надає величезні можливості для українських розроблювачів і компаній-стартапів. SDN приведе до появи нового ємного сегмента ринку ПЗ – програмного забезпечення для мережних додатків. І в нашої країни є гарна можливість виявити себе в цій перспективній області.

Прикладом федерації хмар могла б стати мережа наукових організацій. Вона дозволила б визначати вимоги по надаваних ресурсах (топологія, швидкості обробки й передачі даних, маршрутизація) при проведенні відповідних експериментів, насамперед в області Больших Даних.

При збільшенні обсягів мережного трафіку SDN можна значно здешевити і спростити масштабування мережі. Це серйозний аргумент на користь SDN, тому що навряд чи можна чекати зниження цін на традиційні рішення, а в міру ускладнення мережної інфраструктури, що обслуговує все більші потоки даних, неминуче буде рости і її вартість, що негативно позначиться на доходах операторів. Однак організації, що впроваджують SDN, повинні чітко розуміти, що вони хочуть одержати від цієї технології й наскільки готові ризикувати. Цю методологію можна реалізувати по-різному, але потрібно виходити зі своїх цілей і її можливостей.

Компанії розуміють, що, розгорнувши SDN уже зараз, вони одержать конкурентну перевагу й при масовому її впровадженні зможуть виявитися в лідерах. Саме тому дуже багато операторів і провайдери вже мають тестові конфігурації SDN, але до практичного використання програмно конфігуруємих мереж підходять із великою обережністю.

Іде боротьба технологій, рішень і платформ. Згодом «природний добір» приведе до стандартизації SDN, сумісності рішень різних вендорів. Так, процес стандартизації протоколу OpenFlow демонструє явні ознаки збіжності, оскільки останні його версії розрізняються несуттєво. Найближчим часом цей етап повинен завершитися. Впровадження SDN стимулює й випуск недорогих наборів мікросхем з реалізацій OpenFlow. Подібні продукти вже з'являються.

За SDN майбутнє – ця технологія допомагає вирішити безліч завдань. Однак у реалізаціях SDN можуть застосовуватися різні механізми, і згодом їхній спектр, імовірно, стане ще ширше. Безумовно, з'являться нові способи побудови мереж SDN. Швидше за все, будуть створюватися гібридні, комбіновані мережі, де сполучаються нові й традиційні підходи. Відповідно до підходів компанії NEC, в основі SDN лежить OpenFlow: цей протокол дозволяє перебороти ряд проблем, пов'язаних з масштабуванням, керуванням мережею, обмеженням кількості VLAN. І хоча OpenFlow є єдиним новим стандартизованим протоколом для SDN, ніщо не заважає використовувати старі механізми, наприклад BGP-LS.

Між SDN і OpenFlow не можна ставити знак рівності. У даного протоколу є своя ніша (у деяких областях – значна), однак на ринку існують рішення SDN, де OpenFlow зовсім не використовується. Наприклад, в Juniper Networks створене рішення SDN на базі MPLS, тобто замість OpenFlow можна цілком обійтися іншими засобами. Ідеальний комутатор SDN повинен настроюватися на довільний протокол і структуру заголовка (не обов'язково TCP/IP). Рішенням цього завдання зараз займаються українські й закордонні розроблювачі.

У найближчій перспективі стануть розгортатися переважно гібридні рішення. Більшість комутаторів SDN саме й передбачають такий гібридний

режим, допускаючи програмне перемикання між SDN і комутацією L2/L3. Тим самим знижуються ризики впровадження нової технології: замовник може повернутися до колишньої інфраструктури й спробувати піти іншим шляхом, наприклад розгорнути рішення NFV з баласувальникиками навантаження.

Переваги SDN полягають у стандартних відкритих API для мережних додатків і можливості створення операторами власного ПЗ для керування мережею. Дуже важлива екосистема додатків, формованих навколо тої або іншої платформи. Конкуренція платформ SDN – це конкуренція екосистем додатків. Саме екосистему буде оцінювати замовник при виборі платформи SDN. Такі екосистеми зараз тільки формуються.

3.2 Розробка структурної схеми

Досить успішно навчившись віртуалізувати обчислювальні ресурси, IT-галузь витратила чимало часу на пошук адекватного підходу для віртуалізації мереж, мережних сервісів і розподілених систем зберігання даних. Прорив намітився з появою концепції програмно конфігуруємих мереж. Сьогодні пропонується кілька підходів до її реалізації й ведеться безліч дискусій про те, який з них є більше перспективним.

Концентрація обчислювальної інфраструктури в центрах обробки даних (ЦОД) зажадала не тільки перегляду підходів до рішення завдань енергопостачання, охолодження й інших проблем, що традиційно виникають при організації високопродуктивних обчислень. Така зміна парадигми змусило заново проаналізувати всю архітектуру й організацію програмного забезпечення, насамперед – для віртуалізації ресурсів ЦОД.

Досить успішно навчившись віртуалізувати обчислювальні ресурси, IT-галузь витратила чимало часу на пошук адекватного підходу для віртуалізації мереж, мережних сервісів і розподілених систем зберігання даних (СЗД). Традиційний стік протоколів не пропонував потрібного рішення ні для гнучкого керування розподілом мережних ресурсів і якістю мережних сервісів, ні для

динамічного перерозподілу збережених даних у рознесених СЗД. Прорив намітився з появою концепції програмно конфігуруємих мереж (ПКС; Software Defined Network, SDN; у комп'ютерній літературі широке поширення одержав переклад цього терміна як «програмно визначаємі мережі»). Сьогодні пропонується кілька підходів до її реалізації й ведеться безліч дискусій про те, який з них є більше перспективним.

Перший підхід припускає виділення так званої площини керування (контролера) програмними засобами – віртуальними програмними комутаторами. Саме цей підхід уважається найбільш перспективним, хоча його практичне втілення й породжує безліч питань.

Другий підхід передбачає створення спеціальних апаратних програмувальних комутаторів. Він дозволяє автоматизувати керування фізичною мережею за допомогою додатків для контролера площини керування, а також істотно спростити й скоротити як капітальну складову витрат на створення мережі, так і операційні витрати на її експлуатацію. При цьому успадкована частина системи передачі даних може залишитися без змін.

Тепер розглянемо обидва підходи докладніше. Перший можна умовно розділити на два: один припускає реалізацію накладеної мережі (overlay) за допомогою програмних комутаторів і протоколів тунелювання, іншої – використання серверів агрегації трафіку й спеціального апаратного забезпечення.

Overlay. Реалізація SDN у вигляді накладеної мережі на базі віртуальних комутаторів і тунелювання (протоколи VXLAN, NVGRE та ін.). Основна ідея полягає в тому, що на тих серверах, де «крутяться» віртуальні машини, настроюються комутація між віртуальними портами й відображення віртуальних портів на фізичні, а вже фізичні порти зв'язуються між собою тунелями. Сам віртуальний комутатор програмується за допомогою спеціально виділеного ресурсу, що називається «програмно конфігуруємий контролер». По суті, це операційна система, що управляє, розподіляє, контролює й здійснює моніторинг ресурсів мережі.

SDN на базі серверів агрегації трафіку. Інший вид програмної реалізації концепції SDN припускає виділення спеціального сервера, на який за допомогою тунелювання заводяться канали передачі даних. Потім із цього сервера дані передаються далі під керуванням SDN-контролера із залученням механізму тунелювання.

Слід зазначити, що обоє описуваних підходу – це програмні способи поділу керування передачею даних і властиво передачі даних, тому їм властиві всі достоїнства й недоліки програмної реалізації. Основним недоліком, звичайно, є більше високі затримки при передачі даних у порівнянні з апаратною реалізацією.

Реалізація SDN у вигляді накладеної мережі дозволяє відокремити віртуальні конфігурації й топології мережі від нижчележачої фізичної мережі. За політику маршрутизації відповідає віртуальна мережа, за фізичною же мережею залишаються тільки функції доставки пакетів до місця призначення. Такий підхід дозволяє перебудовувати й модернізувати фізичну мережу в міру необхідності, не міняючи нічого на віртуальному рівні. Дана реалізація відрізняється гнучкістю, швидкістю впровадження й відновлення.

Її недолік полягає в тому, що система віртуальних мереж не може бути ізольована від іншої частини центра обробки даних. У цьому зв'язку слід зазначити, що в доступному для огляду майбутньому важко уявити собі підприємство, що буде користуватися тільки віртуальними мережами. Якщо залишиться успадкована мережна інфраструктура, то накладена мережа повинна буде взаємодіяти із зовнішніми мережами, а для цього будуть потрібні шлюзи з відповідними протоколами.

Концептуально SDN з використанням техніки накладення здатна задовольнити всі основні потреби мережної інфраструктури підприємства. Але, безумовно, це рішення ефективно насамперед там, де для реалізації віртуальної мережі виділений хоча б кластер серверів. У недалекому майбутньому, однак, масштаби накладених мереж, схоже, будуть такі, що зможуть не тільки охоплювати окремі інсталяції, але й поєднувати їх у загальну інфраструктуру.

Реалізація SDN на базі спеціальних апаратних комутаторів (з підтримкою протоколів Openflow, XMPP та ін.). Мережа SDN складається з комутаторів SDN, взаємодіючих з контролерами по протоколі OpenFlow. Комутатор SDN реалізує тільки функції комутації (forwarding) даних. Тому він являє собою дуже простий програмувальний пристрій, що вміє виконувати кілька нескладних команд. Як наслідок, він виявляється значно дешевше існуючих комутаторів і маршрутизаторів. Робота комутатора SDN полягає в тому, щоб виділити із вступника пакета даних заголовок. Якщо комутатор «знає», як обробляти пакети з такими заголовками, то діє по заздалегідь завантаженій у нього програмі. У противному випадку по захищеному каналі OpenFlow він відправляє запит на контролер, а той у відповідь по тім же каналі завантажує програму для обробки пакетів з відповідними заголовками.

Контролер являє собою сервер, роботою якого управляє мережна операційна система (МОС). Подібно традиційній операційній системі, МОС управляє всіма ресурсами мережі. У термінології SDN контролером прийнято називати МОС. На контролері працюють додатки, що програмно реалізують мережні сервіси по керуванню потоками даних, такі як маршрутизація, балансування навантаження, різноманітні протоколи, шлюзи, мережні екрани (Firewall), шифрування, DPI, NAT і т.п. Додаток на підставі інформації про мережу, одержуваної їм від контролера, що управляє даною мережею, формує набори правил, які контролер завантажує в належні комутатори.

Саме головне, що в контролері централізовані всі відомості про актуальний поточний стан мережі. Він завжди має актуальну інформацію про структуру й топологію мережі комутаторів, який управляє (програмує), і розподілі потоків трафіку. Це дозволяє оптимізувати просування пакетів даних, динамічно управляти потоками даних, а не окремими пакетами, балансувати навантаження, оперативно контролювати виконання вимог інформаційної безпеки в мережі.

Така організація керування мережею вирішує, крім іншого, проблему залежності від мережного встаткування конкретного постачальника, оскільки SDN використовує загальні абстракції на програмному рівні для пересилання пакетів, за допомогою яких мережна операційна система здійснює керування мережними комутаторами.

У кожного з перерахованих підходів є свої достоїнства й недоліки. До переваг реалізації SDN у вигляді накладеної мережі ставляться простота віртуалізації адрес і топології й незалежність від організації фізичної мережі (будь-яка мережа IP). Але при цьому виникають додаткові накладні витрати на виконання інкапсуляції й, як наслідок, проявляється цілий ряд недоліків: більше низька – у порівнянні із третім підходом – продуктивність, неможливість управляти якістю транспорту, висока складність контролю й налагодження інкапсульованого трафіку, а також значні обмеження конфігурації мережі.

Переваги реалізації SDN на базі OpenFlow:

– висока продуктивність завдяки відсутності накладних витрат на інкапсуляцію й роботу програмних комутаторів;

– керування якістю транспорту;

– гнучке керування трафіком;

– автоматизація діагностіки й усунення неполадок (troubleshooting);

– віртуалізація й абстракція;

– досить просте мережне встаткування.

Але ця реалізація припускає впровадження спеціального мережного устаткування.

Слід зазначити, що вже сьогодні всі традиційні лідери ринку, такі як Cisco, Extreme Networks, Juniper, Brocade, IBM, NEC, Huawei і деякі інші, пропонують для своїх клієнтів третій шлях, що передбачає використання досить дорогого спеціального апаратного забезпечення. Виготовлення якісних і швидких комутаторів традиційно є однієї з найдужчих сторін цих компаній, і свої частки ринку в цьому сегменті вони втрачати не мають наміру.

Нестандартні й недорогі рішення для реалізації концепції SDN пропонують починаючі компанії, яким важливо на даному етапі залучити до себе клієнтів і відвоювати частку ринку в традиційних лідерів. До таким можна віднести компанію Arista, утворену тільки в 2008 році, що пропонує не тільки комутатори, що працюють винятково з мережами SDN, але й власну операційну систему Extensible Operating System (EOS). З її допомогою інженери можуть програмувати мережі й створювати повністю ізольовані мережні сегменти для додатків.

У підсумку вибір між апаратною реалізацією SDN або її розгортанням за допомогою накладеної мережі клієнтові прийде робити самостійно залежно від наявного бюджету й бізнес-завдань. Виробники мережних рішень, у свою чергу, повинні забезпечити цю можливість, щоб клієнт не зволів більше молоді технологічні стартапи, що пропонують рішення за мінімальні гроші.

Стрімке зростання обсягів трафіку й зміна його структури, необхідність підтримки зростаючої армії мобільних користувачів, формування високопродуктивних кластерів для обробки Больших Даних і добре масштабованих віртуалізованих середовищ для надання хмарних сервісів – все це серйозно змінило вимоги до мережних середовищ. І всі частіше мережа перетворюється в обмежуючий фактор розвитку обчислювальної інфраструктури.

Головна проблема: традиційні мережі занадто статичні й тому не відповідають динаміці, властивої сучасному бізнесу, на відміну від серверів – чим останні зобов'язані технологіям віртуалізації. Сьогодні додатки розподілені між безліччю віртуальних машин, які інтенсивно обмінюються даними (що веде до росту трафіку захід – схід, що починає домінувати над традиційним для архітектур клієнт-сервер трафіком північ – південь). Для оптимізації завантаження серверів віртуальні машини часто мігрують, що міняє крапки «прив'язки» трафіку. Традиційні схеми адресації, логічного розподілу мереж і способи призначення правил обробки трафіку в таких динамічних середовищах стають неефективні.

Наприклад, при запуску нової віртуальної машини, реконфігуровані списків контролю доступу (ACL) на всіх мережних пристроях у великій мережі може зайняти кілька днів. Причина – орієнтація наявних інструментів керування на роботу з окремими пристроями: у найкращому разі автоматизація призначення параметрів поширюється на групу пристроїв, у яку входять представники одного модельного ряду конкретного виробника. У результаті адміністраторам доводиться витрачати масу часу на те, щоб вручну перенастроїти правила обробки трафіку на кожному окремому пристрої. Такої ж проблеми виникають із переконфігурацією механізмів QoS при додаванні в мультисервисну мережу нового додатка, наприклад відеозв'язку. Неприпустимо багато часу в більших мережах займають процедури по зміні параметрів захисту, що не дозволяє оперативно реагувати на виникаючі погрози.

«Клаптева» природа наявних засобів керування значно ускладнює масштабування сучасних мереж. Додаткові складності в частині масштабування створюють і обмеження по числу логічних груп. Наприклад, як відомо, стандартна технологія VLAN забезпечує підтримку тільки 4096 віртуальних локальних мереж, а при розгортанні хмарних сервісів IaaS комерційним ЦОД уже сьогодні потрібно набагато більше число віртуальних мереж. Представте, що послуги IaaS надаються сотні підприємств, у кожного з яких по сотні VLAN, – уже в цьому випадку число логічних мереж становить 10 тис.

Але ще більшу заклопотаність в IT-директорів викликає невизначеність щодо підтримки майбутніх додатків і сервісів. Чи зможуть інсталюємі сьогодні мережні пристрої забезпечити таку підтримку? Якою мірою майбутній розвиток мережі – а виходить, і бізнес компанії – буде прив'язано до продуктової стратегії обраного виробника комутаторів? Архітектура традиційного мережного встаткування робить цю «прив'язку» дуже міцної. SDN обіцяє істотно послабити, а те й повністю ліквідувати залежність замовників від технологій конкретного вендора.

Головна ідея SDN полягає у відділенні функцій передачі трафіку від функцій керування (включаючи контроль як самого трафіку, так і здійснюючу

його передачу пристроїв). У традиційних комутаторах і маршрутизаторах ці процеси невіддільні друг від друга й реалізовані в одній «коробці»: спеціальні мікросхеми забезпечують пересилання пакетів з одного порту на інший, а вишележаче ПЗ визначає правила такого пересилання, виконує необхідний аналіз пакетів, робить зміну службової інформації, що втримується в них, і т.д. Для визначення маршруту передачі або недопущення зациклення трафіку пристрою, звичайно, «спілкуються між собою», для чого розроблене безліч протоколів, таких як OSPF, BGP і Spanning Tree, але при цьому кожне функціонує досить автономно.

Відповідно до концепції SDN, вся логіка керування виноситься в так звані контролери, які здатні відслідковувати роботу всієї мережі (див. рисунок 3.1). Не можна сказати, що це революційно нова ідея: зв'язківці пам'ятають, що подібну логіку у свій час передбачалося реалізувати при модернізації телефонних мереж, результатом чого стала поява «інтелектуальних мереж», а потім і комутаторів класу Softswitch.

Відповідно до задуму розроблювачів, SDN дозволить програмувати мережа як єдине ціле, а адміністраторам не прийде займатися окремими пристроями. Головним стає контролер: він все бачить, все знає й роздає мережним пристроям інструкції з обробки трафіку. Самим пристроям більше не треба розбиратися в сотнях мудрих протоколів – досить додержуватися інструкцій контролера, а виходить, вони можуть бути простими й дешевими.

Реалізація концепції SDN на практиці дозволить підприємствам і операторам зв'язку одержати вендоронезалежний контроль над всією мережею з єдиного місця, що значно спростить її експлуатацію. Що не менш важливо, конфігурування мережі сильно спроститься й адміністраторам не прийде вводити сотні рядків коду окремо для різних комутаторів або маршрутизаторів. Характеристики мережі можна буде оперативно змінювати в режимі реального часу, відповідно, строки впровадження нових додатків і сервісів значно скоротяться.

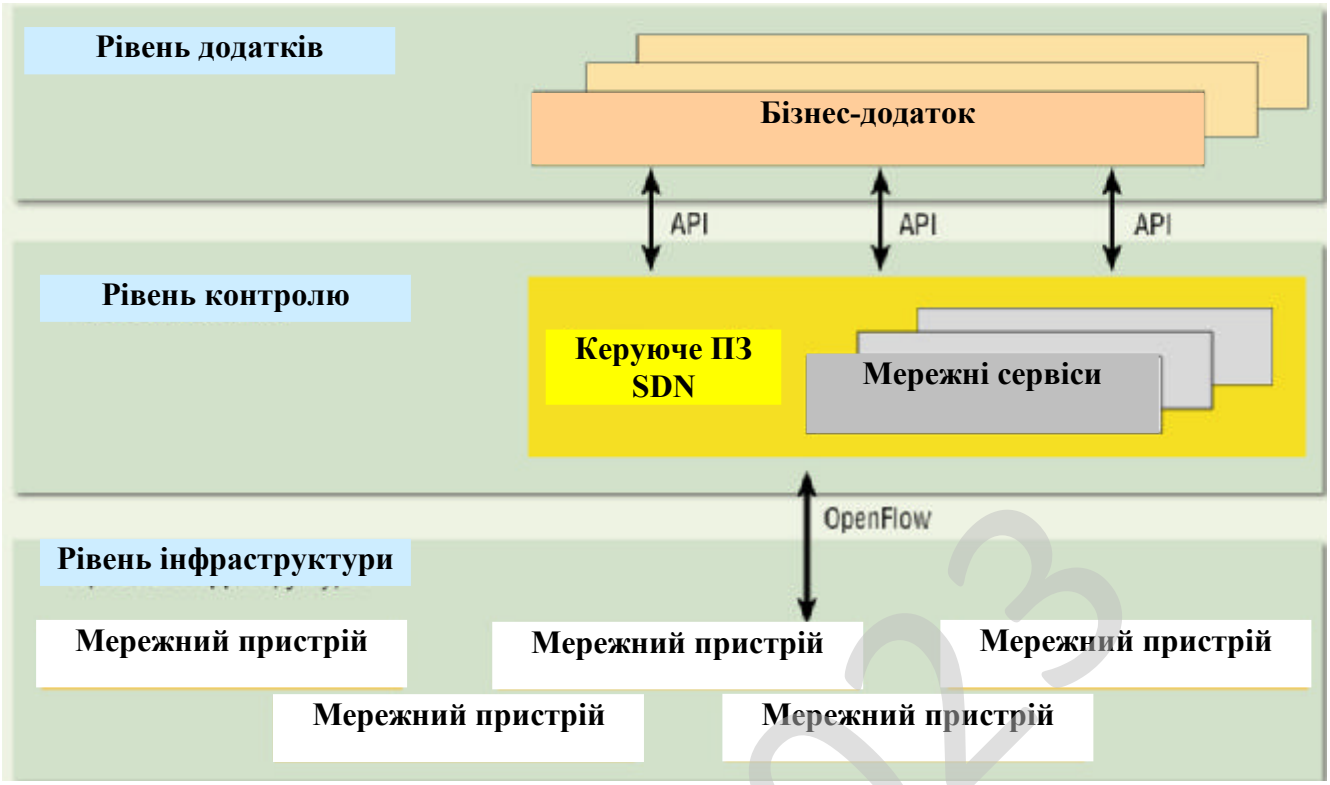

Рисунок 3.1 – Структурна схема системи

Основним елементом концепції SDN є протокол OpenFlow, що забезпечує взаємодію контролера з мережними пристроями (див. рисунок 3.1). На «північній» стороні контролер надає програмні інтерфейси (API), наявність яких дозволяє власникові мережі або сторонніх розроблювачів створювати додатка для керування мережею. Такі додатки можуть виконувати самі різні функції в інтересах бізнес-завдань (наприклад, контролювати доступ, управляти пропускною здатністю й т.п.), причому їхнім розроблювачам не треба знати деталі функціонування конкретних мережних пристроїв. Завдяки контролеру, вся мережа, що складається з безлічі різнотипних пристроїв різних виробників, з'являється для додатка як один логічний комутатор.

Як і виходить з назви, протокол OpenFlow при ідентифікації трафіку оперує поняттям «потоку». Ключовим елементом комутатора, що підтримує цей протокол, є таблиця потоків (Flow Table). Група стовпців у лівій частині таблиці формує поля відповідності, де зазначені характеристики потоків: це можуть бути

різні параметри, включаючи MAC- і IP-адреси відправника й одержувача, ідентифікатор VLAN, номера протокольних портів TCP і UDP, а також інша інформація. Ці дані за допомогою протоколу OpenFlow записує в таблицю комутатора контролер, він же визначає пріоритет різних потоків: чим вище пріоритет, тим вище відповідний запис у таблиці потоків.

Вхідні пакети перевіряються на відповідність зазначеним у таблиці параметрам. Якщо відповідність виявлена, до пакетів застосовується дія, що зазначена в наступному стовпці таблиці. Типовою дією є пересилання пакета на один або кілька вихідних портів. Крім того, комутатор може змінити вміст службових полів пакета, скинути його, направити для аналізу контролеру й т.д. У випадку якщо збіг не знайдений, пакет скидається або направляється контролеру, що визначить, як варто обробляти даний потік, і додасть відповідний запис у таблицю. Статистика по минаючий трафіку – число пакетів, байтів та ін. міститься у відповідні поля.

Використовуючи протокол OpenFlow, контролер додає, модифікує й видаляє записи в таблиці потоків. Крім того, він може запитувати в комутатора його характеристики й зібрану статистику, конфігурувати комутатор і його окремі порти.

Хоча в назвах організації ONF і її основного протоколу присутній слово «відкритий», як ви розумієте, це зовсім не синонім слову «безкоштовний». Члени ONF можуть використовувати протокол OpenFlow у своїх продуктах, але за саме членство стягується щорічна плата в 30 тис. доларів.

Поділ «площин» передачі й керування можна реалізувати взагалі не торкаючи наявної фізичної мережі – задіючи віртуальні комутатори на зразок Cisco Nexus 1000v, VMware DVS, IBM 5000v або навіть Open vSwitch з відкритим вихідним кодом. Програмування таких комутаторів за допомогою контролера дозволяє створити віртуальну мережу SDN поверх наявної фізичної інфраструктури. Деякі експерти розглядають цей підхід як альтернативу що розвивається ONF, але насправді, оскільки описувана схема не виключає

можливості використання стандартного протоколу OpenFlow, протиставляти її рішенням ONF не коштує.

Це далеко не умоглядна конструкція: подібне рішення в ЦОД, де для надання хмарних сервісів використовуються програмні комутатори Open vSwitch, контролери компанії Nicira і протокол OpenFlow. Все це функціонує поверх високопродуктивної фізичної мережі, побудованої на базі встаткування Extreme Networks. Головне завдання, що успішно вирішило – формування більше 1 млн ізольованих мереж на другому рівні (L2). За допомогою звичайних VLAN таке завдання не вирішується в принципі.

Якщо в такій мережі звичайні комутатори також будуть підтримувати OpenFlow, то до віртуальної мережі можна буде підключити й фізичні сервери. Керування такими комутаторами теж можна буде передати контролеру, якщо це не ввійде в суперечність із принципом поділу фізичної й віртуальної мереж. Цей приклад показує, що в моделі SDN конкретна реалізація комутатора – будь той фізичний пристрій або програма на гіпервізорі – не має принципового значення, головне, щоб він міг одержувати й виконувати інструкції контролера.

У рамках своєї стратегії SDN ряд виробників заявили про відкриття функціонала операційних систем своїх пристроїв через API. Дотепер настроювання мережних пристроїв вироблялося переважно через командний рядок або Web-інтерфейс. Але ці інструменти обмежені тією оболонкою програмування, що пропонує виробник. При наявності API можна використовувати більше широкий набір інструментів програмування й створювати додатка не тільки для настроювання мережного пристрою, але й для програмування мережного середовища в цілому. По суті, цей підхід є альтернативою SDN, при цьому він забезпечує доступ до більше широкого набору функцій мережних пристроїв, що потенційно дозволяє реалізувати більше можливостей, чим заповнення таблиці потоків.

Зокрема, наявність доступу через API безпосередньо до функціонала мережних пристроїв дозволяє системам типу VMware vCenter програмувати

мережа – наприклад, задавати настроювання VLAN у рамках загальних завдань по розгортанню й обслуговуванню віртуальних машин. Для багатьох виробників комутаторів інтеграція із системою vCenter надзвичайно важлива, оскільки вона спрощує й автоматизує процедури конфігурування мереж для середовищ віртуалізації VMware.

Одним із прикладів описуваного підходу служить анонсована влітку 2024 року компанією Cisco ініціатива Open Network Environment (ONE), у рамках якої вона представила кілька бібліотек API для своїх основних операційних систем: IOS, IOS XR і NX-OS. Використовуючи ці API, замовники можуть розробляти свої власні додатки, які встановлюються безпосередньо на пристроях або запускаються віддалено. Подібні можливості пропонують і інші виробники, скажемо Juniper (інтерфейс XML API) або F5 Networks (API-інтерфейс iControl).

Надання виробниками подібних API означає, що для реалізації централізованого програмування й керування зовсім необов'язково використовувати протокол OpenFlow. Крім того, такі ініціативи знімають із порядку денного теза про те, що комутатори можуть бути дуже простими й дешевими.

Переваги, які дає SDN, очевидні, причому не тільки для мереж у центрах обробки даних, але й для інших типів мережних інфраструктур. Централізоване керування мультивендорним середовищем, значне спрощення обслуговування й модернізації, скорочення часу на відновлення програмних кодів комутаторів/маршрутизаторів і впровадження нових сервісів – все перераховане важливо як для корпоративних мереж, так і для інфраструктур операторів зв'язку. Однак це не привід разом відмовлятися від переваг традиційного підходу, що розвивається десятиліттями, коли кожний мережний пристрій наділяється «інтелектом», достатнім для автономного функціонування.

Думаю, елементи SDN будуть впроваджуватися поступово, при цьому тільки компанії рівня Google зможуть дозволити собі перейти на комутатори, повністю керовані із центрального контролера. Швидше за все, частина

функціонала, наприклад по конфігуруванню й завданню політики безпеки, будуть виносити в контролери, при цьому самі комутатори залишаться досить «розумними», принаймні для того щоб брати участь у роботі протоколів визначення маршрутів або балансування трафіку.

Що ж стосується українськомовного терміна, то SDN – це, скоріше, «централізовано програмувальна мережа». І треба скоріше прийти до консенсусу щодо такого терміна. Нагадаю, що ISDN неформально розшифровувався як «I Still DoSn't Need it», тобто «мені цього поки не потрібно», причому саме таке відношення й визначило долю цієї технології в Україні. Сподіваюся, дуже розумні ідеї SDN чекає краща частка.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема розробленої системи. Принцип дії системи наступний. З ядра TCP/IP централізовано програмованих мереж на основі технології SDN, одержуємо дані про RTT-затримки для всіх поточних з'єднань (сокетів), ці дані зберігаються в буфері, з буфера послідовність даних (вектор) надходить на вхід аналізатора часового ряду, на виході ми одержуємо прогноз наступного значення RTT-затримки. На підставі прогнозу розраховується розмір наступного TCP-вікна (розмір буфера вікна перевантаження). За допомогою ядра TCP/IP відбувається керуючий вплив на пропускну здатність каналу.

За допомогою протоколу SNMP в MIB формуються якісні й кількісні дані про потік даних (трафік), через даний мережний адаптер. Аналізатор часових рядів у взаємодії зі стандартним SNMP агентом, витягає необхідні дані з бази MIB, обробляє й зберігає в БД для подальшого аналізу.

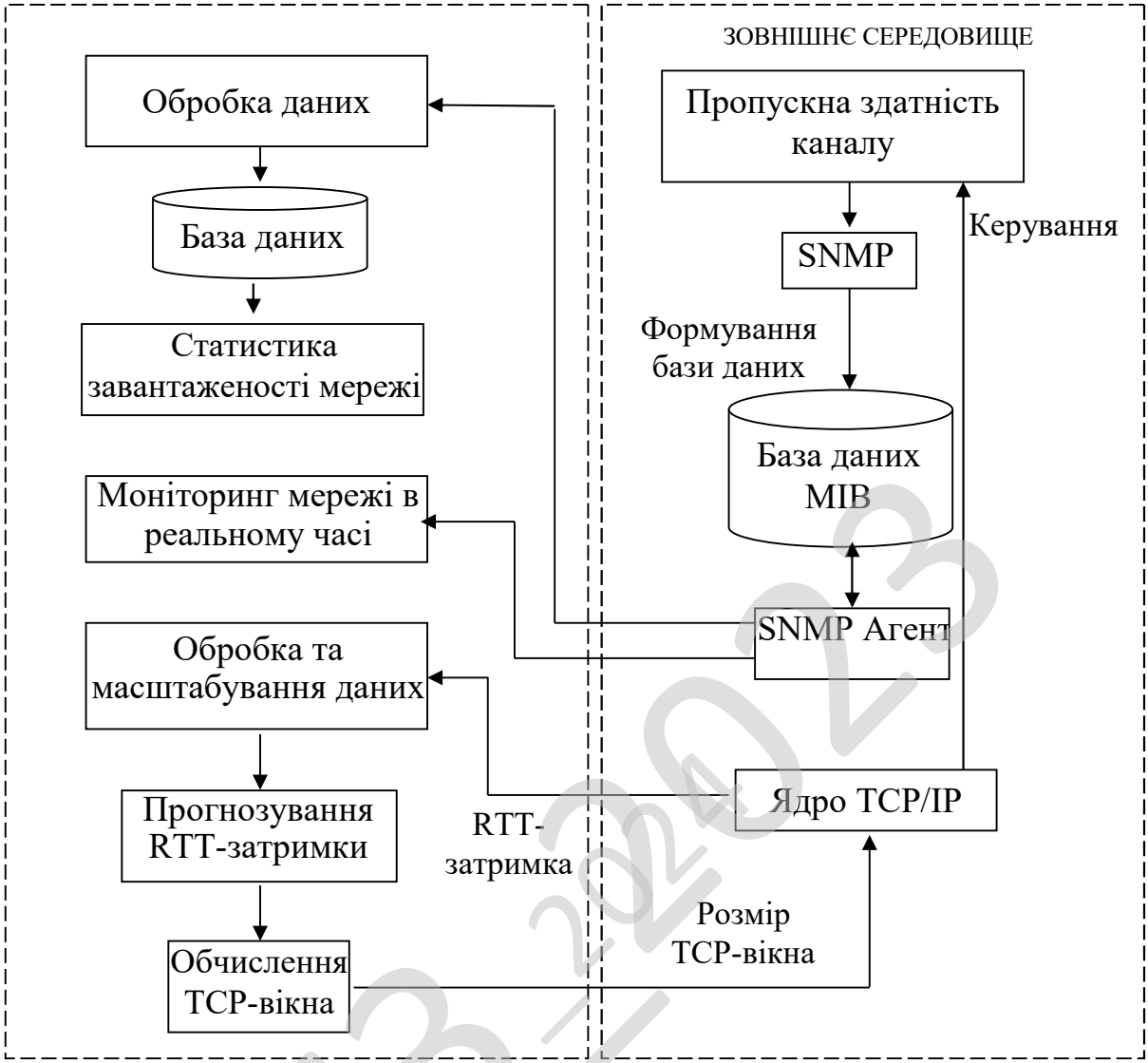

Рисунок 3.2 – Функціональна схема системи

Результати тестування (таблиця 3.1) розробленої системи на реальних даних показали високу точність прогнозу змін параметрів.

По отриманих параметрах RTT робиться прогноз наступного значення RTT. На підставі отриманого прогнозу виробляється розрахунок наступного TCP – вікна.

Таблиця 3.1 – Результати тестування розробленої системи

№	Реальна RTT, мс	Прогнозована RTT, мс	Помилка прогнозу, %
1	0,16	0,15	0,583
2	0,498	0,511	0,758
3	0,625	0,602	1,34
4	0,563	0,544	1,107
5	0.886	0,908	1,282
6	0,158	0,141	0,991
7	0,723	0,711	1,282
8	0,312	0,295	0,991
9	0,156	0,154	0,117
10	0,323	0,32	0,175
11	0,817	0,813	0,233
12	0,692	0,68	0,699
13	0,478	0,458	1,166
14	0,356	0,346	0,583
15	0,396	0,415	1,107
16	0,215	0,21	0,291
17	0,474	0,442	1,865
18	0,309	0,279	1,747

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

3.4 Розробка діаграми процесів

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

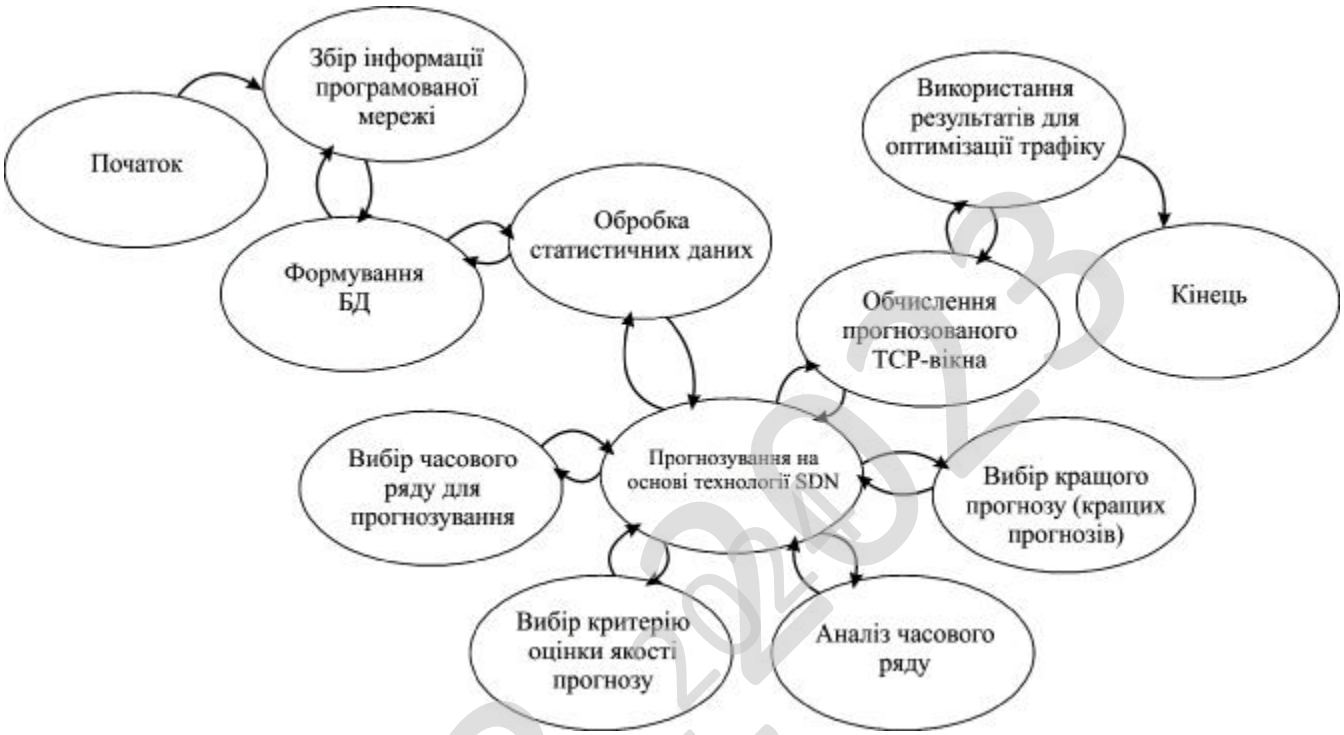

Рисунок 3.3 – Діаграма взаємодії процесів

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.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю розгортання централізовано програмованих мереж на основі технології SDN.

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

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

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

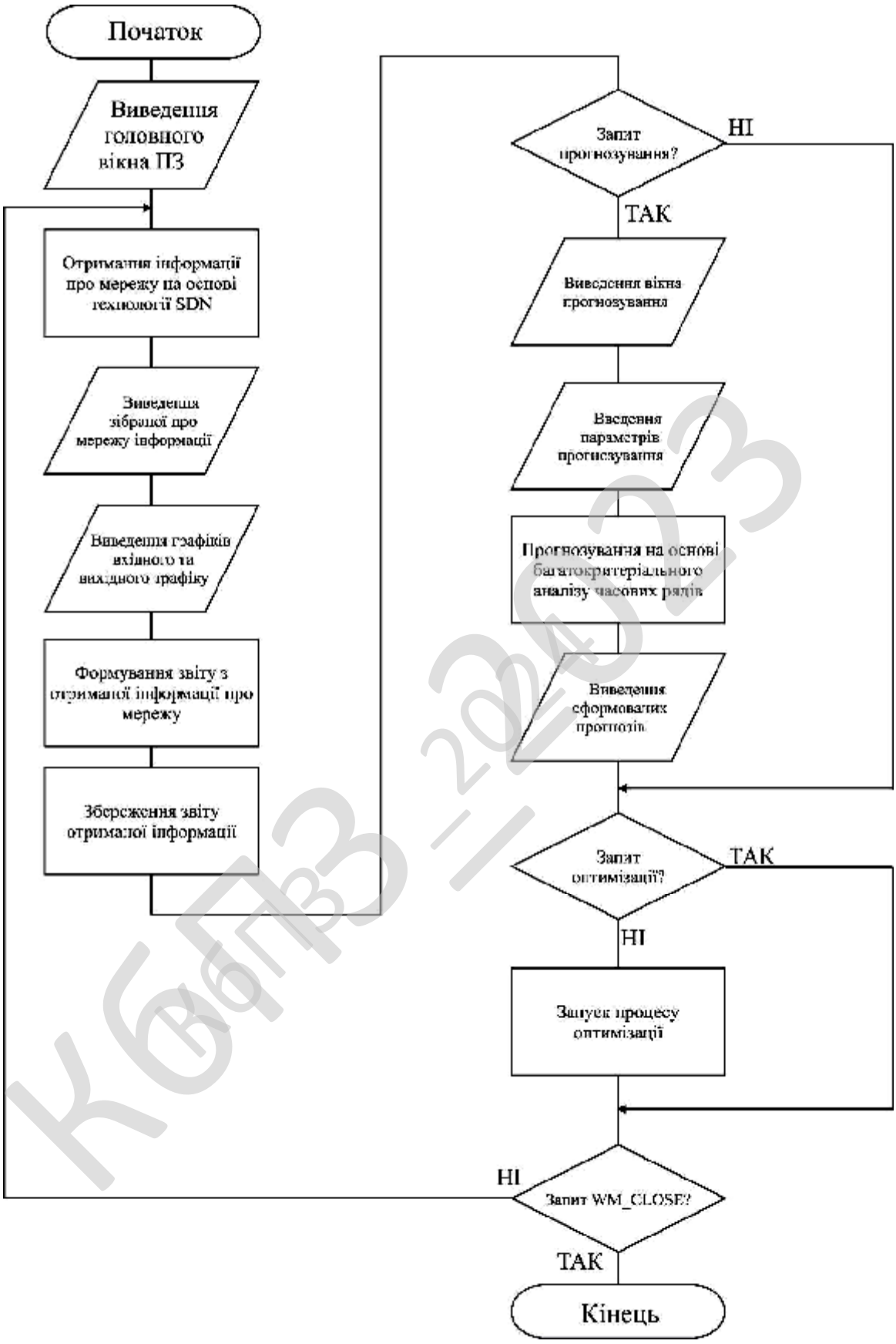

Рисунок 4.1 – Блок-схема основної програми

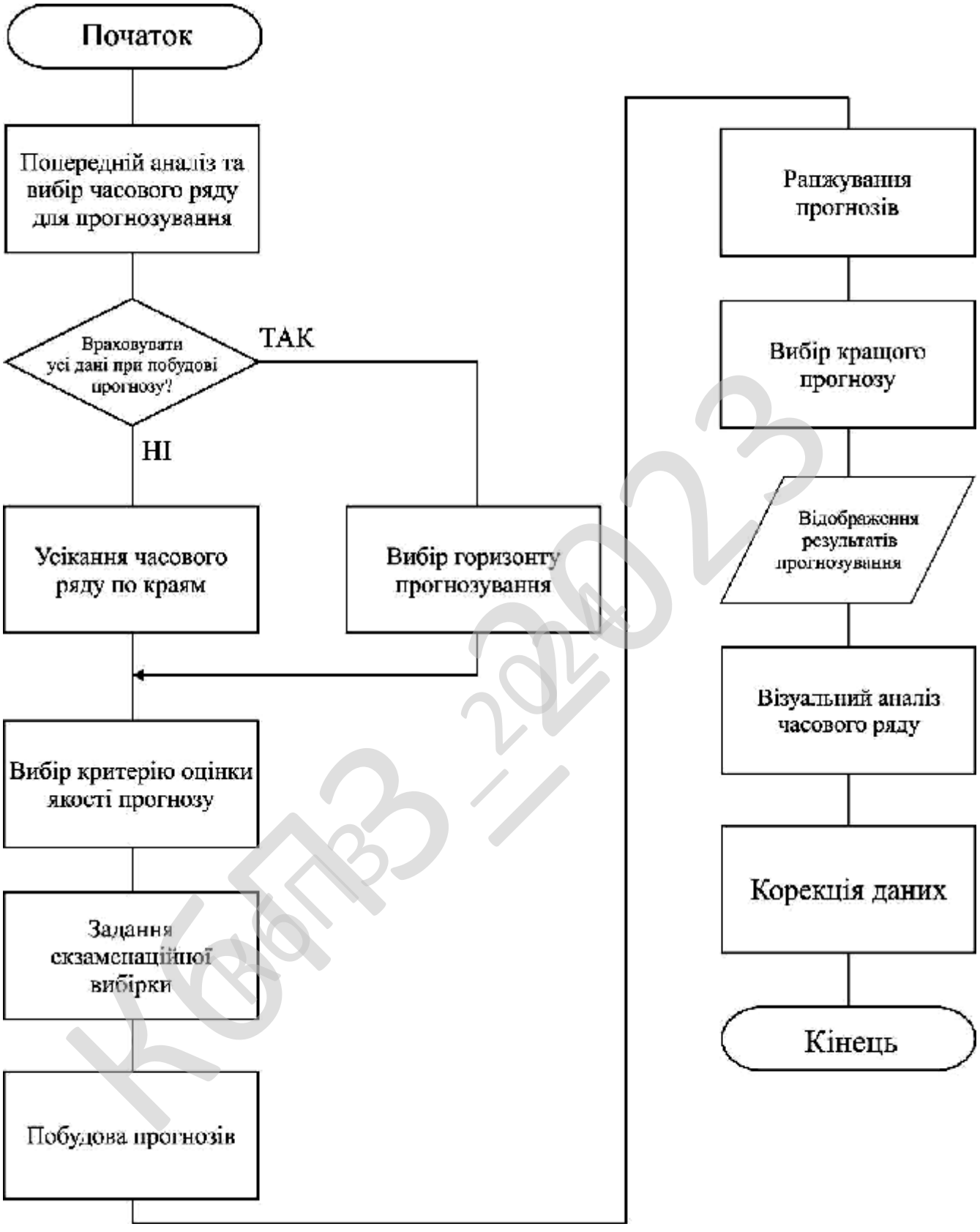

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

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

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

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

– Програмістам які реалізовують модулі інформаційної системи.

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

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML:

– діаграма компонент;

– діаграма об'єктів;

– діаграма розгортання.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

Діаграма об'єктів в UML це діаграма, що відображає об'єкти та їх зв'язки в певний момент часу. Діаграма об'єктів може розглядатись як окремий випадок діаграми класів, на якій можуть бути представлені як класи, так і екземпляри (об'єкти) класів. Схожою за змістом є діаграма взаємодії (collaboration diagram).

Діаграми об'єктів не мають власної нотації. Оскільки діаграми класів можуть відображати об'єкти, то діаграма класів, на якій відображено лише об'єкти, та не відображено класи, може вважатись діаграмою об'єктів.

Діаграма об'єктів відображає об'єкти та зв'язки в певний момент роботи програми. Об'єкти можуть містити інформацію про власні значення а не про описання. Для відображення загальних шаблонів об'єктів та зв'язків, що можуть багаторазово створюватись під час роботи програми, слід використовувати діаграму взаємодії, яка може відображати характеристики об'єктів та зв'язків. Екземпляр діаграми взаємодії створює діаграму об'єктів.

Діаграма об'єктів не відображає еволюцію системи під час роботи. Натомість, слід використовувати діаграми взаємодії з повідомленнями, або діаграми послідовності.

Діаграма розгортання (deployment diagram) це діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

Для того, щоб оцінити трафік мережі та його зміну з плином часу, тобто побудувати залежність завантаженості трафіку від часу, й отримати часовий ряд, треба взяти дані про рух у мережі з таблиць маршрутизації відповідних маршрутизаторів. Для цього використовується наступна процедура.

```
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: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpForwardTable(PTMibIPForwardTable(pBuf ), @TableSize, true
    );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  // заносимо у 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
          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] ) );
        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;
```

Після вибору критерію оцінки якості відбувається завдання екзаменаційної вибірки.

Для цього виділимо з вихідного часового ряду завантаженості трафіку навчальну вибірку $X_{об} = \{x(1), x(2),..., x(N-\tau)\}$, на підставі спостережень якої побудуємо оцінки значень часового ряду завантаженості трафіку на тактах з 1 по $N-\tau$, і прогнозні значення на тактах з $N-\tau+1$ по N. $X_{доп} = \{x'(1), x'(2),..., x'(N-\tau), x'(N-\tau+1), x'(N-\tau+2),..., x'(N)\}$. Потім, вибравши $к$ довільних крапок вихідного часового ряду, складемо з них екзаменаційну вибірку $X_{icn} = \{x(t_1), x(t_2),..., x(t_j),...,x(t_k)\}$, де $1 \le t_j \le N$.

З отриманого допоміжного прогнозу $X'_{доп}$ виберемо оцінки значень часового ряду на часових тактах, що ввійшли в екзаменаційну вибірку $X'_{icn} = \{x'(t_1), x'(t_2),..., x'(t_j),..., x'(t_k)\}$, $1 < t_j < N$. Зіставляючи отримані оцінки значень часового ряду X'_{icn}, зі значеннями екзаменаційної вибірки X_{icn}, оцінимо його якість, використовуючи різні критерії оцінки якості прогнозування. Таким чином, допоміжний прогноз буде перевірений на наявних даних.

Побудуємо прогноз X', на підставі всіх наявних спостережень часового ряду X, використовуючи при цьому тої же метод прогнозування, що й при побудові допоміжного прогнозу.

У зв'язку з тим, що оцінити якість прогнозу X на реальних даних не представляється можливим, будемо припускати, що його якість таке ж, як і якість допоміжного прогнозу.

Таким чином, залежно від критерію оцінки якості прогнозу й обраних прогнозних моделей, експерт вибирає крапки вихідного часового ряду, які необхідно включити в екзаменаційну вибірку, по крапках якої будуть перевірятися побудовані прогнози завантаженості трафіку. За замовчуванням екзаменаційними вважаються τ останніх тактів часового ряду.

За завданням екзаменаційної вибірки відбувається суто побудова прогнозів.

Для побудови набору конкуруючих прогнозів необхідно нажати кнопку «Побудувати прогноз» на панелі інструментів. Після розрахунку в нижній частині вікна системи прогнозування буде виведений ранжируваний список побудованих прогнозів із вказівкою використаної прогнозної моделі і її параметрів, значення критерію якості оцінки прогнозу, а також величина критерію «середня помилка». Цей критерій дозволяє наочно представити й зрівняти якість прогнозів, побудованих на підставі різних по масштабі даних, оскільки в ньому здійснюється перехід до відносних величин, що дозволяє експертові легко інтерпретувати величини помилок.

Після відображення результатів прогнозування відбувається візуальний аналіз часового ряду адміністратором мережі, тобто користувачем розробленої програми.

У результаті візуального аналізу часового ряду завантаженості трафіку адміністратор мережі приймає рішення про корекцію навантаження на ті або інші вузли мережі, яку він обслуговує. Для цього він змінює динамічні таблиці маршрутизації у відповідних маршрутизаторах, або проводить перекомутування каналів у відповідних коммутаторах.

Після проведення перерахованих вище дій, необхідно ще раз запустити програму, для того, щоб пересвідчитися в тому, що оптимізація мережі, яка була проведена на основі попереднього аналізу часових рядів завантаженості трафіку, дала свої позитивні результати. Тобто мережа не є перезавантаженою.

Об'єктна модель системи прогнозування. При побудові інформаційної системи прогнозування завантаженості трафіку на основі багатокритеріального аналізу часових рядів (надалі підсистеми «Прогноз») застосовувався об’єктно-орієнтований підхід, у рамках якого й розглянемо особливості реалізації підсистеми «Прогноз».

У підсистемі «Прогноз» можна виділити наступні логічні групи об’єктів:

– об’єкти, що утворюють ядро системи прогнозування;

– об’єкти, що реалізують різні постановки задач прогнозування;

– об’єкти, що реалізують прогнозні моделі;

– інтерфейсні об’єкти.

4.2 Захист розробленого програмного забезпечення

Дані в програмі захищаються за допомогою використання алгоритму CAST-128 (або CAST5) у криптографії, це блоковий алгоритм симетричного шифрування на основі мережі Фейстеля, який використовується в цілому ряді продуктів криптографічного захисту, зокрема деяких версіях PGP і GPG і крім того схвалений для використання Канадським урядом.

Основні відомості

Алгоритм був створений в 1996 році Карлайлом Адамсом (Carlisle Adams) і Стаффордом Таваресом (Stafford Tavares) використовуючи метод побудови шифрів CAST, який використовується також і іншим їхнім алгоритмом CAST-256 (алгоритм-кандидат AES).

CAST-128 складається з 12 або 16 раундів мережі Фейстеля з розміром блоку 64 біта й довжиною ключа від 40 до 128 біт (але тільки з інкрементацією по 8 біт). 16 раундів використовуються коли розміри ключа перевищують 80 біт. В алгоритмі використовуються 8x16 S- блоки, засновані на бент-функції, операції XOR і модулярной арифметиці (медулярне додавання й вирахування). Є три різні типи функцій раундів, але вони схожі за структурою й різняться тільки у виборі виконуваної операції (додавання, вирахування або XOR) у різних місцях.

Хоча CAST-128 захищений патентом Entrust, його можна використовувати в усьому світі для комерційних або некомерційних цілей безкоштовно.

Опис

CAST – це популярний 64-бітовий шифр, що допускає розміри ключа аж до 128 біт

Алгоритм CAST використовує 64-бітовий блок і 64-бітовий ключ. CAST стійкий до диференціального й лінійного криптоаналізу. Сила алгоритму CAST укладена в його S-блоках. В CAST немає фіксованих S-блоків і для кожного додатка вони конструюються заново. Створений для конкретної реалізації CAST S-блок уже більше ніколи не міняється. Інакше кажучи, S-блоки залежать від реалізації, а не від ключа. Northern Telecom використовує CAST у своєму пакеті програм Entrust для комп'ютерів Macintosh, PC і робочих станцій UNIX. Обрані ними S-блоки не опубліковані, що втім не дивно.

CAST-128 належить компанії Entrust Technologies, але є безкоштовним як для комерційного, так і для некомерційного використання. CAST-256 – безкоштовне доступне розширення CAST-128, яке ухвалює розмір ключа до 256 біт і має розмір блоку 128 біт. CAST-256 був одним з первісних кандидатів на AES.

Опис алгоритму

CAST-128 заснований на мережі Фейстеля. Повний алгоритм шифрування викладений у наступних чотирьох кроках:

ВХІД: текст m_1 ... m_{64}, ключ K = k_1 ... k_{128}.

ВИХІД: зашифрований текст c_1 ... C_{64}.

1. (розгорнення ключа) становить 16 пару підключів {Km_i, Kr_i} отриманих з K (див. розділи Пари раундових ключів і Неідентичні раунди).

2. (L_0, R_0) <- (m_1. .. m_{64}). (Розділяє текст на ліву й праву 32-бітні половини L_0 = m_1 ... m_{32} і R_0 = m_{33} ... m_{64}).

3. (16 раундів) for i from 1 to 16, обчислити L_i і R_i у такий спосіб: L_i = R_{i-1}; R_i = L_{i-1} ^ F(R_{i-1},Km_i,Kr_i), де F визначена в розділі «Пари раундових ключів» (F має тип 1, тип 2, тип 3 або, залежно від i).

4. c_1 ... c_{64} <- (R_{16}, L_{16}). (Міняємо остаточні блоки місцями L_{16}, R_{16} і поєднуємо, щоб сформувати зашифрований текст.)

Розшифруванні збігається з алгоритмом шифрування, наведеним вище, крім того, що раунди (i, отже, пари підключів), використовуються у зворотному порядку, щоб обчислити (L_0, R_0) з (R_{16}, L_{16}).

Пари раундових ключів

CAST-128 використовує пару підключів за раунд: 32-бітні величини Km використовується в якості "маскування" ключа й Kr використовують як "перестановки" ключа, з яких використовуються тільки початкові 5-біт.

Неідентичні раунди

Три різні типів функції використовуються в CAST-128. Типи виглядає в такий спосіб (де "D" є вхідними даними у функцію F і "Ia"-"Id" є найбільш значимий байт – найменш значимий байт I, відповідно). Зверніть увагу, що "+" і "-" додавання й вирахування по модулю 2 ** 32, "" є побітове XOR і "<<<" є циклічним зрушенням уліво.

Раунди	$I = ((Km_i + R_{i-1}) <<< Kr_i)$
1,4,7,10,13,16	$F = ((S1[I_a]\ S2[I_b]) – (S3[I_c]))+ S4[I_d]$
Раунди	$I = ((Km_i \wedge R_{i-1}) <<< Kr_i)$
2,5,8,11,14	$F = ((S1[I_a] – S2[I_b]) + (S3[I_c]))\ S4[I_d]$
Раунди	$I = ((Km_i – R_{i-1}) <<< Kr_i)$
3,6,9,12,15	$F = ((S1[I_a] + S2[I_b])\ (S3[I_c]))- S4[I_d]$

Поля заміни

CAST-128 використовує вісім полів заміни: поля S1, S2, S3 і S4 раундові функції полів заміни, S5, S6, S7 і S8 є ключами розгорнення полів заміни. Незважаючи на те, що 8 полів заміни вимагають у цілому 8 Кбайт для зберігання, зверніть увагу на те, що тільки 4 Кбайта потрібні під час фактичного шифрування / дешифрування, тому що генерація підключа звичайно робиться до будь-якого введення даних.

Ключі розгорнення

Представимо 128-розрядний ключ у вигляді x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf, де x0 старший байт, і xf молодший байт.

Представимо z0..zf проміжними (тимчасовими) байтами. Si[] представляє поле заміни і і " ^ " представляє додавання по Xor'у.

Поля заміни формуються із ключа x0x1x2x3x4x5x6x7x8x9xaxbxcxdxexf у такий спосіб.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K1 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K2 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K3 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K4 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K5 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K6 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K7 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K8 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K9 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K10 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K11 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K12 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K13 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K14 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K15 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K16 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

половина, що залишається, ідентична тому, що дане вище, продовження від останнього створило x0..xf, щоб генерувати ключі K17 – K32.

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K17 = S5[z8] ^ S6[z9] ^ S7[z7] ^ S8[z6] ^ S5[z2]

K18 = S5[zA] ^ S6[zB] ^ S7[z5] ^ S8[z4] ^ S6[z6]

K19 = S5[zC] ^ S6[zD] ^ S7[z3] ^ S8[z2] ^ S7[z9]

K20 = S5[zE] ^ S6[zF] ^ S7[z1] ^ S8[z0] ^ S8[zC]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K21 = S5[x3] ^ S6[x2] ^ S7[xC] ^ S8[xD] ^ S5[x8]

K22 = S5[x1] ^ S6[x0] ^ S7[xE] ^ S8[xF] ^ S6[xD]

K23 = S5[x7] ^ S6[x6] ^ S7[x8] ^ S8[x9] ^ S7[x3]

K24 = S5[x5] ^ S6[x4] ^ S7[xA] ^ S8[xB] ^ S8[x7]

z0z1z2z3 = x0x1x2x3 ^ S5[xD] ^ S6[xF] ^ S7[xC] ^ S8[xE] ^ S7[x8]

z4z5z6z7 = x8x9xAxB ^ S5[z0] ^ S6[z2] ^ S7[z1] ^ S8[z3] ^ S8[xA]

z8z9zAzB = xCxDxExF ^ S5[z7] ^ S6[z6] ^ S7[z5] ^ S8[z4] ^ S5[x9]

zCzDzEzF = x4x5x6x7 ^ S5[zA] ^ S6[z9] ^ S7[zB] ^ S8[z8] ^ S6[xB]

K25 = S5[z3] ^ S6[z2] ^ S7[zC] ^ S8[zD] ^ S5[z9]

K26 = S5[z1] ^ S6[z0] ^ S7[zE] ^ S8[zF] ^ S6[zC]

K27 = S5[z7] ^ S6[z6] ^ S7[z8] ^ S8[z9] ^ S7[z2]

K28 = S5[z5] ^ S6[z4] ^ S7[zA] ^ S8[zB] ^ S8[z6]

x0x1x2x3 = z8z9zAzB ^ S5[z5] ^ S6[z7] ^ S7[z4] ^ S8[z6] ^ S7[z0]

x4x5x6x7 = z0z1z2z3 ^ S5[x0] ^ S6[x2] ^ S7[x1] ^ S8[x3] ^ S8[z2]

x8x9xAxB = z4z5z6z7 ^ S5[x7] ^ S6[x6] ^ S7[x5] ^ S8[x4] ^ S5[z1]

xCxDxExF = zCzDzEzF ^ S5[xA] ^ S6[x9] ^ S7[xB] ^ S8[x8] ^ S6[z3]

K29 = S5[x8] ^ S6[x9] ^ S7[x7] ^ S8[x6] ^ S5[x3]

K30 = S5[xA] ^ S6[xB] ^ S7[x5] ^ S8[x4] ^ S6[x7]

K31 = S5[xC] ^ S6[xD] ^ S7[x3] ^ S8[x2] ^ S7[x8]

K32 = S5[xE] ^ S6[xF] ^ S7[x1] ^ S8[x0] ^ S8[xD]

Маскування й перестановка підключів

Km_1, ..., Km_{16} 32-розрядні підключи маскування (один на раунд).

Kr_1, …, Kr_{16} 32-розрядні перестановки підключів (один на раунд); тільки молодші 5 бітів використовуються в кожному раунді.

for (i=1; i<=16; i++) { Km_i = Ki; Kr_i = K16+i; }

Змінний розмір ключа

CAST-128 Алгоритм шифрування був розроблений, щоб розмір ключа міг варіюватися від 40 до 128 біт, в 8-бітному кроці (тобто припустимі розміри ключа рівняються 40, 48, 56, 64..., 112, 120, і 128 бітам). Для змінної роботи розміру ключа специфікація наступні:

1) Для розмірів ключа до й включаючи 80 бітів (тобто, 40, 48, 56, 64, 72, і 80 бітів) алгоритм точно такої ж, але використовує 12 раундів замість 16;

2) Для розмірів ключа більше, чим 80 бітів, алгоритм використовує повні 16 раундів;

3) Для розмірів ключа менше, чим 128 бітів ключ доповнений нульовими байтами (у самих правих, або молодших, позиціях) до 128 биток (тому що розклад ключа CAST 128 ухвалює вхідний ключ 128 бітів).

Розшифрування

Розшифрування для CAST-128 відносно проста. Розшифрування працює в тому ж алгоритмічному напрямку, що й шифрування, починаючи із зашифрованого тексту як вхідних даних. При цьому підключ використовуються у зворотному напрямку.

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 розгортання централізовано програмованих мереж на основі технології SDN складається з наступних функціональних блоків:

– Функцій представлених у графічному вигляді.

– Розділу вікна графічного представлення прогнозованого вхідного трафіку.

– Розділу вікна графічного представлення прогнозованого вихідного трафіку.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

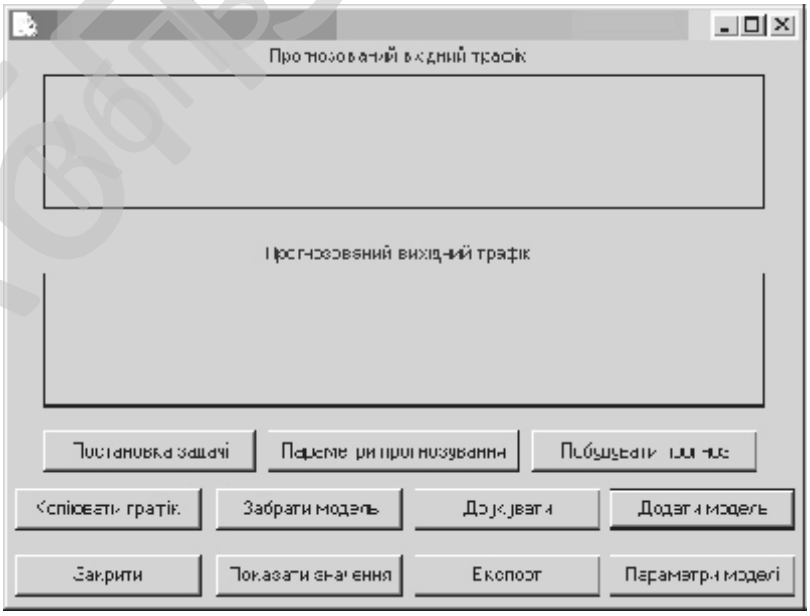

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

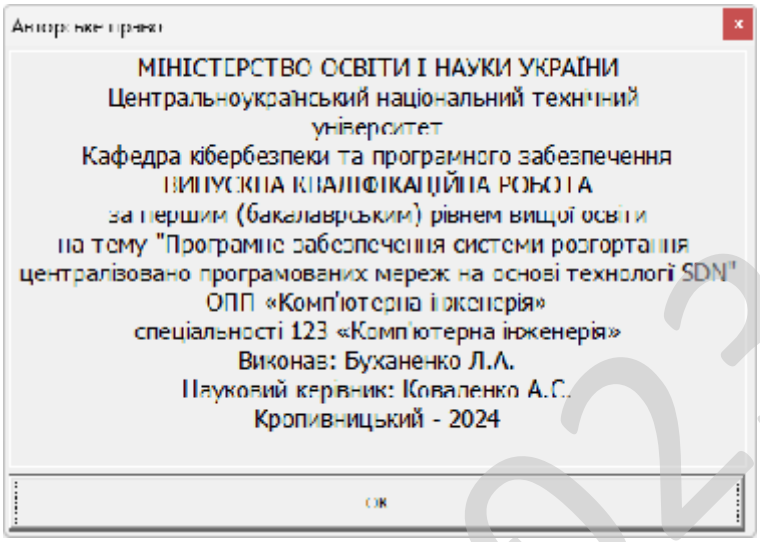

Рисунок 5.2 – Вікно розробника ПЗ

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 ITc).

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

– 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 – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права.

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

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію.

Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

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

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення.

Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи розгортання централізовано програмованих мереж на основі технології SDN.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем розгортання централізовано програмованих мереж на основі технології SDN.

– Досліджена система розгортання централізовано програмованих мереж на основі технології SDN.

– На основі отриманих результатів досліджень створена програмна реалізація системи розгортання централізовано програмованих мереж на основі технології SDN.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання розгортання централізовано програмованих мереж на основі технології SDN.

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. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи розгортання централізовано програмованих мереж на основі технології SDN. 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 CAST-128.

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

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

3. 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.

4. 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.

5. 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.

6. 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.

7. 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.

8. 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.

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

10. 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.

11. 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.

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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

33. 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.

34. 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.

35. 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.

36. 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.

37. 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.

38. 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.

39. 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).

40. 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

41. 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.

42. 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.

43. 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.

44. 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.

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

46. 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.

47. 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.

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

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

50. Смірнов О.А., Кавун С.В., Доренський О.П., Вялкова В.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 151 с.

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи розгортання централізовано програмованих мереж на основі технології SDN.

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 розгортання централізовано програмованих мереж на основі технології SDN.

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 розгортання централізовано програмованих мереж на основі технології SDN;

– 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.

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 – 90 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

_____ Коваленко А.С.

Програмне забезпечення системи розгортання централізовано програмованих мереж на основі технології SDN

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 36

Літера: РП

Кропивницький – 2024 року

Основна програма

Файл NetworkTimeRow.dpr основної програми

```
//Файл основної програми до якого підключаються відповідні //бібліотеки

program NetworkTimeRow;

uses
  Forms,
  IPHelper in ' IPHelper.pas' ,
  IPHLPAPI in ' IPHLPAPI.pas' ,
  MainFormUnit in ' MainFormUnit.pas'  {MainForm},
  TrafficUnit in ' TrafficUnit.pas' ;

{$R *.RES}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.Run;
end.
```

```
interface

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

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

//------перетворення визначених імен портів у сервісні імена------

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

  PTTcpConnStatus = ^TTcpConnStatus;
  TTcpConnStatus = record
    LocalIP      : string;
    LocalPort    : string;
    RemoteIP     : string;
    RemotePort   : string;
    Status       : string;
  end;


const
    // для самих розповсюджених сервісів...
  WellKnownPorts: array[1..29] of TWellKnownPort
  = (
    ( Prt: 0; Srv: ' LOOPBACK' ),
    ( Prt: 7; Srv: ' ECHO'  ),      {Пінгування    }
    ( Prt: 9; Srv: ' DISCRD' ),     { Відмова}
    ( Prt: 13; Srv: ' DAYTIM'  ),   {Час}
    ( Prt: 17; Srv: ' QOTD'  ),     {Вказник на час}
    ( Prt: 19; Srv: ' CHARGEN'  ),  {Генерація символів}
    ( Prt: 20; Srv: ' FTP '  ),     { File Transfer Protocol}
    ( Prt: 21; Srv: ' FTPC'  ),     { File Transfer Control Protocol}
    ( Prt: 23; Srv: ' TELNET'  ),   {TelNet}
    ( Prt: 25; Srv: ' SMTP'  ),     { Simple Mail Transfer Protocol}
    ( Prt: 37; Srv: ' TIME'  ),     { Часовий протокол }
    ( Prt: 43; Srv: ' WHOIS' ),     { WHO IS service  }
    ( 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'  ),   { Gopher       }
    ( Prt: 79; Srv: ' FING'  ),     { Finger       }
    ( Prt: 80; Srv: ' HTTP'  ),     { HTTP         }
    ( Prt: 88; Srv: ' KERB'  ),     { Kerberos     }
    ( Prt: 109; Srv: ' POP2'  ),    { Post Office Protocol Version 2 }
    ( Prt: 110; Srv: ' POP3'  ),    { Post Office Protocol Version 3 }
    ( Prt: 119; Srv: ' NNTP'  ),    { Network News Transfer Protocol }
    ( Prt: 123; Srv: ' NTP '  ),    { Network Time protocol          }
    ( Prt: 135; Srv: ' LOCSVC' ),   { Локальні сервіси             }
    ( Prt: 137; Srv: ' NBNAME'  ),  { NETBIOS Імя сервісу         }
    ( Prt: 138; Srv: ' NBDGRAM'  ), { NETBIOS Сервіс датаграм      }
    ( Prt: 139; Srv: ' NBSESS'  ),  { NETBIOS Сессійний сервіс        }
    ( Prt: 161; Srv: ' SNMP'  )     { Simple Netw. Management Protocol }
    );


//-----------перетворення ICMP кодів помилок у рядок-------------------------
```

```
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'
  );


//----------перетворення різних величин до рядку ----------------//

  ARPEntryType  : array[1..4] of string = ( ' Other' , ' Invalid' ,
    ' Dynamic' , ' Static'
    );

  TCPConnState  :      // стани підключення TCP
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

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

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

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


//---------------експортуємі данні-----------------------------------------
-

// дані перетворюються у Tstrings для представлення на дісплей
procedure Get_AdaptersInfo( List: TStrings );
procedure Get_NetworkParams( List: TStrings );
procedure Get_ARPTable( List: TStrings );
procedure Get_TCPTable( List: TStrings );
procedure Get_TCPStatistics( List: TStrings );
procedure Get_UDPTable( List: TStrings );
procedure Get_UDPStatistics( List: TStrings );
procedure Get_IPAddrTable( List: TStrings );
procedure Get_IPForwardTable( List: TStrings );
procedure Get_IPStatistics( List: TStrings );
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint;
  var RTT: longint; var HopCount: longint ): integer;
procedure Get_ICMPStats( ICMPIn, ICMPOut: TStrings );
procedure Get_IfTable( NameList, ItemList: TStrings );
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
```

```
procedure Get_RecentDestIPs( List: TStrings );

// додаємо функцію
procedure Get_OpenConnections( List: TList );



// 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;

//--------------Основні утіліти-----------------------------------------------

{ перетворюємо наступний токен у рядок, потім зчитуємо рядок та видаляємо його }
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;

//------------------------------------------------------------------------------
{ перетворюємо цифрові MAC-адреса у ww-xx-yy-zz строку }
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 у крапкову десяткову строку}
function IpAddr2Str( IPAddr: DWORD ): string;
var
  i             : integer;
begin
  Result := ' ' ;
  for i := 1 to 4 do
  begin
```

```
    Result := Result + Format( ' %3d.' , [IPAddr and $FF] );
    IPAddr := IPAddr shr 8;
  end;
  Delete( Result, Length( Result ), 1 );
end;

//---------------------------------------------------------------------------
{ перетворення крапкової десяткової IP-адреси у 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 номера порту у мережний байт типу DWORD }
function Port2Wrd( nwoPort: DWORD ): DWORD;
begin
  Result := Swap( WORD( nwoPort ) );
end;

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

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

//---------------------------------------------------------------------------
{ general,  fixed network parameters }
procedure Get_NetworkParams( List: TStrings );
var
  InfoSize      : Longint;
  ErrorCode     : DWORD;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  InfoSize := 0;
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  GetMem( pBuf, InfoSize );
  ErrorCode := GetNetworkParams( PTFixedInfo(pBuf), @InfoSize );
  if ErrorCode = ERROR_SUCCESS then
    with PTFixedinfo(pBuf)^ do
```

```
    begin
      List.Add( ' HOSTNAME          : '  + string( Ім'я хосту ) );
      List.Add( ' DOMAIN            : '  + string( ім'я домену ) );
      List.Add( ' SCOPE             : '  + string( ScopeID ) );
      List.Add( ' NETBIOS NODE TYPE : '  + NETBIOSTypes[NodeType] );
      List.Add( ' ROUTING ENABLED   :'  + IntToStr( Дозволена маршрутизація ) );
      List.Add( ' PROXY   ENABLED   :'  + IntToStr( Дозволений Proxy ) );
      List.Add( ' DNS     ENABLED   :'  + IntToHex( Дозволений DNS,8 ) );
    end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
  FreeMem(pBuf);
end;

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

//---------------------------------------------------------------------------
//процедура отримання трафіку централізовано програмованих мереж на основі
технології SDN та занесення у таблицю
procedure Get_IfTable( NameList, ItemList: TStrings );
var
  IfRow         : TMibIfRow;
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  if (not Assigned( NameList ))
  or (not Assigned( ItemList ))  then EXIT;
  NameList.Clear;
  ItemList.Clear;
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 1 to NumEntries do
      begin
        IfRow := PTMibIfRow( pBuf )^;
        with IfRow do
        begin
          SetLength( sDescr, dwDescrLen );
          move( bDescr, sDescr[1], Length( sDescr ) );
          sDescr := trim( sDescr );
          NameList.Add( sDescr );
          ItemList.Add( Format( ' %0.8x|%2d| %16s| %4d| %8d| %8d| %8d' ,
            [dwIndex, dwType,
```

```
            MacAddr2Str( TMacAddress( bPhysAddr ), dwPhysAddrLen )
              , dwMTU, dwSpeed,
              dwInOctets, dwOutOctets,
              dwOPerStatus] )
              );
        end;
        inc( pBuf, SizeOf( IfRow ) );
      end;
    end
    else  begin
      NameList.Add( ' без даних' );
      ItemList.Add( ' немає даних'  );
    end;
  end
  else begin
    NameList.Add( ' Oops' );
    ItemList.Add( SysErrorMessage( GetLastError ) );
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IfRow ) );
  FreeMem( pBuf );
end;


//---------------------------------------------------------------------------
//Занесення даних у таблицю MIB
procedure Get_IfTableMIB( var MIBIfArray: TMIBIfArray );
var
  i,
    Error,
    TableSize   : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
  sDescr,
    Temp        : string;
begin
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;
  GetMem( pBuf, TableSize );

   // отримуємо таблицю показчиків
  Error := GetIfTable( PTMibIfTable( pBuf ), @TableSize, false );
  if Error = NO_ERROR then
  begin
    NumEntries := PTMibIfTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( MIBIfArray, NumEntries );
      inc( pBuf, SizeOf( NumEntries ) );
      for i := 0 to pred(NumEntries) do
      begin
        MIBIfArray[i] := PTMibIfRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIfRow ) );
      end;
    end
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMIBIfRow ) );
  FreeMem( pBuf );
end;



//---------------------------------------------------------------------------
//Заносимо дінні про адреси у таблиці MIB
procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
```

```
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;



//* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
procedure Get_AdaptersInfo( List: TStrings );
var
  Error,
  BufLen      : DWORD;
  P           : Pointer;
  AdapterInfo   : PTIP_ADAPTER_INFO;
  Descr,
  LocalIP,
  GatewayIP,
  DHCPIP      : string;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  BufLen := SizeOf( AdapterInfo^ );
  New( AdapterInfo );
  Error := GetAdaptersInfo( AdapterInfo, @BufLen );
  P := AdapterInfo;
  if Error = NO_ERROR then
  begin
    while P <> nil do
      with TIP_ADAPTER_INFO(P^) do
      begin
        SetLength( Descr, SizeOf( Description ) );
        Descr := Trim( string( Description ) );
        //
        if IPAddressList.IpAddress[1] <> #0 then
          LocalIP := IPAddressList.IpAddress
        else
          LocalIP := NULL_IP;
        //
        if GateWayList.IPAddress[1] <> #0 then
```

```
          GateWayIP := GatewayList.IPAddress
        else
          GateWayIP := NULL_IP;
        //
        if DHCPServer.IPAddress[1] <> #0 then
          DHCPIP := DHCPServer.IPAddress
        else
          DHCPIP := NULL_IP;

        List.Add( Descr );
        List.Add( Format(
          ' %8.8x|%6s|%16s|%2d|%16s|%16s|%16s' ,
          [Index, AdaptTypes[aType],
          MacAddr2Str( TMacAddress( Address ), AddressLength ),
            DHCPEnabled, LocalIP, GatewayIP, DHCPIP] )
            );
        List.Add( '    ' );
        P := Next;  //  TIP_ADAPTER_INFO(P^).Next  points to next entry
      end // with
    end // while
  else
    List.Add( SysErrorMessage( Error ) );
  Dispose( AdapterInfo );
end;



//---------------------------------------------------------------------------
{ записуємо час завантаження трафіка мережі 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; // Розташування BAD_HOST_NAME,й.. т.і.
    HopCount := -1;
  end
  else
    Result := NO_ERROR;
end;



//---------------------------------------------------------------------------
{ ARP-таблиця записує відношення між IP та MAC-адресам.
}
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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetIPNetTable( PTMIBIpNetTable( pBuf ), @TableSize, false );
  //
  if ErrorCode = ERROR_NO_DATA then
  begin
    List.Add( '  ARP-cache empty.'  );
    EXIT;
  end;
  // заносимо у таблицю
```

```
  GetMem( pBuf, TableSize );
  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 ) ); // записуємо 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!
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPNetRow ) );
  FreeMem( pBuf );

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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  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 з
навантаженістю мережного трафіку централізовано програмованих мереж на основі
технології SDN
        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;


//---------------------------------------------------------------------------
// Опитуємо відкриті підключення до мережі
procedure Get_OpenConnections( List: TList );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
  CStat         : PTTcpConnStatus;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  // перший виклик: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам' ять. Викликаємо знову
  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, true );
  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 з
навантаженістю мережного трафіку централізовано програмованих мереж на основі
технології SDN
        with TCPRow do
          if dwState in [2,5] then   //
          begin
            New( CStat );
            CStat^.LocalIP   := IPAddr2Str( dwLocalAddr );
```

```
            CStat^.LocalPort := Port2Svc( Port2Wrd( dwLocalPort ));
            if dwRemoteAddr <> 0 then
            begin
              CStat^.RemoteIP     := IPAddr2Str( dwRemoteAddr );
              CStat^.RemotePort   := Port2Svc( Port2Wrd( dwRemotePort ));
            end
            else begin
              CStat^.RemoteIP   := ' ...' ;
              CStat^.RemotePort := ' ...' ;
            end;
            CStat^.Status       := TCPConnState[dwState];
            List.Add( CStat );
          end;
        inc( pBuf, SizeOf( TMIBTCPRow ) );
      end;
    end;
  end;
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibTCPRow ) );
  FreeMem( pBuf );
end;



//---------------------------------------------------------------------------
//Записуємо статистику трафіка по TCP
procedure Get_TCPStatistics( List: TStrings );
var
  TCPStats      : TMibTCPStats;
  ErrorCode     : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  ErrorCode := GetTCPStatistics( @TCPStats );
  if ErrorCode = NO_ERROR then
    with TCPStats do
    begin
      List.Add( ' Алгоритм повторної передачі  :'  + TCPToAlgo[dwRTOAlgorithm]
);
      List.Add( ' Мініміальний час          :'  + IntToStr( dwRTOMin ) + '  ms'
);
      List.Add( ' Максимальний час          :'  + IntToStr( dwRTOMax ) + '  ms'
);
      List.Add( ' Максимальна кількість підключень :'  + IntToStr(
dwRTOAlgorithm ) );
      List.Add( ' Активні підключення             :'  + IntToStr( dwActiveOpens
) );
      List.Add( ' Пасивні підключення            :'  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Помилка невдалого відкриття     :'  + IntToStr( dwAttemptFails
) );
      List.Add( ' Скидання встановленого підключення  :'  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.:'  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отриманий сегмент        :'  + IntToStr( dwInSegs ) );
      List.Add( ' Відправлений сегмент            :'  + IntToStr( dwOutSegs ) );
      List.Add( ' Переправлений сегмент   :'  + IntToStr( dwReTransSegs ) );
      List.Add( ' Помилка входження         :'  + IntToStr( dwInErrs ) );
      List.Add( ' Скидання виходу          :'  + IntToStr( dwOutRsts ) );
      List.Add( ' Сукупні зв'язки    :'  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );

end;

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

```
//Запис часу та завантаженості трафіку централізовано програмованих мереж на
основі технології SDN по UDP
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: необхідно отримати розмір таблиці у БД
  TableSize := 0;
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // резервуємо пам’ ять. Викликаємо знову
  GetMem( pBuf, TableSize );

  // заносимо у таблицю
  ErrorCode := GetUDPTable( PTMIBUDPTable( pBuf ), @TableSize, true );
  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 з
навантаженістю мережного трафіку централізовано програмованих мереж на основі
технології SDN
        with UDPRow do
          List.Add( Format( ‘ %15s : %-6s’ ,
            [IpAddr2Str( dwLocalAddr ),
            Port2Svc( Port2Wrd( dwLocalPort ) )
              ] ) );
        inc( pBuf, SizeOf( TMIBUDPRow ) );
      end;
    end
    else
      List.Add( ‘ без даних.’  );
  end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibUDPRow ) );
  FreeMem( pBuf );
end;

//-----------------------------------------------------------------------------
procedure Get_IPAddrTable( List: TStrings );

//Запис часу та завантаженості трафіку централізовано програмованих мереж на
основі технології SDN по IP

var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
```

```
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у 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( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;


(*
//---------------------------------------------------------------------------
//Запис IP адресів до MIB

procedure Get_IPAddrTableMIB( var IPAddrTable:TMibIPAddrArray  );
var
  IPAddrRow     : TMibIPAddrRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  TableSize := 0; ;
  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  GetMem( pBuf, TableSize );
  // заносимо у таблицю
  ErrorCode := GetIpAddrTable( PTMibIPAddrTable( pBuf ), @TableSize, true );
  if ErrorCode = NO_ERROR then
  begin
    NumEntries := PTMibIPAddrTable( pBuf )^.dwNumEntries;
    if NumEntries > 0 then
    begin
      SetLength( IPAddrTable, NumEntries);
```

```
      inc( pBuf, SizeOf( DWORD ) );
      for i := 1 to NumEntries do
      begin
        IPAddrTable[ i-1 ] := PTMIBIPAddrRow( pBuf )^;
        inc( pBuf, SizeOf( TMIBIPAddrRow ) );
      end;
    end;
  end;

  // відновлюємо показчик !
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( IPAddrRow ) );
  FreeMem( pBuf );
end;
*)

//----------------------------------------------------------------------------
{ отримуємо данні з таблиць маршрутизації }
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  TableSize := 0;

  // перший виклик: необхідно отримати розмір таблиці у БД
  ErrorCode := GetIpForwardTable( PTMibIPForwardTable( pBuf ), @TableSize, true
    );
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // заносимо у 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
          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
              ] ) );
        inc( pBuf, SizeOf( TMibIPForwardRow ) );
      end;
    end
    else
      List.Add( ' без даних.'  );
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
```

```
  dec( pBuf, SizeOf( DWORD ) + NumEntries * SizeOf( TMibIPForwardRow ) );
  FreeMem( pBuf );
end;

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

// Надання статистики по IP
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Пересилання дозволено       : '  + ' Так'  )
      else
        List.add( ' Пересилання дозволено       : '  + ' Ні'  );
      List.add( ' Вбудований TTL              : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Отримана датаграмма       : '  + inttostr( dwInReceives ) );
      List.add( ' Помилки заголовку     (У)  : '  + inttostr( dwInHdrErrors ) );
      List.add( ' Помилка адреси    (У)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Невідомий протокол  (У)  : '  + inttostr( dwInUnknownProtos )
);
      List.add( ' Відхілені датаграми       : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграми встановлені     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запит           : '  + inttostr( dwOutRequests ) );
      List.add( ' Маршрут відхилений        : '  + inttostr( dwRoutingDiscards )
);
      List.add( ' Немає маршруту          (Out): '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебирання часу      : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Перебирання запитів     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору      : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації    : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграму фрагментовано     : '  + inttostr( dwFRagCreates ) );
      List.add( ' Кількість інтерфейсів     : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут у таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по UDP

procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (У)     : '  + inttostr( dwInDatagrams ) );
```

```
      List.add( ' Датаграми (З)    : '  + inttostr( dwOutDatagrams ) );
      List.add( 'Немає портів          : '  + inttostr( dwNoPorts ) );
      List.add( ' Помилки     (У)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

//---------------------------------------------------------------------------
// Надання статистики по ICMP

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( ' Повідомлення прийнято    : '  + IntToStr( dwMsgs ) );
      ICMPIn.Add( ' Помилка                : '  + IntToStr( dwErrors ) );
      ICMPIn.Add( ' Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPIn.Add( ' Час перевищений       : '  + IntToStr( dwTimeEcxcds ) );
      ICMPIn.Add( ' Проблеми з параметрами    : '  + IntToStr( dwParmProbs )
);
      ICMPIn.Add( ' Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPIn.Add( ' Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPIn.Add( ' Ехо запит          : '  + IntToStr( dwEchos ) );
      ICMPIn.Add( ' Ехо повтор           : '  + IntToStr( dwEchoReps ) );
      ICMPIn.Add( ' Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPIn.Add( ' Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );

      ICMPIn.Add( ' Запит адреси маски : '  + IntToStr( dwAddrMasks ) );
      ICMPIn.Add( ' Повтор адреси маски   : '  + IntToStr( dwAddrReps ) );
    end;
     //
    with ICMPStats^.OutStats do
    begin
      ICMPOut.Add( 'Повідомлення відправлено: '  + IntToStr( dwMsgs ) );
      ICMPOut.Add( 'Помилка                : '  + IntToStr( dwErrors ) );
      ICMPOut.Add( 'Розташування недосягнене    : '  + IntToStr( dwDestUnreachs
) );
      ICMPOut.Add( 'Час перевищений        : '  + IntToStr( dwTimeEcxcds ) );
      ICMPOut.Add( 'Проблеми з параметрами      : '  + IntToStr( dwParmProbs )
);
      ICMPOut.Add( 'Джерело відключене        : '  + IntToStr( dwSrcQuenchs ) );
      ICMPOut.Add( 'Перепризначення          : '  + IntToStr( dwRedirects ) );
      ICMPOut.Add( 'Ехо запит         : '  + IntToStr( dwEchos ) );
      ICMPOut.Add( 'Ехо повтор          : '  + IntToStr( dwEchoReps ) );
      ICMPOut.Add( 'Запит мітки часу    : '  + IntToStr( dwTimeStamps ) );
      ICMPOut.Add( 'Повтор мітки часу    : '  + IntToStr( dwTimeStampReps ) );
      ICMPOut.Add( 'Запит адреси маски: '  + IntToStr( dwAddrMasks ) );
      ICMPOut.Add( 'Повтор адреси маски   : '  + 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.
```

unit IPHLPAPI – бібліотека API функцій для роботи з IP мережею

```
interface
uses
  Windows, winsock;
const

  VERSION       = '1.3';


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

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

  NETBIOSTypes  : array[0..8] of string[20] =
    ( 'UNKNOWN', 'BROADCAST', 'PEER_TO_PEER', '', 'MIXED', '', '', '', 'HYBRID'
      );

 // Типи адаптерів
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;
 //
  AdaptTypes    : array[0..6] of string[10] =
    ( 'інший', 'ethernet', 'tokenring', 'FDDI', 'PPP', 'loopback', 'SLIP' );


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

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

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

//------Структура IP-адрес -------------------------------------------------

  PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
  TIP_ADDRESS_STRING = array[0..15] of char; //  IP в xxx.xxx.xxx.xxx рядок
  //
  PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
  TIP_ADDR_STRING = packed record //
    Next: PTIP_ADDR_STRING;
    IpAddress: TIP_ADDRESS_STRING;
    IpMask: TIP_ADDRESS_STRING;
    Context: DWORD;
  end;

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

```
  PTFixedInfo = ^TFixedInfo;
  TFixedInfo = packed record
    HostName: array[0..MAX_HOSTNAME_LEN + 4] of char;
    DomainName: array[0..MAX_DOMAIN_NAME_LEN + 4] of char;
    CurrentDNSServer: PTIP_ADDR_STRING;
    DNSServerList: TIP_ADDR_STRING;
    NodeType: UINT;
    ScopeID: array[0..MAX_SCOPE_ID_LEN + 4] of char;
    EnableRouting: UINT;
    EnableProxy: UINT;
    EnableDNS: UINT;
  end;

//----------INTERFACE структура-----------------------------------------------


  PTMibIfRow = ^TMibIfRow;
  TMibIfRow = packed record
    wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
    dwIndex: DWORD;
    dwType: DWORD;
    dwMTU: DWORD;
    dwSpeed: DWORD;
    dwPhysAddrLen: DWORD;
    bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
    dwAdminStatus: DWORD;
    dwOperStatus: DWORD;
    dwLastChange: DWORD;
    dwInOctets: DWORD;
    dwInUcastPkts: DWORD;
    dwInNUCastPkts: DWORD;
    dwInDiscards: DWORD;
    dwInErrors: DWORD;
    dwInUnknownProtos: DWORD;
    dwOutOctets: DWORD;
    dwOutUCastPkts: DWORD;
    dwOutNUCastPkts: DWORD;
    dwOutDiscards: DWORD;
    dwOutErrors: DWORD;
    dwOutQLen: DWORD;
    dwDescrLen: DWORD;
    bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
  end;

  TMIBIfArray = array of TMIBIFRow;


 //
  PTMibIfTable = ^TMIBIfTable;
  TMibIfTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIfRow;
  end;

//------ADAPTER INFO структура--------------------------------------------------

  TTIME_T = array[1..325] of byte; //

  PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
  TIP_ADAPTER_INFO = packed record
    Next: PTIP_ADAPTER_INFO;
    ComboIndex: DWORD;
    AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;
    Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
    AddressLength: UINT;
    Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;
    Index: DWORD;
    aType: UINT;
```

```
    DHCPEnabled: UINT;
    CurrentIPAddress: PTIP_ADDR_STRING;
    IPAddressList: TIP_ADDR_STRING;
    GatewayList: TIP_ADDR_STRING;
    DHCPServer: TIP_ADDR_STRING;
    HaveWINS: BOOL;
    PrimaryWINSServer: TIP_ADDR_STRING;
    SecondaryWINSServer: TIP_ADDR_STRING;
    LeaseObtained: TTIME_T; //??
    LeaseExpires: TTIME_T; //??
  end;

//----------------TCP структура-----------------------------------------------

  PTMibTCPRow = ^TMibTCPRow;
  TMibTCPRow = packed record
    dwState: DWORD;
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
    dwRemoteAddr: DWORD;
    dwRemotePort: DWORD;
  end;
  //
  PTMibTCPTable = ^TMibTCPTable;
  TMibTCPTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..0] of TMibTCPRow;
  end;
  //
  PTMibTCPStats = ^TMibTCPStats;
  TMibTCPStats = packed record
    dwRTOAlgorithm: DWORD;
    dwRTOMin: DWORD;
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//---------UDP структура-----------------------------------------------------

  PTMibUDPRow = ^TMibUDPRow;
  TMibUDPRow = packed record
    dwLocalAddr: DWORD;
    dwLocalPort: DWORD;
  end;
 //
  PTMibUDPTable = ^TMIBUDPTable;
  TMIBUDPTable = packed record
    dwNumEntries: DWORD;
    UDPTable: array[0..ANY_SIZE - 1] of TMibUDPRow;
  end;
 //
  PTMibUdpStats = ^TMIBUdpStats;
  TMIBUdpStats = packed record
    dwInDatagrams: DWORD;
    dwNoPorts: DWORD;
    dwInErrors: DWORD;
    dwOutDatagrams: DWORD;
    dwNumAddrs: DWORD;
```

```
  end;

//-----------IP структура-------------------------------------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
      Unused2: WORD;
  end;

  TMibIPAddrArray = array of TMIBIPAddrRow;


  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPAddrRow;
  end;

  //
```

```
  PTMibIPForwardRow = ^TMibIPForwardRow;
  TMibIPForwardRow = packed record
    dwForwardDest: DWORD;
    dwForwardMask: DWORD;
    dwForwardPolicy: DWORD;
    dwForwardNextHop: DWORD;
    dwForwardIFIndex: DWORD;
    dwForwardType: DWORD;
    dwForwardProto: DWORD;
    dwForwardAge: DWORD;
    dwForwardNextHopAS: DWORD;
    dwForwardMetric1: DWORD;
    dwForwardMetric2: DWORD;
    dwForwardMetric3: DWORD;
    dwForwardMetric4: DWORD;
    dwForwardMetric5: DWORD;
  end;
  //
  PTMibIPForwardTable = ^TMibIPForwardTable;
  TMibIPForwardTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE - 1] of TMibIPForwardRow;
  end;

//--------ICMP-структура-------------------------------------------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

  PTMibICMPInfo = ^TMibICMPInfo;
  TMibICMPInfo = packed record
    InStats: TMibICMPStats;
    OutStats: TMibICMPStats;
  end;




//------------------імпортовано з IPHLPAPI.DLL---------------------------------
-

function GetAdaptersInfo( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetNetworkParams( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpTable( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetTcpStatistics( pStats: PTMibTCPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';
```

```
function GetUdpTable( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetUdpStatistics( pStats: PTMibUdpStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpStatistics( pStats: PTMibIPStats ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpNetTable( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpAddrTable( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdcall; external 'IPHLPAPI.DLL';

function GetIpForwardTable( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG;
  bOrder: BOOL ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

function GetIcmpStatistics( pStats: PTMibICMPInfo ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetRTTAndHopCount( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL;
stdCall; external 'IPHLPAPI.DLL';
function GetIfTable( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD;
stdCall; external 'IPHLPAPI.DLL';
function GetIfEntry( pIfRow: PTMibIfRow ): DWORD;
stdCall; external 'IPHLPAPI.DLL';

implementation
end.
```

unit TrafficUnit – визначення параметрів трафіку централізовано програмованих мереж на основі технології SDN;

```
interface

uses SysUtils, Windows, IPHelper, IPHLPAPI;


type
  TTraffic = Class;

  TNewInstanceEvent = procedure(Sender :TTraffic) of object;
  TFreezeEvent = procedure(Sender :TTraffic) of object;

  TTraffic = Class
  private
    FIP: string;
    FMac: string;
    FInPerSec: Dword;
    FInTotal: Dword;
    FPeakInPerSec: Dword;
    FInterfaceIndex: DWord;
    FActiveCountIn: Dword;
    FSecondsActive: Cardinal;
    FPrevCountIn: DWord;
    FDescription: string;
    FOutTotal: Dword;
    FPeakOutPerSec: Dword;
    FOutPerSec: Dword;
    FPrevCountOut: DWord;
    FActiveCountOut: Dword;
    FAverageInPerSec: Dword;
    FAverageOutPerSec: Dword;
    FStartedAt: TDateTime;
    FRunning: boolean;
    FOnFreeze: TFreezeEvent;
    FOnUnFreeze: TFreezeEvent;
    FConnected: boolean;
    FFound: boolean;
    FSpeed: DWord;

    function GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
  public
    property Found : boolean read FFound write FFound;
    property Connected : boolean read FConnected;
    property Running : boolean read FRunning;
    property InterfaceIndex : DWord read FInterfaceIndex;
    property IP : string read FIP;
    property Mac : string read FMac;
    property Description : string read FDescription;
    property StartedAt : TDateTime read FStartedAt;
    property SecondsActive : Cardinal read FSecondsActive;
    property Speed : DWord read FSpeed;
    property ActiveCountIn : Dword read FActiveCountIn; { }
    property PrevCountIn : DWord read FPrevCountIn; { }
    property InPerSec : Dword read FInPerSec; { байт підраховано в останній
період }
    property AverageInPerSec : Dword read FAverageInPerSec; { У середньому }
    property InTotal : Dword read FInTotal; { загальна кількість байтів }
    property PeakInPerSec : Dword read FPeakInPerSec; { максимальне число
байтів}

    property ActiveCountOut : Dword read FActiveCountOut; { підраховує число
байтів при передачі }
    property PrevCountOut : DWord read FPrevCountOut; { попереднє підрахування
байтів }
    property OutPerSec : Dword read FOutPerSec; { Кількість байтів у останій
період }
    property AverageOutPerSec : Dword read FAverageOutPerSec; { }
```

```
    property OutTotal : Dword read FOutTotal; { Загальна кількість байтів
передано }
    property PeakOutPerSec : Dword read FPeakOutPerSec; { Максимальна кількість
байтів передано }

    procedure NewCycle(const InOctets, OutOctets, TrafficSpeed : Dword);
    procedure Reset;
    procedure Freeze;
    procedure UnFreeze;
    procedure MarkDisconnected;
    function GetStatus : string;
    function FriendlyRunningTime:string;
    constructor Create(const AMibIfRow : TMibIfRow; OnNewInstance :
TNewInstanceEvent);
  published
    property OnFreeze :TFreezeEvent read FOnFreeze write FOnFreeze;
    property OnUnFreeze :TFreezeEvent read FOnUnFreeze write FOnUnFreeze;
  end;

  function BytesToFriendlyString(Value : DWord) : string;
  function BitsToFriendlyString(Value : DWord) : string;

implementation

function BytesToFriendlyString(Value : DWord) : string;
const
 OneKB=1024;
 OneMB=OneKB*1024;
 OneGB=OneMB*1024;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 B',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 KB', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 MB', Value/OneMB)
end;

function BitsToFriendlyString(Value : DWord) : string;
const
 OneKB=1000;
 OneMB=OneKB*1000;
 OneGB=OneMB*1000;
begin
 if Value<OneKB
 then Result:=FormatFloat('#,##0.00 bps',Value)
 else
  if Value<OneMB
  then Result:=FormatFloat('#,##0.00 Kbps', Value/OneKB)
  else
   if Value<OneGB
   then Result:=FormatFloat('#,##0.00 Mbps', Value/OneMB)
end;

constructor TTraffic.Create(const AMibIfRow: TMibIfRow; OnNewInstance :
TNewInstanceEvent);
var
 Descr: string;
begin
 inherited Create;

 FRunning:=true;
 FConnected:=true;

 self.FInterfaceIndex:=AMibIfRow.dwIndex;
 self.FIP:=GetIPFromIFIndex(self.InterfaceIndex);
```

```
 self.FMac:=MacAddr2Str(TMacAddress(AMibIfRow.bPhysAddr),
AMibIfRow.dwPhysAddrLen);

 SetLength(Descr, Pred(AMibIfRow.dwDescrLen));
 Move(AMibIfRow.bDescr, Descr[1], pred(AMibIfRow.dwDescrLen));
 self.FDescription:=Trim(Descr);

 self.FPrevCountIn:=AMibIfRow.dwInOctets;
 self.FPrevCountOut:=AMibIfRow.dwOutOctets;

 self.FStartedAt:=Now;
 self.FSpeed:=AMibIfRow.dwSpeed;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;

  if Assigned(OnNewInstance)
 then OnNewInstance(self);
end;

procedure TTraffic.NewCycle(const InOctets, OutOctets, TrafficSpeed: Dword);
begin
 inc(self.FSecondsActive);
 if not Running
 then Exit;
 FSpeed:=TrafficSpeed;
 // прийнято
 self.FInPerSec:=InOctets-self.PrevCountIn;
 Inc(self.FInTotal, self.InPerSec);
 if InPerSec>0
 then Inc(FActiveCountIn);
 if InPerSec>PeakInPerSec
 then FPeakInPerSec:=InPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageInPerSec:=InTotal div ActiveCountIn
 except
  self.FAverageInPerSec:=0;
 end;
 FPrevCountIn:=InOctets;
 // передано
 self.FOutPerSec:=OutOctets-self.PrevCountOut;
 Inc(self.FOutTotal,self.OutPerSec);
 if OutPerSec>0
 then Inc(FActiveCountOut);
 if OutPerSec>PeakOutPerSec
 then FPeakOutPerSec:=OutPerSec;
 try
  if ActiveCountIn<>0
  then self.FAverageOutPerSec:=OutTotal div ActiveCountOut
 except
  self.FAverageOutPerSec:=0;
 end;
 FPrevCountOut:=OutOctets;
end;

function TTraffic.GetIPFromIFIndex(InterfaceIndex: Cardinal): string;
var
 i: integer;
 IPArr: TMIBIPAddrArray;
begin
 Result:='Не знайдено!'; //...
 Get_IPAddrTableMIB(IpArr); // беремо таблицю IP-адрес
```

```
 if Length(IPArr)>0
 then
  for i:=low(IPArr) to High(IPArr) do  // проглядаємо індекси у таблиці...
   if IPArr[i].dwIndex=InterfaceIndex
   then
    begin
     Result:=IPAddr2Str(IParr[i].dwAddr);
     Break;
    end;
end;

procedure TTraffic.Reset;
begin
 self.FPrevCountIn:=InPerSec;
 self.FPrevCountOut:=OutPerSec;

 self.FStartedAt:=Now;
 FSecondsActive:=0;

 FActiveCountIn:=0;
 FActiveCountOut:=0;
 FInTotal:=0;
 FOutTotal:=0;
 FInPerSec:=0;
 FOutPerSec:=0;
 FPeakInPerSec:=0;
 FPeakOutPerSec:=0;
end;

procedure TTraffic.Freeze;
begin
 FRunning:=false;
 if Assigned(FOnFreeze)
 then OnFreeze(Self);
end;

procedure TTraffic.UnFreeze;
begin
 FRunning:=true;
 if Assigned(FOnUnFreeze)
 then OnUnFreeze(Self);
end;

procedure TTraffic.MarkDisconnected;
begin
 self.FConnected:=false;
 self.FRunning:=false;
end;
function TTraffic.GetStatus: string;
begin
 if self.Connected
 then Result:='Підключено'
 else Result:='Не підключено';
 if self.Running
 then Result:=Result+', Запущено'
 else Result:=Result+', Не запущено';
end;
function TTraffic.FriendlyRunningTime: string;
var
 H,M,S: string;
 ZH,ZM,ZS: integer;
begin
 ZH:=SecondsActive div 3600;
 ZM:=Integer(SecondsActive) div (60-ZH*60);
 ZS:=Integer(SecondsActive)-(ZH*3600+ZM*60);
 H:=Format('%.2d',[ZH]);
 M:=Format('%.2d',[ZM]);
 S:=Format('%.2d',[ZS]);
 Result:=H+':'+M+':'+S;
```

```
end;
end.
```

unit MainFormUnit – Запуск основної форми програми;

```
interface

uses
  Windows, Graphics, ExtCtrls, Controls, StdCtrls, Buttons, Tabs,
  ComCtrls, Classes, SysUtils, Forms, dialogs,
  TrafficUnit, IPHelper, IPHLPAPI, ShellAPI;

type
  TMainForm = class(TForm)
    pnlMain: TPanel;
    pnlBottom: TPanel;
    pc: TPageControl;
    tsAbout: TTabSheet;
    tsTraffic: TTabSheet;
    ExitButton: TButton;
    TrafficTabs: TTabSet;
    GroupBox: TGroupBox;
    ledAdapterDescription: TLabeledEdit;
    UnFreezeButton: TBitBtn;
    FreezeButton: TBitBtn;
    ClearCountersButton: TBitBtn;
    ledMACAddress: TLabeledEdit;
    gbIN: TGroupBox;
    ledOctInSec: TLabeledEdit;
    ledAvgINSec: TLabeledEdit;
    ledPeakINSec: TLabeledEdit;
    ledTotalIN: TLabeledEdit;
    gbOUT: TGroupBox;
    ledOctOUTSec: TLabeledEdit;
    ledAvgOUTSec: TLabeledEdit;
    ledPeakOUTSec: TLabeledEdit;
    ledTotalOUT: TLabeledEdit;
    Timer: TTimer;
    gbTime: TGroupBox;
    ledStartedAt: TLabeledEdit;
    ledActiveFor: TLabeledEdit;
    RemoveInactiveButton: TBitBtn;
    StatusText: TStaticText;
    cbOnTop: TCheckBox;
    Panel3: TPanel;
    ProductName: TLabel;
    lblURL: TLabel;
    Label3: TLabel;
    ProgramIcon: TImage;
    StaticText1: TStaticText;
    ledSpeed: TLabeledEdit;
    procedure TimerTimer(Sender: TObject);
    procedure ClearCountersButtonClick(Sender: TObject);
    procedure cbOnTopClick(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
    procedure TrafficTabsChange(Sender: TObject; NewTab: Integer;
      var AllowChange: Boolean);
    procedure ExitButtonClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FreezeButtonClick(Sender: TObject);
    procedure UnFreezeButtonClick(Sender: TObject);
    procedure RemoveInactiveButtonClick(Sender: TObject);
    procedure lblURLClick(Sender: TObject);
    procedure StaticText1Click(Sender: TObject);
    procedure pcChange(Sender: TObject);
    procedure ledAdapterDescriptionChange(Sender: TObject);
  private
    procedure HandleNewAdapter(ATraffic : TTraffic);
    procedure HandleFreeze(ATraffic : TTraffic);
    procedure HandleUnFreeze(ATraffic : TTraffic);
    function LocateTraffic(AdapterIndex : DWord) : TTraffic;
    procedure ProcessMIBData;
```

```
    procedure ClearDisplay;
    procedure RefreshDisplay;
  public
    { }
  end;

var
  MainForm: TMainForm;
  ActiveTraffic : TTraffic;

implementation
{$R *.dfm}

procedure TMainForm.ClearDisplay;
var
 j:integer;
begin
 TrafficTabs.Tabs.Clear;
 StatusText.Caption:='';
 for j:=0 to GroupBox.ControlCount-1 do
  begin
   if GroupBox.Controls[j] is TCustomEdit
   then TCustomEdit(GroupBox.Controls[j]).Text:='';
  end;
end;

procedure TMainForm.TimerTimer(Sender: TObject);
begin
 Timer.Enabled:=false;
 ProcessMIBData;
 Timer.Enabled:=true;
end;

procedure TMainForm.ClearCountersButtonClick(Sender: TObject);
begin
 ActiveTraffic.Reset;
 RefreshDisplay;
end;

procedure TMainForm.cbOnTopClick(Sender: TObject);
begin
 if cbOnTop.Checked=true
 then FormStyle:=fsSTAYONTOP
 else FormStyle:=fsNORMAL;
end;
procedure TMainForm.FormDestroy(Sender: TObject);
var
 i: integer;
begin
 Timer.OnTimer:=nil;
 ActiveTraffic:=nil;
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  TrafficTabs.Tabs.Objects[i].Free;
end;

procedure TMainForm.TrafficTabsChange(Sender: TObject; NewTab: Integer; var
AllowChange: Boolean);
begin
 if NewTab=-1
 then ActiveTraffic:=nil
 else ActiveTraffic:=TTraffic(TrafficTabs.Tabs.Objects[NewTab]);
 RefreshDisplay;
end;

procedure TMainForm.ExitButtonClick(Sender: TObject);
begin
 Close;
end;
```

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  Timer.Interval:=1000; // усі розрахунки за 1 сек.
 //
 ClearDisplay;
 ActiveTraffic:=nil;
 pcChange(Sender);
 Timer.Enabled:=True;
end;

procedure TMainForm.RefreshDisplay;
begin
 if not Assigned(ActiveTraffic)
 then
  begin
   ClearDisplay;
   Exit;
  end;
 with ActiveTraffic do
  begin
   FreezeButton.Visible:=Connected;
   UnFreezeButton.Visible:=Connected;
   ClearCountersButton.Visible:=Connected;
   RemoveInactiveButton.Visible:=not Connected;

   FreezeButton.Enabled:=Running;
   UnFreezeButton.Enabled:=not Running;

   ledAdapterDescription.Text:=Description;
   ledMACAddress.Text:=MAC;

   ledSpeed.Text:=BitsToFriendlyString(Speed);

   ledOctInSec.Text:=BytesToFriendlyString(InPerSec);
   ledPeakInSec.Text:=BytesToFriendlyString(PeakInPerSec);
   ledAvgINSec.Text:=BytesToFriendlyString(AverageInPerSec);
   ledTotalIN.Text:=BytesToFriendlyString(InTotal);

   ledOctOUTSec.Text:=BytesToFriendlyString(OutPerSec);
   ledPeakOUTSec.Text:=BytesToFriendlyString(PeakOutPerSec);
   ledAvgOUTSec.Text:=BytesToFriendlyString(AverageOutPerSec);
   ledTotalOUT.Text:=BytesToFriendlyString(OutTotal);

   self.ledStartedAt.Text:=DateTimeToStr(StartedAt);
   self.ledActiveFor.Text:=FriendlyRunningTime;

   StatusText.Caption:=GetStatus;
  end;
end;

procedure TMainForm.ProcessMIBData;
var
 MibArr: IpHlpAPI.TMIBIfArray;
 i: integer;
 ATraffic: TTraffic;
begin
 Get_IfTableMIB(MibArr);  // Беремо поточні MIB дані
 //Мітку не знайдено, або не підключено
 for i:= 0 to -1 + TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[i]);
   if ATraffic.Connected
   then ATraffic.Found:=false;
  end;
  //процесс
 if Length(MibArr)>0
 then
  begin
   for i:=Low(MIBArr) to High(MIBArr) do
```

```
    begin
     ATraffic:=LocateTraffic(MIBArr[i].dwIndex);
     if Assigned(ATraffic)
     then
      begin
       //заново підключаємось
       ATraffic.NewCycle(MIBArr[i].dwInOctets, MIBArr[i].dwOutOctets,
MIBArr[i].dwSpeed);
      end
     else
      begin
       //новий запис у таблицю!
       ATraffic:=TTraffic.Create(MIBArr[i], HandleNewAdapter);
       ATraffic.Found:=true;
       ATraffic.OnFreeze:=HandleFreeze;
       ATraffic.OnUnFreeze:=HandleUnFreeze;
      end;
    end;
  end;
 //Мітка не знайдена
 for i:=0 to -1+TrafficTabs.Tabs.Count do
  if not TTraffic(TrafficTabs.Tabs.Objects[i]).Found
  then TTraffic(TrafficTabs.Tabs.Objects[i]).MarkDisconnected;
 RefreshDisplay;
end;

function TMainForm.LocateTraffic(AdapterIndex : DWord): TTraffic;
var
 j: cardinal;
 ATraffic: TTraffic;
begin
 Result:=nil;
 if TrafficTabs.Tabs.Count=0
 then Exit;

 for j:= 0 to -1+TrafficTabs.Tabs.Count do
  begin
   ATraffic:=TTraffic(TrafficTabs.Tabs.Objects[j]);
   if ATraffic.InterfaceIndex=AdapterIndex
   then
    begin
     Result:=ATraffic;
     Result.Found:=true;
     Break;
    end;
  end;
end;

procedure TMainForm.HandleNewAdapter(ATraffic: TTraffic);
begin
 //додаємо адаптер
 TrafficTabs.Tabs.AddObject(ATraffic.IP, ATraffic);
 // вибираємо
 TrafficTabs.TabIndex:=-1+TrafficTabs.Tabs.Count;
end;

procedure TMainForm.FreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.Freeze;
end;

procedure TMainForm.UnFreezeButtonClick(Sender: TObject);
begin
 ActiveTraffic.UnFreeze;
end;

procedure TMainForm.HandleFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
```

```
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.HandleUnFreeze(ATraffic: TTraffic);
begin
 self.FreezeButton.Enabled:=ATraffic.Running;
 self.UnFreezeButton.Enabled:=not ATraffic.Running;
end;

procedure TMainForm.RemoveInactiveButtonClick(Sender: TObject);
begin
 if not ActiveTraffic.Connected
 then //точна перевірка
  begin
   ActiveTraffic.Free;
   ActiveTraffic:=nil;
   TrafficTabs.Tabs.Delete(TrafficTabs.TabIndex);
   TrafficTabs.SelectNext(False);
  end;
 RefreshDisplay;
end;

procedure TMainForm.lblURLClick(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.StaticText1Click(Sender: TObject);
begin
 ShellExecute(Handle, 'open','',nil,nil,SW_SHOWNORMAL);
end;

procedure TMainForm.pcChange(Sender: TObject);
begin
 pnlBottom.Visible:=pc.ActivePage=tsTraffic;
end;

procedure TMainForm.ledAdapterDescriptionChange(Sender: TObject);
begin
  ledAdapterDescription.Hint:=ledAdapterDescription.Text;

ledAdapterDescription.ShowHint:=Canvas.TextWidth(ledAdapterDescription.Text)>led
AdapterDescription.ClientWidth;
end;

end.
```

unit about – підпрограмма про розробників та місце розроблення програми;

```
interface

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

type
  TForm2 = class(TForm)
    Image1: TImage;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Button1: TButton;
    Label7: TLabel;
    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.
```