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 макровіртуалізації SDDC”

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

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 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

Цеську Сергію Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи макровіртуалізації SDDC*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(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 макровіртуалізації SDDC*

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
_____ Буравченко К.О.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Цесько С.Р.
(прізвище та ініціали)

АНОТАЦІЯ

Цесько С.Р. Програмне забезпечення системи макровіртуалізації SDDC. 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 макровіртуалізації SDDC.

Метою розробки є програмне забезпечення системи макровіртуалізації SDDC.

Результат роботи – програмна реалізація системи макровіртуалізації SDDC.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі Delphi 10.

Ключові слова: комп'ютерна інженерія, SDDC

ABSTRACT

Tsesko S.R. SDDC macro virtualization system software. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the SDDC macro virtualization system.

The goal of the development is the SDDC macro virtualization system software.

The result of the work is the software implementation of the SDDC macro virtualization system.

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

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

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

The program was developed in the Delphi 10 environment.

Keywords: computer engineering, SDDC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІПМ	–	імітаційна програмна модель
ЛОМ	–	локальні обчислювальні мережі
ПрЗд	–	пропускна здатність
ПМ	–	програмна модель
ТПП	–	таймер повторної передачі
ARTCP	–	удосконалений транспортний протокол
BER	–	імовірність бітової помилки
CR	–	запит на з'єднання
CBR	–	протокол передачі даних без підтверджень
FIFO	–	принцип first input fipst output
IP	–	адресний протокол
OSI	–	еталона модель мережної архітектури
RTT	–	час затрачуваємий підтвердженням
TCP	–	протокол транспортного рівня
TCP/IP	–	протокол передачі даних
TPDU	–	повідомлення транспортного протоколу
UDP	–	протокол користувальницьких датаграмм
WAN	–	територіальні мережі

ВСТУП

Актуальність теми. Програмно обумовлених ЦОД у власному значенні цього терміна сьогодні немає ні в Україні, ні у світі. Поки ще не вироблені єдині стандарти й не створені програмні продукти, що охоплюють повний спектр завдань SDDC. Однак вендори, що мають у своєму портфелі рішення DCIM або ПЗ керування віртуальними машинами (ВМ), активно працюють у даному напрямку. Дуже імовірно, що в доступному для огляду майбутньому ця тенденція, що зароджується, стане цілком відчутною реальністю. Зробити ЦОД програмно обумовленим (Software-Defined Data Center, SDDC) можна за допомогою ряду IT-технологій і інструментів. У першу чергу при побудові SDDC варто задуматися про технологію макровіртуалізації. Вона має на увазі застосування тандема із двох систем керування – віртуальними машинами (ВМ) і інфраструктурою ЦОД (Data Center Infrastructure Management, DCIM) – з метою зниження експлуатаційних витрат. DCIM збирає й агрегує інформацію про стан інженерної інфраструктури центра обробки даних, наявності вільних площ, а також місця в стійках. Система допомагає одержувати максимально повну картину того, що відбувається в ЦОД компанії-провайдеру «тут і зараз». У сукупності з інформацією із системи керування ВМ це дозволяє виявити проблемні крапки в ЦОД (локальне перевищення граничних значень температури, недостача електроживлення й ін.), на підставі чого може бути затверджене рішення про переміщення обчислювального навантаження в ту або іншу фізичну зону центра обробки даних (або в інший ЦОД). Завдяки переносу високонавантажених віртуальних машин ближче до тих елементам інженерної інфраструктури, які здатні забезпечити їхню належну підтримку, «розжарення серверних страстей» знижується.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи макровіртуалізації SDDC.

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

– Огляд існуючих систем макровіртуалізації SDDC.

– Дослідження системи макровіртуалізації SDDC.

– Програмна реалізація системи макровіртуалізації SDDC.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі макровіртуалізації SDDC.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи макровіртуалізації SDDC, 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

Фізичні центри обробки даних провайдеру, об'єднані технологією макровіртуалізації, трансформуються в єдину «екосистему», що володіє, крім іншого, підвищеною надійністю. При невисокому рівні надійності інженерної інфраструктури кожного окремого ЦОД разом вони здатні забезпечити повне взаємне резервування.

Природно, однією макровіртуалізацією у випадку SDDC не обійтися. У концепцію програмно обумовленого ЦОД органічно вписується стійка тенденція спрощення фізичної інфраструктури й переносу складності у віртуальне програмне середовище, що дозволяє забезпечити комплексне керування інфраструктурою.

1.2 Область застосування

Областю застосування розроблювальної системи є ЦОД. Дата центр – усе ще рідке поняття для українських компаній. І якщо серед IT-шників його роль питань не викликає, то починаючи від менеджерів до власників компаній інших областей виникає маса питань. Питання дуже різноманітні – від того, що таке Дата-центр, як улаштований процес його роботи й властиво навіщо він потрібний.

Так що ж це таке Дата-центр? Звичайно можна звернутися до Википедии й прочитати, що: Дата-центр (від англ. data center), або центр (зберігання й) обробки даних (ЦОД/ЦЗОД) – це спеціалізований будинок для розміщення (хостингу) серверного й мережного встаткування... Це ясно для згаданих вище все тих же IT-шників. Дата-центр є високотехнологічною охоронюваною

площадкою, де розміщаються сервера різних компаній. Простіше говорячи, дата-центр – це своєрідний «будинок серверів». У першу чергу варто згадати, що сама послуга буде корисна компаніям, чия діяльність прямо залежить від безперебійного забезпечення швидкої й ефективної обробки більших, а іноді й колосальних потоків інформації. У століття інформаційних технологій, часто з метою викрадення, інформація піддається всіляким атакам. Саме впровадження спеціальних катастрофостійких рішень, організація резервного копіювання даних у дата-центах максимально убезпечить дані від ризиків втрати. Прийнято думати, що послуги Дата-центра затребувані тільки великими компаніями, але насправді спостерігається тенденція використання послуг і компаніями, що розвиваються, особливо якщо мова йде про стартапах з іноземними інвестиціями.

Фактично призначення комерційного Дата центра полягає в наданні клієнтам послуг, пов'язаних із забезпеченням надійності й відказостійкости зберігання й обробки інформації (текстові, графічні, цифрові й інші дані), для забезпечення працездатності великих інтернет-порталів, для об'ємних обчислень.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи макровіртуалізації SDDC, 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

Послуги Cisco Nexus

Ці послуги призначені для швидкої й ефективної інтеграції нових рішень і технологій для центрів обробки даних. Якщо організація має потребу в консолідації або створенні нового ЦОД, забезпеченні безперервності бізнес-процесів або поліпшенні відповідності нормативним вимогам, послуги Cisco Nexus допоможуть прискорити перехід на масштабовану платформу Nexus.

– Прискорення переходу на уніфіковану матрицю комутації.

– Побудова більше масштабованої, ефективної й стійкої архітектури ЦОД за допомогою послуг Cisco Nexus.

– Керування мережею з можливістю безпосереднього звертання до інженерів Cisco і великій кількості технічних ресурсів у будь-який час за допомогою послуги Cisco SMARTnet.

Сімейство комутаторів Cisco Nexus надає широкий набір переваг для діяльності організації.

Простота

– Підвищення ефективності, спрощення мобільного доступу для фізичних і віртуальних машин і сервісів, а також забезпечення повного контролю для будь-якої топології.

– Набагато більше швидке надання доступу для фізичних і віртуальних розгортань завдяки автоматизації мережі й виділенню мережних ресурсів, а також більше тісної інтеграції з інструментами координації, автоматизації й хмарних платформ.

– Більш просте усунення неполадок за рахунок централізованого керування матрицею комутації для фізичних і віртуальних навантажень.

Конвергенція

Комутатори Cisco Nexus спрощують мережа центра обробки даних за рахунок конвергенції локальних обчислювальних мереж і мереж зберігання даних з використанням протоколів DCB, Fibre Channel over Ethernet (FCoE) й уніфікованих портів:

– Низька сукупна вартість володіння (зниження до 50 %).

– Скорочення капітальних витрат за рахунок зменшення кількості адаптерів хостів, комутаторів і кабелів.

– Скорочення експлуатаційних витрат за рахунок зменшення енергоспоживання, зниження вимог по охолодженню, вільному місцю в стійці й займаній площі.

– Поетапна установка нових систем без повної модернізації.

– Мінімальні порушення виконуваних операцій і процесів керування.

Масштабованість

Комутатори Cisco Nexus допомагають підприємствам масштабувати більше складну за рахунок віртуалізації робоче навантаження, ураховувати швидке поширення віртуальних машин і вирішувати проблеми, що стосуються хмарних обчислень:

– Об'єднання всіх вузлів мережі в єдине середовище.

– Підтримка ефективного доступу й використання ресурсів незалежно від обсягу або області охвату.

– Створення відказостійких, масштабованих мереж з передбачуваними характеристиками й невисокою складністю.

– Спрощення керування за рахунок можливості розширення матриці комутації.

Інтелектуальні можливості

Комутатор Cisco Nexus надає доступ до унікальних сервісів безпосередньо через матрицю комутації мережі. Цей комутатор розширює мережу, зберігаючи її підконтрольність, для об'єднання всіх мережних сегментів у єдине, розширене середовище з погодженими сервісами й політиками:

– Погоджене надання доступу до сервісів для додатків або робочих навантажень.

– Автоматичне збільшення пропускної здатності для доступу до сервісів.

– Прискорення розгортання додатків з дотриманням заданих політик замість внесення змін у фізичну інфраструктуру.

Побудова центрів обробки даних нового покоління

Віртуалізація ресурсів є центральним елементом архітектури Cisco Data Center Business Advantage. Вона допомагає скоротити витрати по трьох напрямках:

– збільшення коефіцієнта використання ресурсів;

– підвищення стійкості бізнесу;

– прискорення реакції на зміну потреб бізнесу.

Цим центрам обробки даних нового покоління потрібна мережна інфраструктура, що надає доступ до повного арсеналу технологій, включаючи віртуалізацію серверів і уніфіковану матрицю комутації.

Продукти сімейства Cisco Nexus

– Cisco Nexus серії 7000: масштабована платформа для центрів обробки даних.

– Комутатори Cisco Nexus серії 5000: спрощення процесу перетворення центра обробки даних.

– Модуль комутації Cisco Nexus 4001I для IBM BladeCenter.

– Комутатор Cisco Nexus серії 3000: зверхмалі затримки й комутація рівнів L2/L3 зі швидкістю середовища передачі.

– Пристрої розширення матриці комутації Cisco Nexus серії 2000.

– Комутатори Cisco Nexus серії 1000V.

Продукти Cisco Catalyst

– Ethernet-комутатор Cisco Catalyst 4948E.

– Комутатор Cisco Catalyst серії 4900M.

Комутатори Cisco Nexus серії 9000

– Основа для розгортання інфраструктури, орієнтованої на додатки (ACI).

– Швидкість комутації до 1,92 Тбіт/с на з'єднання.

– Висока щільність Ethernet зі швидкістю передачі даних 1, 10 і 40 Гбіт/с і готовність до підтримки 100 Гбіт/с у майбутньому.

– Робота в стандартній операційній системі Cisco NX-OS, а також у режимі ACI.

– Повна підтримка ретрансляції, маршрутизації й шлюзів у віртуальній розширюваній локальній мережі (VXLAN).

Комутатори Cisco Nexus серії 7000

– Найвища швидкість комутації: до 1,3 Тбіт/с на з'єднання, більше 83 Тбіт/с на шасі.

– Широкі можливості масштабування за рахунок підтримки Ethernet-З'єднань зі швидкістю передачі даних 1, 10, 40 і 100 Гбіт/с.

– Самий широкий у галузі набір функцій для розгортання центрів обробки даних.

– Розширені сервіси; висока доступність; відновлення програмного забезпечення без переривання роботи (ISSU).

– Ідеально підходять для рівнів доступу, агрегації і ядра мережі центра обробки даних.

Комутатори Cisco Nexus серії 6000

– Розгортання до 96 портів на 40 Гбіт/с або 384 портів на 10 Гбіт/с у компактному корпусі з форм-фактором 4U.

– Інтегрована підтримка функцій рівнів 2 і 3 при роботі зі швидкістю мережі й з малими затримками.

– Підвищення ефективності за рахунок підтримки архітектури Cisco Fabric Extender (FEX) і повного набору аналітичних інструментів.

– Підтримка інтерфейсів FCo зі швидкістю передачі даних 40 Гбіт/с для конвергенції локальних мереж з мережами зберігання даних.

– Ідеально підходять для розгортання систем рівня агрегації з обмеженнями по доступі й вільному просторі.

Комутатори Cisco Nexus серії 5000

– Підвищення адаптивності за рахунок більше швидкої й більше великої віртуалізації центра обробки даних.

– Широкий вибір варіантів підключення, включаючи Ethernet-порти зі швидкістю передачі даних 1, 10 і 40 Гбіт/с, 10 GBase-T, FC і FCo.

– Підтримка трафіку локальних мереж і мереж зберігання даних рівнів 2 і 3.

– Підвищення стійкості бізнесу завдяки забезпеченню безперервності діяльності організації.

Комутатори Cisco Nexus серії 4000

– Зниження вимог по кількості комутаторів, мережних плат і електроживленню.

– Застосування спеціалізованих інтегральних мікросхем серії Unified Switch (Ethernet-комутатор на 10 Гбіт/с, що працює зі швидкістю мережі й мінімальних затримок).

Комутатори Cisco Nexus серії 3000

– Гнучкість і продуктивність для розміщення комутаторів у стійках із серверами (top-of-rack).

– Функції Cisco Algo Boost з дуже малими затримками, поліпшеним контролем і керуванням.

– Спрощення керування за рахунок застосування сценаріїв Python, технології Energy Efficient Ethernet (EEM) і інструментів керування з підтримкою XML.

– Підвищення адаптивності бізнесу за рахунок використання комплексної програмно-визначаємою мережної інфраструктури (SDN), включаючи OpenFlow і Cisco OnePK.

Пристрою розширення матриці комутації Cisco Nexus серії 2000

– Можливість розширення матриці комутації з використанням централізованого керування.

– Ethernet-інтерфейси для підключення серверів зі швидкістю 100 Мбіт/с, 1 і 10 Гбіт/с із портами каскадування на 10 і 40 Гбіт/с.

– Скорочення витрат на прокладку кабелів для ЦОД і зменшення займаного встаткуванням простору за рахунок оптимізації кабельних з'єднань між стійками.

– Можливість підтримки блейд-серверів партнерів (наприклад, HP і Fujitsu).

Комутатор Cisco Nexus 1000V

– Інтеграція безпосередньо з гіпервізорами серверів.

– Мережні сервіси, що враховують вимоги віртуальної машини, такі як забезпечення мобільності віртуальних машин, у хмарних середовищах.

– Прискорення віртуалізації серверів і спрощення керування.

– Скорочення сукупної вартості володіння, погодженість і контроль стану мережі.

Комутатори Cisco Catalyst серії 6500

– Захист капіталовкладень у традиційні центри обробки даних.

– Поліпшення операційного керування, підвищення доступності.

– Спрощення інфраструктури й застосування засобів всебічного захисту мережі.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

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

Основні можливості Delphi 10.4.1:

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

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

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

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

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

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

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

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

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

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

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

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

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

RAD Studio 10.4 Короткий огляд:

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

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

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

Істотне поліпшення Delphi Code Insight

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

Delphi Custom Managed Records

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

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

Єдине керування пам'яттю

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

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

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

Розширена підтримка бібліотек C++

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

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

Win 64-відладник і збирач для C++

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

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

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

Змінені стилі VCL для High DPI

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

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

Нові High DPI стилі й стилізація окремих VCL компонент

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

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

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

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

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

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

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

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

Універсальний інсталятор для установки Online і Offline

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

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

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи макровіртуалізації SDDC.

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

Дата-центр (центр обробки даних, ЦОД) надає спеціальні захищені канали для здійснення міжнародного зв'язку. Гарантовані безпека, надійність дата-центра, а також максимальна швидкість зараз є комерційно затребуваними на світовому ринку.

Основні послуги Дата-центра

Основними послугами є:

– Оренда стійка.

– Оренда серверів.

– Colocation (фізичне розміщення серверів).

– VPS.

– Віртуальний хостинг.

Додаткові послуги

Також існує ряд і додаткові послуги:

– Резервне копіювання (бекап).

– Хмарні рішення.

– Адмініструємий сервер.

– Вилучений робочий стіл.

Технології, застосовувані в дата-центрах

Високотехнологічна інфраструктура, що забезпечує безперебійну роботу встаткування в ЦОД – головна характеристика сучасного дата-центра. Для цього площадка Дата-центра обладнається системами клімат контролю, безперебійного електроживлення, безпеки й ін. системами життєзабезпечення.

Основними технічними характеристиками надійного датацентра є:

– Наявність спеціального будинку, призначеного для розміщення дата-центра.

– Гарантовані система електроживлення й кондиціювання.

– Промислова система вентиляції.

– Система автоматичного пожежогасіння.

– Наявність охорони й контролю доступу.

– Наявність дизельного генератора.

– Доступ 24/7 до серверного встаткування як для персоналу, так і для клієнтів.

Часто дата-центри розташовують безпосередньо в близькості від крапки присутності декількох операторів зв'язку або вузла зв'язку, щоб забезпечити моментальний обмін гігантських обсягів даних і швидке завантаження даних з будь-якої крапки миру. Ключовим критерієм оцінки надійності ЦОД є аптайм (uptime), тобто час доступності сервера.

Навіщо так потрібний Дата-центр?

На перший погляд питання здається простим, адже сервера й інше встаткування для роботи необхідно десь розміщати, але навіщо будувати для цього цілі будинки, якщо можна обійтися серверним приміщенням. Насамперед, відповідь полягає в економічній вигоді. Консолідація обчислювальних ресурсів і засобів зберігання даних у ЦОД дозволяє зменшити загальну вартість експлуатації IT-ресурсів.

Фахівці виділяють наступні шляхи оптимізації фінансових витрат за умови розміщення встаткування в комерційному ЦОД:

– відсутність необхідності створення власної інфраструктури;

– знижки на інтернет-підключення, оскільки датацентр має власні підключення до основних Інтернет-Вузлам;

– скорочення витрат на адміністрування завдяки обслуговуванню серверів співробітниками дата-центра;

– зменшення орендної плати у зв'язку з відсутністю займаної серверами площі;

– перерозподіл навантажень, з метою більше оперативного рішення бізнес-завдань.

Інакше кажучи, розміщення серверів у надійних Дата-центрах не тільки забезпечує схоронність, цілісність і захист даних, але й оптимізує фінансові витрати кожної окремої компанії.

3.2 Розробка структурної схеми

Одним з важливих цеглинок програмно обумовленого мережного середовища є технологія Software Defined Networking (SDN). Забезпечуючи централізоване керування високопродуктивними мережними комутаторами, SDN знаходить собі всі нових прихильників.

Основна ідея SDN – поділ керуючих і транспортних функцій мережної інфраструктури. Весь «інтелект» зосереджений в окремій апаратній/програмній базі – виділеному керуючому контролері SDN, що відповідно до заданих правил визначає роботу мережі. Комутатори при цьому виконують елементарні дії з пакетами й втрачають більшості інтелектуальних функцій.

Керування трафіком – взаємодія керуючого контролера й комутаторів – здійснюється за допомогою спеціальних протоколів (найбільш перспективний з них і активно розвивається – OpenFlow), які оперують поняттям «потік» (flow). Через них виконуються різноманітні дії із трафіком: заборона, дозвіл, перенапрямок і т.д. SDN забезпечує гнучкість керування мережею й істотно спрощує її адміністрування.

Але основна ізюминка SDN все-таки в іншому. Контролер SDN повинен мати засобу інтеграції із системами оркестрації, а в перспективі – з додатками. Це дозволить забезпечити керування мережними ресурсами на основі актуальних запитів з боку інформаційних систем. Наприклад, мережа буде динамічно

виділяти більше широку смугу пропущення на час сеансу зв'язки-зв'язок^-зв'язкові-зв'язку-сполучення-відео^-конференц-звЙРзку, а потім перерозподіляти її на користь інших додатків.

Розуміння того, що в мережі є в наявності не тільки пакети й потоки, але й додатки, – одна із ключових особливостей мереж SDN, що забезпечує їм майбутнє в корпоративному світі. І саме централізована архітектура SDN уможливлює реалізацію цього завдання.

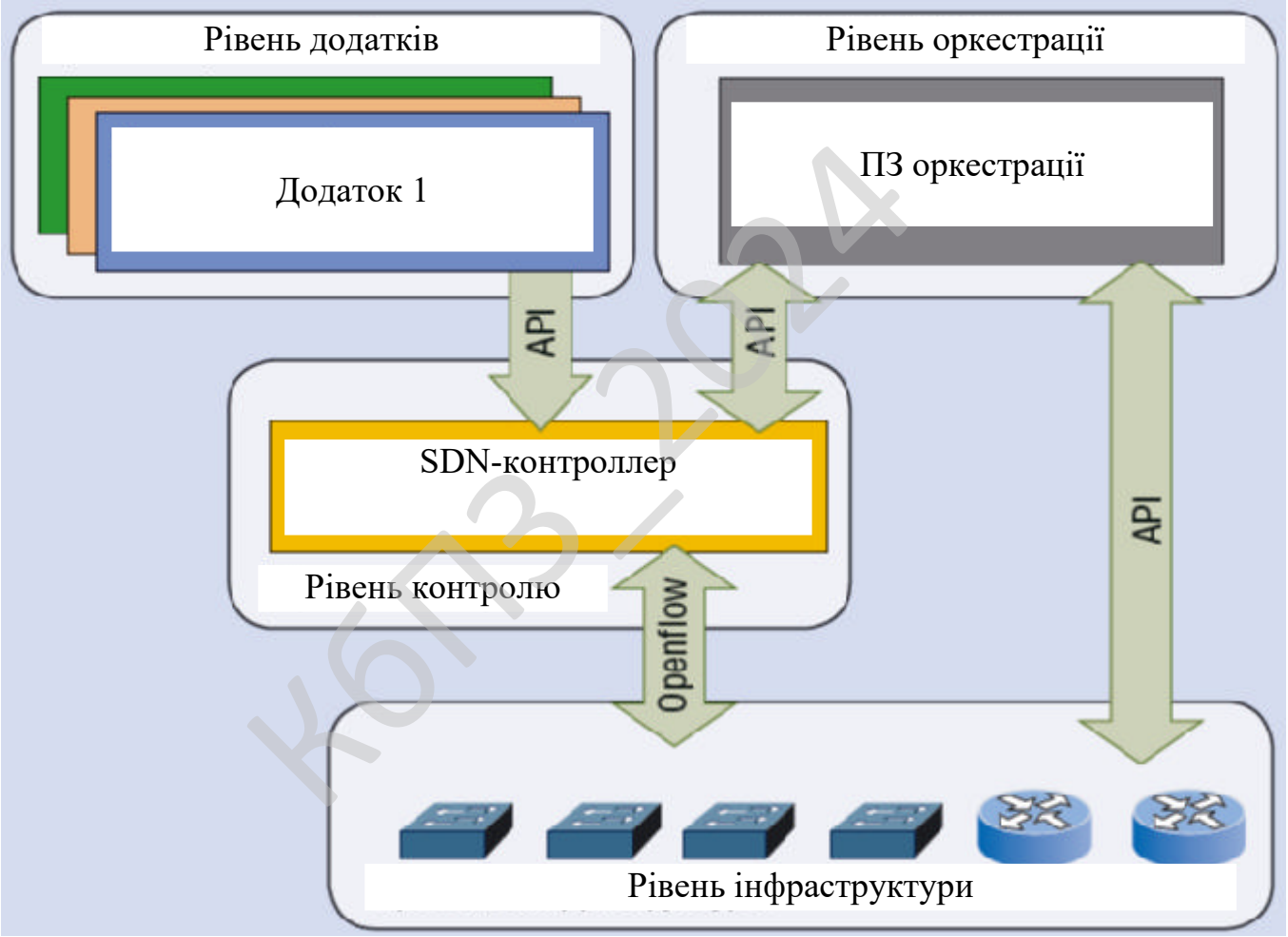

Рисунок 3.1 – Структурна схема системи

Далеко не всі завдання можуть бути вирішені засобами, які надає SDN. Зокрема, це стосується складних завдань обробки трафіку й організації міжмережного взаємодії, коли SDN органічно доповнюється віртуалізацією

мережних функцій (Network Function Virtualization, NFV). Уже існують як Open Source, так і комерційні реалізації у віртуальному середовищі мережних пристроїв – комутаторів, маршрутизаторів, балансувальників навантаження, міжмережних екранів і т.д.

У доступному для огляду майбутньому очікується, що в ЦОД залишаться тільки сервери (вони ж розподілені СЗД) і високопродуктивні комутатори з обмеженим набором функцій. Подібна інфраструктура стане забезпечувати переважну більшість вимог сучасного ПЗ, що споконвічно проектується з урахуванням хмарного застосування й віртуалізації.

Серед відзначених переваг NFV майже із трикратним відривом лідирує показник гнучкості: віртуальний сервер значно простіше придбати, масштабувати й т.д., ніж фізичний пристрій. А показники капітальних і операційних витрат, хоча й відзначаються респондентами в якості істотних, не перебувають на лідируючих позиціях.

Якщо конкретизувати результати опитування, можна відзначити наступні переваги:

– можливість створити персональне мережне оточення для кожного додатка або абонента в хмарній інфраструктурі;

– скорочення номенклатури використовуваних пристроїв;

– зниження витрат на обслуговування й забезпечення функціонування IT-інфраструктури: віртуальні мережні пристрої розміщаються у віртуалізованому середовищу на стандартних, однотипних серверах, тому для їхньої установки не потрібно окремого місця в стійці, а для обслуговування – широкої номенклатури ЗІП;

– зменшення строків поставки: NFV – це ПЗ й ліцензії, їх поставляти значно простіше, ніж устаткування;

– простота тиражування й масштабування поряд зі спрощенням автоматизації;

– нескладне відновлення за короткий час: резервні копії конфігурацій віртуальних мережних пристроїв або просто образи їхніх віртуальних машин можна відновити за лічені мінути на інших серверах.

Завдяки застосуванню NFV, провайдер дуже швидко може надати абонентові необхідна кількість і потрібна номенклатура мережних пристроїв, які той самостійно настроїть у відповідності зі своїми унікальними завданнями. До речі, можливість повністю управляти всіма настроюваннями орендованих мережних пристроїв якісно відрізняє послугу від варіанта, коли мережі набудовує сам провайдер (в останньому випадку замовник не одержує контролю над настроюваннями, експлуатація мережі для оператора багаторазово ускладнюється й у підсумку рішення виявляється значно дорожче для обох сторін).

У ситуації, коли не вся мережа віртуалізована, виникає необхідність забезпечити стик між фізичною й віртуальною мережами, а виходить, доводиться вирішувати завдання керування й настроювання такої «гібридної» мережі. Традиційні методи, як правило, не влаштовують або через їх недостатні функціональні можливості, або внаслідок обмеженого набору підтримуваних пристроїв. Тому IT-галузь стала шукати обхідні шляхи.

Дійсно, замість того щоб щораз займатися переконфігуруванням фізичної мережі, чому б не «дати їй спокій», один раз настроївши й забезпечивши тільки базові функції – надійність, продуктивність, загальну зв’язність? У результаті з'явився новий вид мереж – створювані поверх фізичних накладені (Overlay), або логічні, мережі. Подібне робилося й раніше: під наведене визначення попадає добре відомий стандарт IEEE 802.1q (VLAN). Різниця в тім, що протоколи для накладених мереж орієнтовані на маршрутизуємі мережі й передбачають істотно більше міток для їхньої ідентифікації (як правило, 16 млн). У загальному випадку накладена мережа створюється за допомогою програмних або апаратних комутаторів (gateway) і протоколів тунелювання (VXLAN, NVGRE, STT, GENEVE).

Завдяки накладеним мережам адміністратор віртуального середовища може конфігурувати тунелі між віртуальними машинами, не виконуючи ніяких настроювань фізичних комутаторів. Така мережа дозволяє реалізувати необхідні додаткам сервіси поверх будь-якої надійної мережної інфраструктури. Вона надає наступні переваги:

– забезпечення зв’язності віртуальних машин у різних сегментах мережі й навіть у різних ЦОД;

– підвищення стабільності роботи мережі за рахунок використання маршрутизуємої мережі як транспорт;

– можливість спільної роботи у віртуальній обчислювальній інфраструктурі й взаємодії із пристроями NFV;

– повна сумісність із існуючою мережною інфраструктурою завдяки тому, що всі протоколи накладених мереж працюють за принципом інкапсуляції й використовують стандартний формат кадру Ethernet.

До недоліків технології можна віднести потребу в багатоадресному, або груповому розсиланні (multicast) (вона повинна підтримуватися у фізичній мережній інфраструктурі) і високу завантаженість серверів внаслідок необхідності інкапсуляції-декапсуляції даних, що пересилаються. Втім перший з названих недоліків уже частково усунутий: на ринку з'явилися рішення для створення накладених мереж, де не потрібна підтримка багатоадресного розсилання.

Обговорюючи програмно обумовлений ЦОД, ми торкнулися актуальних технологій в області інженерної й мережної інфраструктур. Не менш важливою складової SDDS є віртуалізація систем зберігання (Software-Defined Storage, SDS). СЗД програмно обумовленого типу повинна відповідати декільком загальноприйнятим базовим принципам. Концепція SDS припускає «сервісний» підхід, тобто сукупність програмно-апаратних, організаційних і сервісних елементів, убудованих у корпоративні IT, включаючи, наприклад, раніше далеке рядовому адміністраторові СЗД поняття «сервісний каталог».

У якості SDS розглядається такий тип сервісу зберігання даних, коли фізична складова СЗД (масиви, диски, контролери, порти й т.д.) відділена від логічні й організаційної (логічні томи, сервіси, політики доступу й т.п.). Скоріше, тут варто говорити про черговий рівень апаратної абстракції (Hardware Abstraction Layer, HAL), що потрібен був для ефективної підтримки віртуалізації й сервісу зберігання в рамках концепції SDDC.

Технології SDN, NFV, DCIM, SDS і інші викликають інтерес у багатьох українських компаній – провайдерів IT-послуг, але повноцінних реалізацій SDDC у нашій країні усе ще немає. Це можна пояснити декількома принциповими причинами.

Так, на ринку відсутні готові рішення для інтеграції системи DCIM і ПЗ керування віртуальними машинами. Компанії прийдеться самостійно вирішувати це питання за допомогою своїх IT-фахівців або команди системного інтегратора. Та й сам вибір DCIM викликає певні складності.

У цей час це ПЗ пропонують дві групи виробників. Перша містить у собі вендорів, що історично спеціалізуються на створенні рішень для інженерної інфраструктури ЦОД. При вибудовуванні системи керування фізичними активами центра обробки даних вони дотримуються підходу знизу нагору – починаючи зі збору детальних даних про стан компонентів інженерної інфраструктури.

Друга група – виробники, що розробляють рішення для комплексного керування IT-інфраструктурою. Подібні системи мають широкого функціонала для проведення детальних інвентаризацій, планування розміщення встаткування в ЦОД, прогнозування, оперативного моніторингу електровикористання й багато чого іншого. У цьому випадку підхід протилежний – зверху долілиць.

Вибір того або іншого рішення залежить від конкретних умов. Компанія повинна визначитися з тим, яка система необхідна саме їй (з урахуванням стану інфраструктури й планів розвитку), провести RFI, розробити KPI, сформувати шортів-аркуш і виконати тестові випробування. Все це сполучено зі значними

часовими й трудовими витратами. Логічно, що багато провайдерів відкладають впровадження DCIM у довгий ящик, відсуваючи й перехід до програмно обумовленого середовища.

У свою чергу, поширення технологій SDN, NFV і накладених мереж сьогодні стримується їхньою новизною й відсутністю повноцінних, готових до впровадження рішень. У центрах обробки даних використовується звичне, давно застосовуване мережне «залізо», IT-фахівці досконально знають його плюси й мінуси, на відміну від особливостей поводження віртуалізованих мережних пристроїв. Зміна парадигми вимагає додаткових фінансових вкладень, але компанії й так уже витратилися на побудову традиційної мережі. Розраховувати ж на контролери SDN з відкритими вихідними кодами поки особливо не доводиться.

ПЗ Open Source для SDN являє собою по більшій частині «заготовку», яку потрібно допрацьовувати, що по кишені не кожній компанії. Очікування ринку із приводу дешевих SDN-комутаторів теж не виправдуються: оскільки необхідна для підтримки необхідної кількості потоків пам'ять (Ternary Content Addressable Memory, TCAM) є дорогим компонентом, вартість усього пристрою зростає. До того ж вендори по очевидних причинах не роблять свої рішення повністю відкритими: ніхто із провідних виробників не упустить можливості «прив'язати» компанію-замовника пропрієтарними доробками.

Проблемою SDS на даний момент є відсутність єдиних стандартів для програмних інтерфейсів і протоколів, що забезпечують взаємодію між фізичним, логічним і організаційним рівнями рішення. По суті, програмно обумовлена СЗД дня сьогоднішнього й найближчого майбутнього на 90% складається з віртуалізованої СЗД і на 10% – із програмного інструментарію (який зараз тільки розробляється), що дозволяє ефективно управляти СЗД і взаємодіяти з іншими компонентами SDDC.

Разом з тим апаратні рішення рано або пізно зажадають заміни в силу їх фізичної й моральної амортизації, тому при плануванні подальшого розвитку IT-

інфраструктури компанії варто враховувати можливість розширення її віртуального рівня. До цього процесу необхідно попередньо готуватися, перевіряючи різні компоненти й варіанти реалізації рішень.

3.3 Розробка функціональної схеми

На рисунку 3.2 наведена функціональна схема розробленої системи. З рисунка 3.2 бачимо, що для реалізації імітаційної моделі встановлюється два хоста ЦОД та два маршрутизатора ЦОД сервісу макровіртуалізації SDDC, через які йде трафік. Зафарбована область позначає топологічні елементи мережі ЦОД (канали й маршрутизатори).

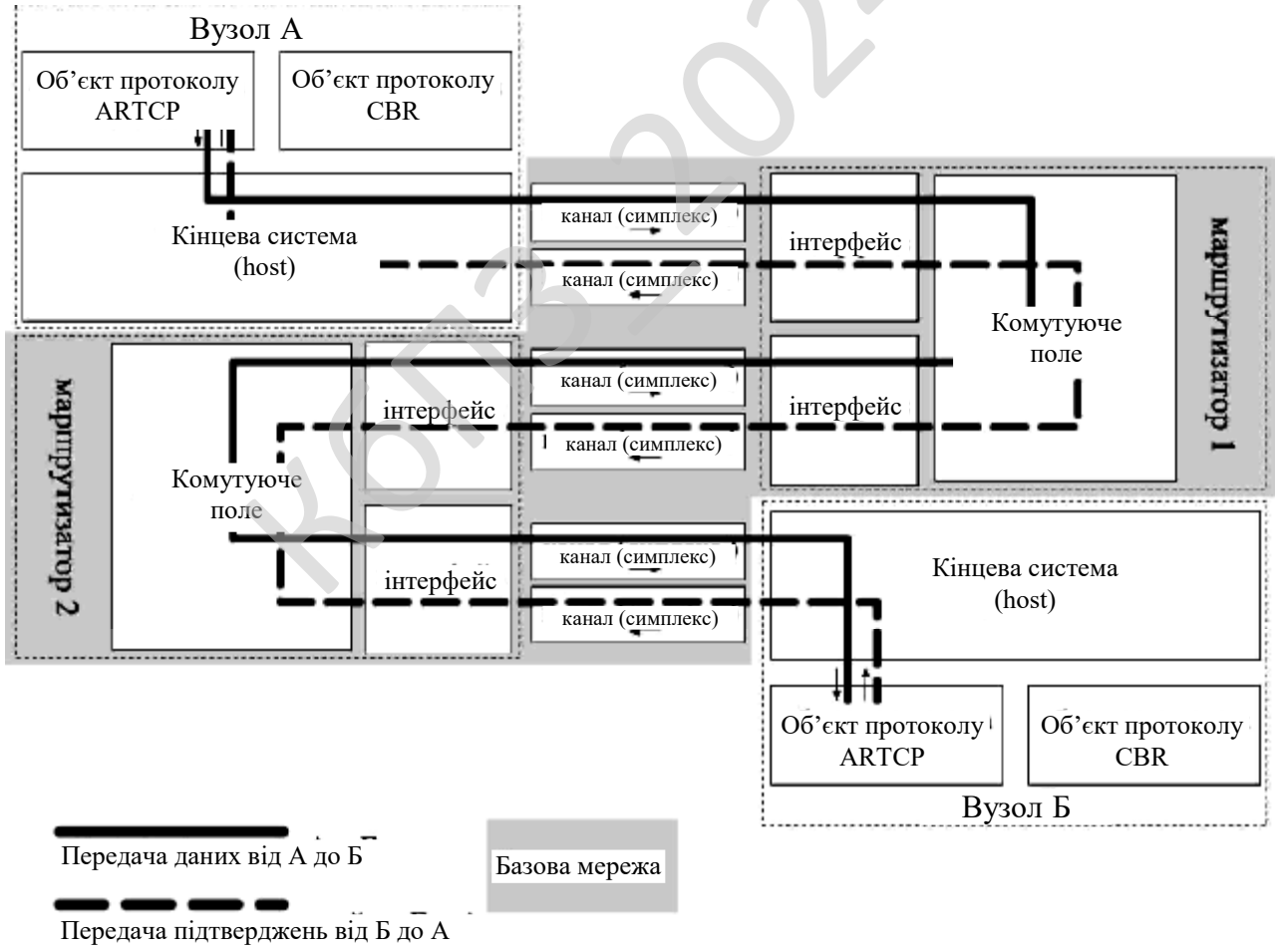

Рисунок 3.2 – Функціональна схема системи

На функціональній схемі вказані усі елементи, які необхідні для передачи пакета по сервісу макровіртуалізації SDDC у мережі ЦОД. До них відносяться:

– об’єкти протоколу ARTCP та CBR;

– хости ЦОД;

– симплексні канали передачі даних ЦОД;

– інтерфейси по яким відбувається з’єднання між маршрутизаторами ЦОД;

– маршрутизатори ЦОД;

– комутуюче поле у маршрутизаторі ЦОД (таблиця маршрутизації).

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

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі.

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування).

Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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.

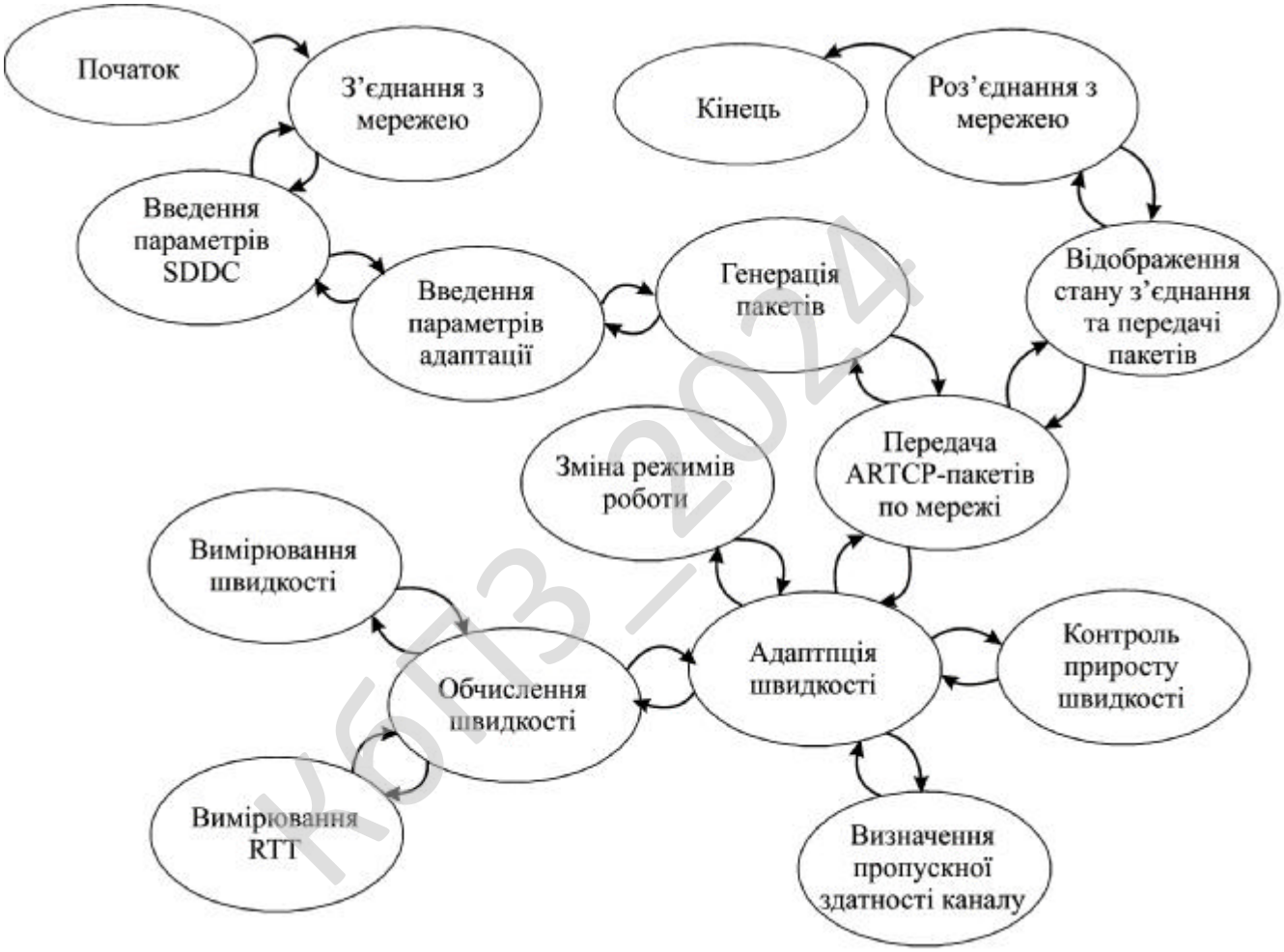

Рисунок 3.3 – Діаграма взаємодії процесів

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

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

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

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

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

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

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

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

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

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

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

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

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

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

– діаграма класів;

– діаграма компонент;

– діаграма об'єктів;

– діаграма розгортання.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то

весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

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.

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

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

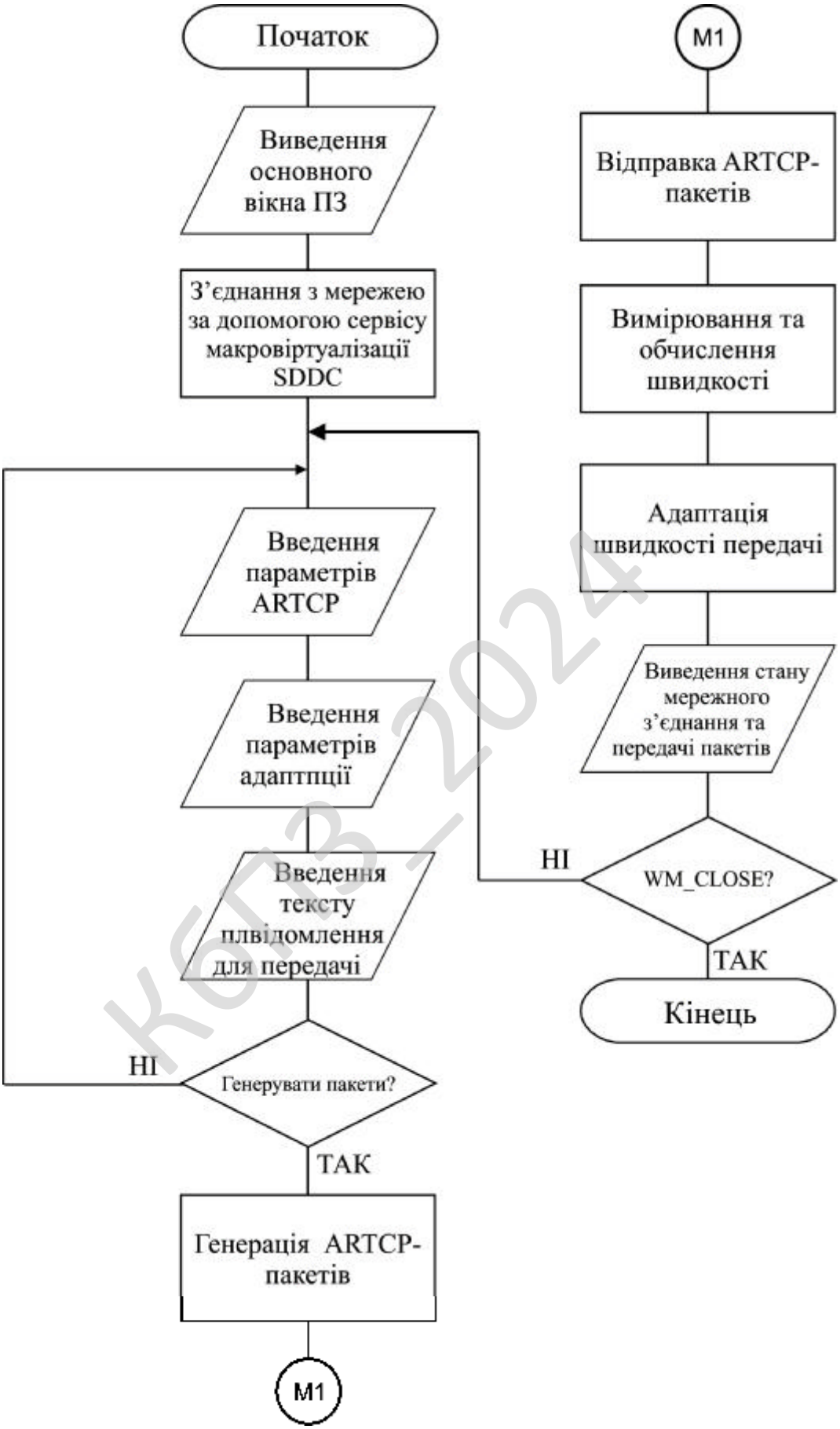

Рисунок 4.1 – Блок-схема основної програми

Розглянемо формат повідомлення, використовуваний у моделі, ієрархію й реалізацію її основних класів.

Формат повідомлення. Нові сегменти створюються усередині об'єктів протоколів ARTCP або CBR. Всі об'єкти топології передають один одному покажчик на область пам'яті, що містить відповідний сегмент, замість копіювання реальних даних. Крім того, розміри заголовка сегмента й поля корисного навантаження задаються в самій структурі даних, і всі об'єкти визначають розмір сегмента, інтерпретуючи значення цих полів.

Оскільки посилання на об'єкт може одночасно втримуватися у всіх об'єктах, де реалізована буферізація: ARTCP, link, interface, то щоб уникнути конфліктів лічильник посилань зберігає кількість об'єктів, у буферах, в яких перебуває посилання на даний сегмент. Звільнення області пам'яті, що містить сегмент, дозволено тільки у випадку обнуління лічильника. За винятком полів "TI" і "PS" і службових полів link, id, всі інші поля сегмента відповідають полям стандартного TCP заголовка.

Поле заголовка ARTCP у моделі не містить перевірочної суми, ушкодження сегментів у транзиті можуть моделюватися шляхом реалізації випадкового відкидання сегментів на мережних вузлах або в кінцевій системі з імовірністю, що відповідає ймовірності бітових помилок середовища передачі, яку необхідно змоделювати.

Всі класи, що описують топологічні елементи мережі: кінцева система (host), канал (link), маршрутизатор (router), інтерфейс (interface), є спадкоємцями базового класу element.

Інтерфейс кожного із цих класів визначається трьома найважливішими компонентами:

– прийом сегмента ;
– запит сервісу;
– обробка переривання.

Ці методи є перевизначенням віртуальних функцій базового класу. Крім того, кожний похідний клас розширений додатковими специфічними для нього методами. За допомогою цих елементів інтерфейсу моделюється передача даних у системі. Кожний зі спадкоємців класу element посилається на елемент, з яким він зв'язаний топологічно, по покажчику на базовий клас.

Екземпляр кожного з наведених класів використовує виклик топологічно прив'язаного до нього об'єкта для передачі йому сегмента. Залежно від наявності ресурсів метод викликуваного об'єкта може прийняти або не прийняти сегмент на обслуговування.

Метод кожного екземпляра викликається з основної програми для обробки черги й ухвалення рішення про обслуговування чергового сегмента.

Об'єднання елементів мережі в топологічну схему здійснюється за допомогою виклику спеціального методу для кожного екземпляра всіх класів.

Клас link: аспекти реалізації. Клас, що моделює канал, одержує значення пропускної здатності й затримки передачі при ініціалізації. Структура даних класу реалізується однозв'язною динамічною чергою, у яку містяться сегменти, прийняті на обслуговування. Після початку обслуговування сегмента канал блокує наступні запити протягом часу $t=s/R$, де s – розмір сегмента, а R – швидкість каналу, тобто часу прийому сегмента розміру s у канал. У черзі каналу сегмент затримується протягом установленого часу затримки передачі. Після закінчення цього часу, об'єкт каналу викликає метод наступного об'єкта в топологічній схемі мережі. Якщо наступний об'єкт відмовляє в обслуговуванні сегмента, то цей сегмент відкидається. Для моделювання дуплексного зв'язку застосовуються 2 екземпляри каналу. Параметри каналів можуть бути різними для моделювання асиметричних систем.

Метод всіх екземплярів класу link викликається з головного циклу. Обробка переривання переводить внутрішній лічильник часу й викликає передачу готового сегмента з каналу наступному об'єкту топології.

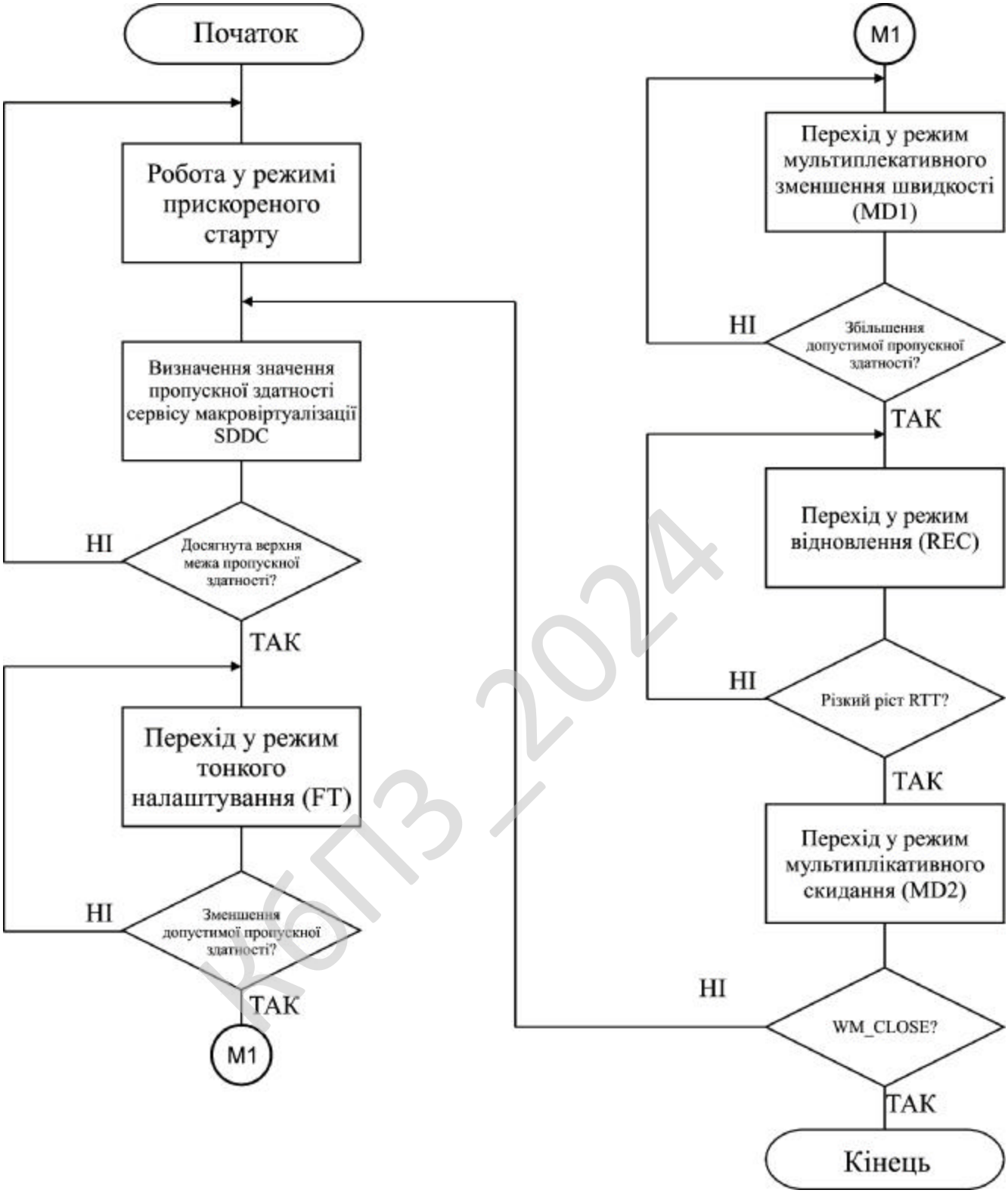

Рисунок 4.2 – Блок-схема роботи підпрограми

Для моделювання помилок середовища передачі екземпляр класу link може бути ініційований ініціалі затором, що перевантажується, який крім

параметрів швидкості й затримки каналу одержує також параметр відповідному резидентному значенню помилки передачі *BER*. З імовірністю $1 - (1 - BER)^s$ сегмент відкидається замість передачі наступному елементу топології. Таким чином, можна моделювати супутникові, радіо, стільникові канали.

Клас router: аспекти реалізації. Структура класу router є складною. У його склад входять кілька екземплярів класу interface. При ініціалізації класу router йому передаються два параметри: кількість інтерфейсів і об'єм буферного простору (максимальна довжина черги), однаковий для всіх інтерфейсів. Об'єкт маршрутизатора створює задана кількість екземплярів класу interface. Покажчики на кожний інтерфейс розташовуються в спеціальному масиві, проіндексованому один по одному створення інтерфейсів. Після цього на конкретний інтерфейс можна посилатися як router.interface(X), де X – номер потрібного інтерфейсу. Кожний інтерфейс при ініціалізації одержує покажчик на його об'єкт, що створив, класу router, по якому він може посилатися на методи класу router для передачі сегментів у матрицю комутації (позначена в схемах як "поле комутації").

При прийманні сегмента інтерфейс негайно передає його об'єкту маршрутизатора. Блокування при цьому відбутися не може, тобто моделюється не комутаційна матриця, що блокує. Об'єкт маршрутизатора вносить запис, що складається із двох полів: {номер інтерфейсу, адреса відправника} у таблицю маршрутизації. Після цього зіставлення адреси призначення сегмента із другим полем таблиці маршрутизації дає номер вихідного інтерфейсу для даного сегмента. При відсутності збігу із записами таблиці маршрутизації, сегмент передається для розсилання всім інтерфейсам маршрутизатора, крім того, з якого він був отриманий. Отриманий від комутуючої матриці, сегмент міститься у вихідну чергу інтерфейсу, якщо вільний простір у ній $Q_{max} - Q \geq s$, у противному випадку сегмент відкидається. Черга моделюється списком подвійної зв’язності, вишиковується динамічно в пам'яті [63]. Черга обслуговується зі швидкістю каналу, до якого підключений інтерфейс.

Метод обробки переривання кожного з інтерфейсів викликається методом обробки переривання об'єкта маршрутизатора. Також об'єкти маршрутизатора й інтерфейсу постачені методами для видачі звіту по поточних параметрах трафіку:

– середня швидкість;

– лічильники пропущених/відкинутих сегментів;

– таблиця маршрутизації;

– середня довжина черги.

Таким чином, об'єкт маршрутизатора моделює сучасний міжмережний пристрій з не комутаційною матрицею, що блокує, і буферизаціею на виході [67]. Можливо також розширення класу маршрутизатора алгоритмами RED і WFQ або CBQ [65,66].

Поводження запропонованої моделі маршрутизатора щодо побудови таблиці маршрутизації несуттєво й збігається зі схемою функціонування комутатора ЛОМ. Таблиці маршрутизації заповнюються на підставі спостереження за трафsком, а не в результаті роботи алгоритмів маршрутизації, тому не потрібно ніякої конфігурації маршрутизатора, у процесі роботи він навчається топології мережі.

Клас host: аспекти реалізації. Клас host має у своєму складі екземпляр класу ARTCP, що моделює сам протокол і екземпляр класу CBR, що моделює протокол передачі даних без підтверджень і керування потоком з постійної бітової швидкості. У кожний момент часу тільки один із протоколів може бути в активному стані.

При відправленні сегмента метод екземпляра класу host викликається одним із протоколів. У випадку прийому сегмента на обслуговування об'єктом каналу передачі, позитивне підтвердження вертається об'єкту протоколу, що викликав метод, і покажчик на область пам'яті, що зберігає сегмент передається об'єкту канального рівня. У випадку відсутності можливості передати сегмент на канальному рівні, метод об'єкта host повертає негативне підтвердження. При

ініціалізації екземпляра класу host він одержує мережну адресу. Окремо встановлюється мережна адреса призначення.

У випадку прийому сегмента від об'єкта каналу, метод передає покажчик протоколу обумовленому за значенням поля port заголовка сегмента. Сегменти, чия адреса призначення не збігається з адресою даного екземпляра вузла, відкидаються.

Метод класу host використовується для передачі запиту на обробку переривання об'єкту активного протоколу.

Об'єкт протоколу CBR. Протокол постійної швидкості передачі прямо відповідає реальному режиму передачі даних з постійною бітовою швидкістю, наприклад CBR в ATM [68] або (Circuit emulation services) CES [69]в IP мережах. Коректна взаємодія з таким режимом передачі даних є актуальною задачею будь-якого адаптивного транспортного протоколу в сучасних мережах з інтеграцією сервісів. Саме тому протокол CBR включений у модель середовища виконання ARTCP. Протокол CBR генерує сегменти й відправляє їх у мережу із передвстановленою швидкістю R_{CBR} , тобто часові проміжки між моментами початку послідовних трансляцій становлять $\tau_{CBR} = s / R_{CBR}$.

Підтвердження й, отже, ретрансляція сегментів і контроль швидкості не реалізується протоколом CBR.

Об'єкт протоколу ARTCP. Модель ARTCP реалізує:

– керування швидкістю відправлення сегментів за допомогою механізму ковзного вікна й алгоритму ARTCP;

– корекцію помилок передачі не пов'язану з управлінням передачі;

– швидку ретрансляцію сегментів і стандартну ретрансляцію по спрацьовуванню ТПП.

Основні призначення методів класу ARTCP такі:

– застосовуються для початкової синхронізації з'єднання;

– обчислюють значення області компенсації;

– запитують результат відправлення сегмента вузлом. З них три останні є відкритими й становлять інтерфейс класу.

До складу класу ARTCP входять два екземпляри класу Queue, які реалізують прийомний і передавальний буфер і методи для маніпуляції з ними.

Клас Queue. Даний клас реалізує схему стандартного керування потоком по методу ковзного вікна. Клас містить динамічний список подвійної зв'язності, у який записуються покажчики на сегменти, поставлені в чергу разом з екземпляром класу таймера, що реалізує ТПП. Методи класу Queue дозволяють здійснювати вставку сегмента із сортуванням, сканування списку, знаходження чергового сегмента готового до передачі. Ретрансляція реалізована за допомогою зміни черговості готових до відправлення сегментів.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою RC6 – симетричний блоковий криптографічний алгоритм, похідний від алгоритму RC5. Був створений Роном Рівестом, Меттом Робшау і Реєм Сіднеєм для задоволення вимог конкурсу Advanced Encryption Standard (AES). Алгоритм був одним з п'яти фіналістів конкурсу, був також представлений NESSIE і CRYPTREC. Є власницьким (пропрієтарним) алгоритмом, і запатентований RSA Security, одначе дія патентів сплила, і зараз алгоритм знаходиться у відкритому доступі. В той же час, "RC6" залишається зареєстрованою торговою маркою RSA.

Варіант шифру RC6, заявлений на конкурс AES, підтримує блоки довжиною 128 біт і ключі довжиною 128, 192 і 256 біт, але сам алгоритм, як і RC5, може бути налаштований для підтримки більш широкого діапазону довжин як блоків, так і ключів (від 0 до 2040 біт)[1]. RC6 дуже схожий на RC5 за своєю структурою і також досить простий у реалізації.

Є фіналістом AES, проте одна з примітивних операцій – операція множення, повільно виконується на певному обладнанні і ускладнює реалізацію шифру на ряді апаратних платформ і, що виявилося сюрпризом для авторів, на системах з архітектурою Intel IA-64 також реалізована досить погано. В даному випадку алгоритм втрачає одну зі своїх ключових переваг – високу швидкість виконання, що стало причиною для критики і однією з перепон для обрання як нового стандарту.

Деталі RC6

Так само, як і RC5, RC6 – повністю параметризована сім'я алгоритмів шифрування. Для специфікації алгоритму з конкретними параметрами, прийнято позначення RC6-w/r/b, де

– W – довжина машинного слова в бітах.

– R – число раундів.

– B – довжина ключа в байтах. Можливі значення 0 .. 255 байт.

Для того щоб відповідати вимогам AES, блочний шифр повинен працювати з 128-бітовими блоками. Так як RC5 – виключно швидкий блочний шифр, розширення його, щоб працювати з 128-бітовими блоками, привело б до використання двох 64-бітових робочих регістрів. Але архітектура і мови програмування ще не підтримують 64-бітні операції, тому довелося змінити проект так, щоб використовувати чотири 32-бітних регістри замість двох 64-бітних.

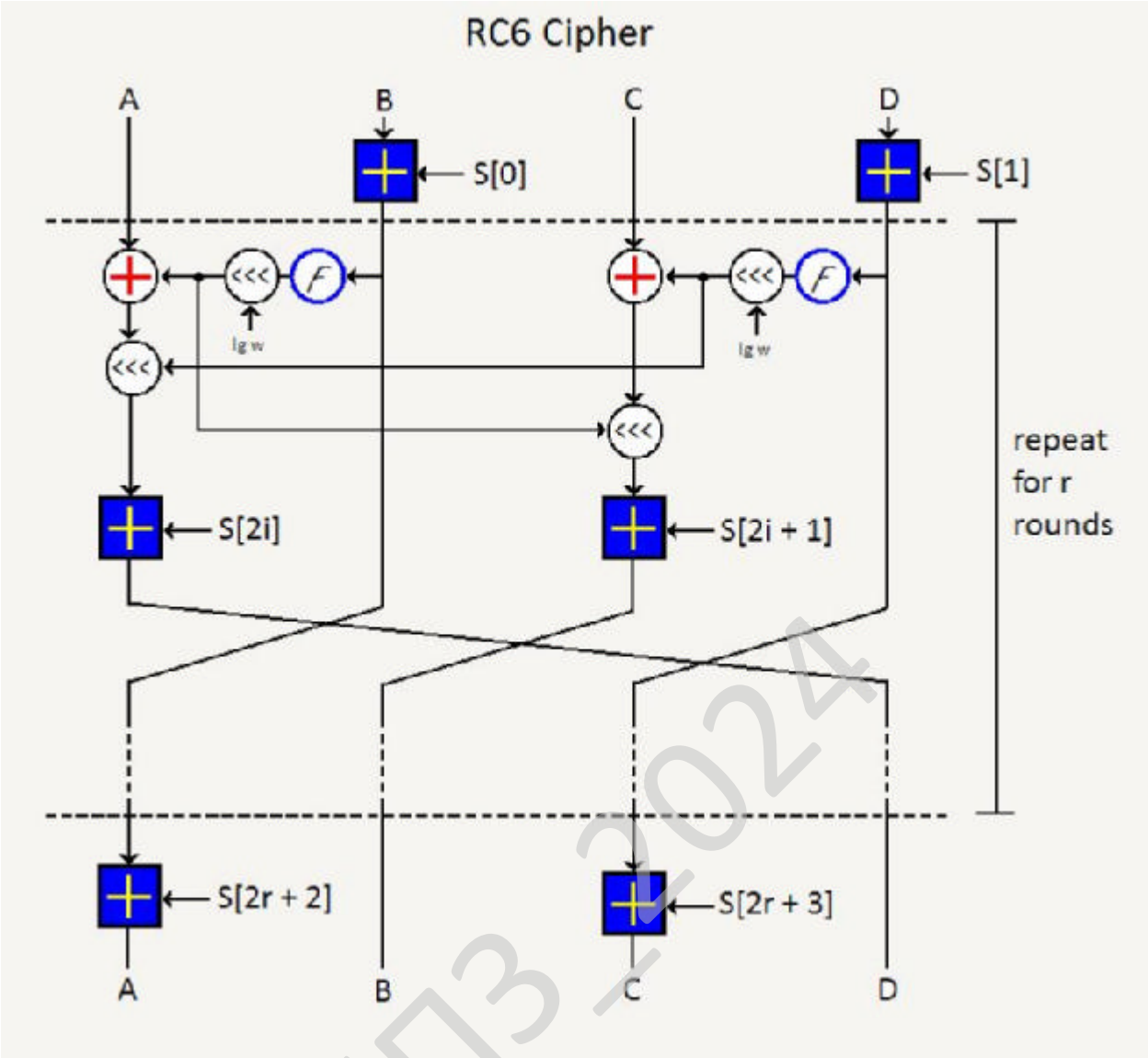

Рисунок 4.3 – Структура RC6

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

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

Розроблене програмне забезпечення сервісу макровіртуалізації SDDC складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Генерація пакету; Параметри; Довідка.

– Функції представлені у графічному вигляді (іконки).

– Вікно IP параметри.

– Вікно з вкладками: Параметри ARTCP; Параметри адаптації.

– Навігаційне вспливаюче меню яке визивається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

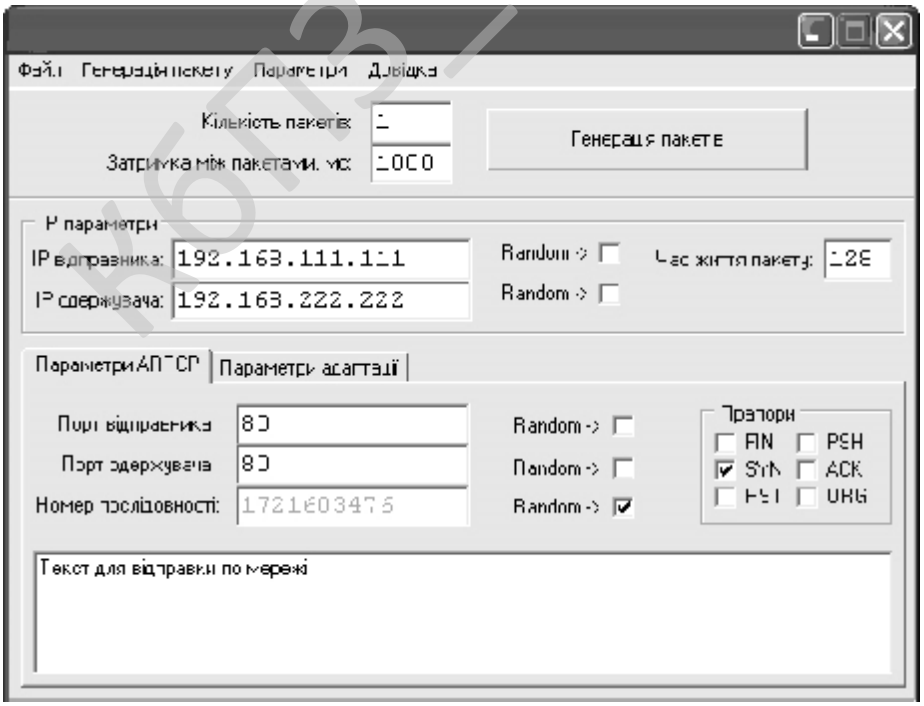

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

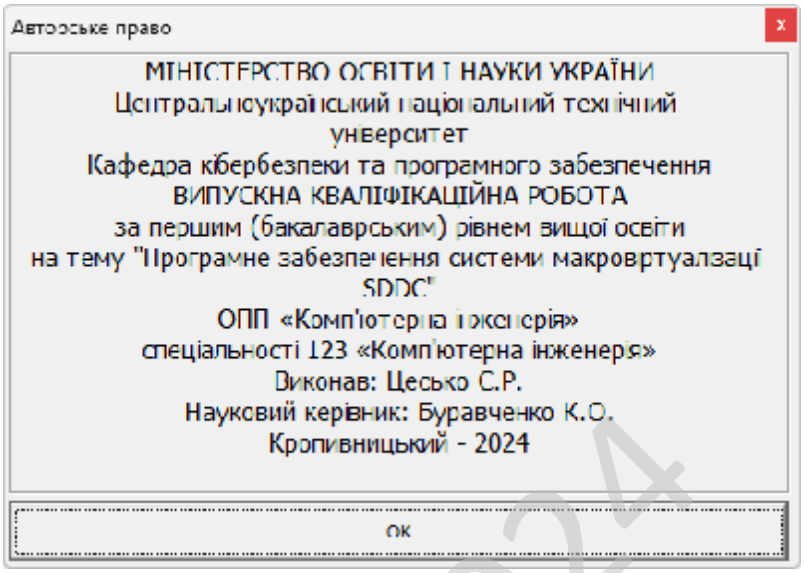

Рисунок 5.2 – Вікно розробника ПЗ

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 білої скриньки та чорної скриньки.

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Проводилось тестування 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 – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються. Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи макровіртуалізації SDDC.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем макровіртуалізації SDDC.

– Досліджена система макровіртуалізації SDDC.

– На основі отриманих результатів досліджень створена програмна реалізація системи макровіртуалізації SDDC.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання макровіртуалізації SDDC.

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

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

Програма реалізована на мові високого рівня Delphi 10. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи макровіртуалізації SDDC. 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 RC6.

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

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

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

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

5. 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.

6. 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.

7. 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.

8. 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.

9. 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.

10. 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.

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

12. 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.

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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

35. 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.

36. 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.

37. 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.

38. 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.

39. 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.

40. 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.

41. 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).

42. 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

43. 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.

44. 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.

45. 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.

46. 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.

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

48. 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.

49. 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.

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

51. 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.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи макровіртуалізації SDDC.

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 макровіртуалізації SDDC.

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 макровіртуалізації SDDC;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Delphi 10.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 61 аркуш.

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 13.06.2024 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

_____ Буравченко К.О.

Програмне забезпечення системи макровіртуалізації SDDC

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 27

Літера: РП

Кропивницький – 2024 року

Файл SDDC_GEN.DPR – головний файл проекту

```
program SDDC_Gen;

uses
  Forms,
  USDDC_Gen in 'USDDC_GEN.PAS' {Form1},
  artcpiphlp in 'artcpiphlp.pas',
  USDDC_Snd in 'USDDC_SND.PAS',
  About in About.PAS'; {Form2}

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(Tform2, Form2);
  Application.Run;
end.
```

Файл SDDC_GEN.PAS – генерація ARTCP-пакетів

```
unit USDDC_Gen;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, USDDC_Snd, ArtcpIpHlp,
  Menus, About;

type
  TForm1 = class(TForm)
    TopPanel: TPanel;
    MainPanel: TPanel;
    GroupBox2: TGroupBox;
    DstIpEdit: TEdit;
    SrcIpEdit: TEdit;
    TtlEdit: TEdit;
    RandomSrcIP: TCheckBox;
    RandomDstIP: TCheckBox;
    SrcPortEdit: TEdit;
    DstPortEdit: TEdit;
    SequenceEdit: TEdit;
    FlagsGroupBox: TGroupBox;
    FinFlag: TCheckBox;
    SynFlag: TCheckBox;
    RstFlag: TCheckBox;
    PshFlag: TCheckBox;
    AckFlag: TCheckBox;
    UrgFlag: TCheckBox;
    RandomSrcPort: TCheckBox;
    RandomDstPort: TCheckBox;
    UseRandomSeq: TCheckBox;
    DataMemo: TMemo;
    NumPacketsEdit: TEdit;
    DelayEdit: TEdit;
    ResultMemo: TMemo;
    GoButton: TButton;
    TabControl: TTabControl;
    SrcIPLabel: TLabel;
    DstIPLabel: TLabel;
    TTLLabel: TLabel;
    SrcPortLabel: TLabel;
    DstPortLabel: TLabel;
    SeqNoLabel: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    procedure RandomSrcIPClick(Sender: TObject);
    procedure RandomDstIPClick(Sender: TObject);
    procedure RandomSrcPortClick(Sender: TObject);
    procedure RandomDstPortClick(Sender: TObject);
    procedure UseRandomSeqClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure GoButtonClick(Sender: TObject);
    procedure TabControlChange(Sender: TObject);
  private
    { Private declarations }
    FRandomSrcARTCP, { save checkmarks for ARTCP,UDP and Ping }
    FRandomDstARTCP,
    FRandomSrcUDP,
    FRandomDstUDP,
```

```
    FRandomIdPing,
    FRandomSqPing: Boolean;
    FProtocolType: TProtocolType; { protocol tab }

    function  AcceptUserInput(Sender: TSenderIP; x: TProtocolType): Boolean;
    function  AcceptIPSettings(ip: TSenderIP): Boolean;
    function  AcceptArtcpSettings(artcp: TSenderARTCP): Boolean;
    function  AcceptUdpSettings(udp: TSenderUDP): Boolean;
    function  AcceptPingSettings(ping: TSenderICMP): Boolean;
    procedure PrintLine(s: String);

  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

var WarnedAboutW2k: Boolean = FALSE;

procedure WarnAboutW2k;
begin
  if NOT WarnedAboutW2k then
  begin
    WarnedAboutW2k := TRUE;
    if NOT Win2KDetected then
      ShowMessage('Попередження: Цей додаток розроблено під Windows 2000/XP, '
                  +'на вашому комп"ютері встановлена інша операційна система. '
                  +'Тому можуть виникати помилки у роботі з сокетами, '
                  +'так як для їх коректної роботи потрібний MS Winsock');
  end
end;

procedure TForm1.PrintLine(s: String);
// вивід рядка тексту
begin
  Form1.ResultMemo.Lines.Add(s)
end;

function TForm1.AcceptIPSettings(ip: TSenderIP): Boolean;
// Перевірка установочних параметрів IP заданих користувачем
begin
  Result := FALSE;

  ip.UseRandomSrcIP   := RandomSrcIP.Checked;
  ip.UseRandomDstIP   := RandomDstIP.Checked;
  ip.Data             := DataMemo.Text;

  ip.DelayBetweenPackets := StrToIntDef(DelayEdit.Text, 1000);
  DelayEdit.Text := IntToStr(ip.DelayBetweenPackets);

  ip.NumPackets := StrToIntDef(NumPacketsEdit.Text, 1);
  NumPacketsEdit.Text := IntToStr(ip.NumPackets);

  ip.TimeToLive := StrToIntDef(TtlEdit.Text, 1);
  TtlEdit.Text := IntToStr(ip.TimeToLive);

  if NOT ip.UseRandomSrcIP then
  begin
    try
      ip.SourceHost := SrcIpEdit.Text;
    except
      ShowMessage('Невідомий хост відправника: '+ SrcIpEdit.Text);
      Exit;
    end;
```

```
  end;

  if NOT ip.UseRandomDstIP then
  begin
    try
      ip.DestinationHost := DstIpEdit.Text;
    except
      ShowMessage('Невідомий хост одержувача: '+ DstIpEdit.Text);
      Exit;
    end;
  end;

  Result := TRUE;
end;

function TForm1.AcceptArtcpSettings(artcp: TSenderARTCP): Boolean;
// Перевірка установочних параметрів ARTCP заданих користувачем
VAR x: DWORD;
begin
  Result := AcceptIPSettings(artcp);
  if Result then
  begin
    artcp.UseRandomSrcPort := RandomSrcPort.Checked;
    artcp.UseRandomDstPort := RandomDstPort.Checked;

    artcp.FinFlag := FinFlag.Checked;
    artcp.SynFlag := SynFlag.Checked;
    artcp.RstFlag := RstFlag.Checked;
    artcp.PshFlag := PshFlag.Checked;
    artcp.AckFlag := AckFlag.Checked;
    artcp.UrgFlag := UrgFlag.Checked;

    if NOT RandomSrcPort.Checked then
    begin
      x := StrToIntDef(DstPortEdit.Text, 80);
      if x < 1 then
      begin
        x := 80;
        SrcPortEdit.Text := '80';
      end;
      artcp.SourcePort := x;
    end;


    if NOT RandomDstPort.Checked then
    begin
      x := StrToIntDef(DstPortEdit.Text, 80);
      if x < 1 then
      begin
        x := 80;
        DstPortEdit.Text := '80';
      end;
      artcp.DestinationPort := x;
    end;

    x := DWORD(StrToIntDef(SequenceEdit.Text, 12345678));
    if (x < 1) OR UseRandomSeq.Checked then
      x := DWORD(Random(MaxInt-1)+1);
    SequenceEdit.Text := IntToStr(x);
    artcp.SequenceNumber := x;
  end;
end;

function TForm1.AcceptUdpSettings(udp: TSenderUDP): Boolean;
// Перевірка установочних параметрів UDP заданих користувачем
begin
  Result := AcceptIPSettings(udp);
  if Result then
  begin
```

```
    udp.UseRandomSrcPort := RandomSrcPort.Checked;
    udp.UseRandomDstPort := RandomDstPort.Checked;

    if NOT RandomSrcPort.Checked then
      udp.SourcePort := StrToIntDef(DstPortEdit.Text, 0);

    if NOT RandomDstPort.Checked then
      udp.DestinationPort := StrToIntDef(DstPortEdit.Text, 0);
  end;
end;

function TForm1.AcceptPingSettings(ping: TSenderICMP): Boolean;
// Перевірка установочних параметрів ICMP/Ping заданих користувачем
begin
  Result := AcceptIPSettings(ping);
  if Result then
  begin
    ping.UseRandomPingID := RandomSrcPort.Checked;
    ping.UseRandomPingSequence := RandomDstPort.Checked;

    if NOT RandomSrcPort.Checked then
      ping.PingID := StrToIntDef(DstPortEdit.Text, $1234);
    if NOT RandomDstPort.Checked then
      ping.PingSequence := StrToIntDef(DstPortEdit.Text, $4321);
  end;
end;

function  TForm1.AcceptUserInput(Sender: TSenderIP; x: TProtocolType): Boolean;
begin
  ResultMemo.Clear;  { clear the results field }

  Sender.OnPrintLine := PrintLine; { progress feedback }

  // Перевірка установочних параметрів протоколів заданих користувачем
  // в залежності від обраного типу протоколу
  //
  case x of
    ptUDP:  Result := AcceptUdpSettings(TSenderUDP(Sender));
    ptICMP: Result := AcceptPingSettings(TSenderICMP(Sender));
    else    Result := AcceptArtcpSettings(TSenderARTCP(Sender));
  end
end;

procedure TForm1.RandomSrcIPClick(Sender: TObject);
begin
  SrcIpEdit.Enabled := NOT RandomSrcIP.Checked
end;

procedure TForm1.RandomDstIPClick(Sender: TObject);
begin
  DstIpEdit.Enabled := NOT RandomDstIP.Checked
end;

procedure TForm1.RandomSrcPortClick(Sender: TObject);
begin
  SrcPortEdit.Enabled := NOT RandomSrcPort.Checked;
  case FProtocolType of
    ptUDP:  FRandomSrcUDP := RandomSrcPort.Checked;
    ptICMP: FRandomIdPing := RandomSrcPort.Checked;
    else    FRandomSrcARTCP := RandomSrcPort.Checked;
  end;
end;

procedure TForm1.RandomDstPortClick(Sender: TObject);
begin
  DstPortEdit.Enabled := NOT RandomDstPort.Checked;
  case FProtocolType of
    ptUDP:  FRandomDstUDP := RandomDstPort.Checked;
    ptICMP: FRandomSqPing := RandomDstPort.Checked;
```

```
    else    FRandomDstARTCP := RandomDstPort.Checked;
  end;
end;

procedure TForm1.UseRandomSeqClick(Sender: TObject);
begin
  SequenceEdit.Enabled := NOT UseRandomSeq.Checked
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  FRandomSrcARTCP := TRUE;
  UseRandomSeq.Checked := TRUE;
  TabControl.OnChange(Sender); { init tab control w/ARTCP settings }
end;

procedure TForm1.GoButtonClick(Sender: TObject);
VAR x: TSenderIP;
begin


  // Заборона вводу користувача
  //
  TopPanel.Enabled := FALSE;
  MainPanel.Enabled := FALSE;
  Application.ProcessMessages;

  try

    WarnAboutW2k;      // для Win2K

    // створюємо специфічний обєкт користувача (ARTCP, UDP
    // або ICMP/Ping) в залежності від обраного протоколу


    case FProtocolType of
      ptUDP:  x := TSenderUDP.Create;
      ptICMP: x := TSenderICMP.Create;
      else begin
        FProtocolType := ptARTCP;
        x := TSenderARTCP.Create;
      end;
    end;
    x.OnExecute := AcceptUserInput;

    try

      // перевіряємо та приймаємо введення користувача
      // та відправляємо один або декілька пакетів у мережу.
      //
      x.Execute;
    finally
      x.Free;
    end
  finally
    MainPanel.Enabled := TRUE;
    TopPanel.Enabled := TRUE;
  end
end;

procedure TForm1.TabControlChange(Sender: TObject);
begin
  // якщо протокол змінюєжться, коректується заголовок
  //
  case TabControl.TabIndex of
    1:   { UDP }
      begin
        FProtocolType := ptUDP;
        GoButton.Caption := 'Генерація пакетів';
```

```
        SrcPortLabel.Caption  := 'Порт відправника: ';
        DstPortLabel.Caption  := 'Порт одержувача: ';
        RandomSrcPort.Checked := FRandomSrcUDP; { restore checkmarks }
        RandomDstPort.Checked := FRandomDstUDP;
        SeqNoLabel.Visible    := FALSE;
        SequenceEdit.Visible  := FALSE;
        UseRandomSeq.Visible  := FALSE;
        FlagsGroupBox.Visible := FALSE;
      end;
    2:   { ICMP/Ping }
      begin
        FProtocolType := ptICMP;
        GoButton.Caption := 'Генерація пакетів';
        SrcPortLabel.Caption  := 'ID: ';
        DstPortLabel.Caption  := 'Послідовність: ';
        RandomSrcPort.Checked := FRandomIdPing;
        RandomDstPort.Checked := FRandomSqPing;
        SeqNoLabel.Visible    := FALSE;
        SequenceEdit.Visible  := FALSE;
        UseRandomSeq.Visible  := FALSE;
        FlagsGroupBox.Visible := FALSE;
      end;
    else { ARTCP }
      begin
        FProtocolType := ptARTCP;
        GoButton.Caption := 'Генерація пакетів';
        SrcPortLabel.Caption  := 'Порт відправника: ';
        DstPortLabel.Caption  := 'Порт одержувача: ';
        RandomSrcPort.Checked := FRandomSrcARTCP;
        RandomDstPort.Checked := FRandomDstARTCP;
        SeqNoLabel.Visible    := TRUE;
        SequenceEdit.Visible  := TRUE;
        UseRandomSeq.Visible  := TRUE;
        FlagsGroupBox.Visible := TRUE;
      end;
  end
end;

end.
```

Файл USDDC_SND.PAS – відправка ARTCP-пакетів

```
unit USDDC_Snd;

interface

uses
  Windows, SysUtils, Forms, Winsock, ArtcpIpHlp;

type
  TSenderIP = class;
  TProtocolType = (ptARTCP, ptUDP, ptICMP);
  TOnExecuteEvent = function (Sender: TSenderIP; x: TProtocolType): Boolean of
Object;
  TOnPrintLineEvent = procedure (s: String) of Object;

  TSenderIP = class(TObject)  // IP клас пакета передавача
  private
    FProtocol: Word;
    FSocket: TSocket;

    FNumPackets: Word;
    FMaxPackets: Word;
    FSrcIP, FDstIP: DWORD;
    FSourceHost: String;
    FDestinationHost: String;
    FDelay: DWORD;

    FTimeToLive: Byte;

    FRandomSrcIP,
    FRandomDstIP: Boolean;

    FData: String;
    FDataLen: Integer;

    FOnExecute: TOnExecuteEvent;
    FOnPrintLine: TOnPrintLineEvent;

    function  GetLength: Word; virtual; abstract;
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
virtual; abstract;
    procedure UpdateSequence; virtual;

    procedure PrintLine(s: String);
    procedure ThrowException(msg: String; eCode: Integer);

    procedure SetNumPackets(value: Word);
    procedure SetSrcIP(value: String);
    procedure SetDstIP(value: String);
    procedure SetDelay(value: DWORD);
    procedure SetData(value: String);

    procedure SendPackets;
    procedure SendOnePacket;
    function  SendToRemote(addr, port: Integer;
                           buffer: PChar; len: Integer): Integer;

   {$ifdef DUMP_MODE}
    procedure Dump(data: PByte; len: Integer);
   {$endif}

  protected
    FRandomSrcPort,
    FRandomDstPort: Boolean;
    FSrcPort, FDstPort: Word;
    FIpBuffer: Array [1..$FFFF] of Char; // IP пакет не більше 64K
```

```
  public
    constructor Create; virtual;
    destructor Destroy; override;
    procedure Execute;

    property  NumPackets: Word read FNumPackets write SetNumPackets;
    property  SourceHost: String read FSourceHost write SetSrcIP;
    property  DestinationHost: String read FDestinationHost write SetDstIP;
    property  DelayBetweenPackets: DWORD read FDelay write SetDelay;
    property  TimeToLive: Byte read FTimeToLive write FTimeToLive;

    property  UseRandomSrcIP: Boolean read FRandomSrcIP write FRandomSrcIP;
    property  UseRandomDstIP: Boolean read FRandomDstIP write FRandomDstIP;

    property  Data: String read FData write SetData;

    property  OnExecute: TOnExecuteEvent read FOnExecute write FOnExecute;
    property  OnPrintLine: TOnPrintLineEvent read FOnPrintLine write
FOnPrintLine;
  end;

  TSenderARTCP = class(TSenderIP)  // ARTCP пакет передавача
  private
    FSequence: DWORD;
    FFinFlag, { ARTCP flags }
    FSynFlag,
    FRstFlag,
    FPshFlag,
    FAckFlag,
    FUrgFlag: Boolean;

    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
    procedure UpdateSequence; override;
  public
    constructor Create; override;

    property  SequenceNumber: DWORD read FSequence write FSequence;
    property  FinFlag: Boolean read FFinFlag write FFinFlag;
    property  SynFlag: Boolean read FSynFlag write FSynFlag;
    property  RstFlag: Boolean read FRstFlag write FRstFlag;
    property  PshFlag: Boolean read FPshFlag write FPshFlag;
    property  AckFlag: Boolean read FAckFlag write FAckFlag;
    property  UrgFlag: Boolean read FUrgFlag write FUrgFlag;

    property  SourcePort: Word read FSrcPort write FSrcPort;
    property  DestinationPort: Word read FDstPort write FDstPort;
    property  UseRandomSrcPort: Boolean read FRandomSrcPort write
FRandomSrcPort;
    property  UseRandomDstPort: Boolean read FRandomDstPort write
FRandomDstPort;
  end;

  TSenderUDP = class(TSenderIP)  // UDP пакет передавача
  private
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
  public
    constructor Create; override;

    property  SourcePort: Word read FSrcPort write FSrcPort;
    property  DestinationPort: Word read FDstPort write FDstPort;
    property  UseRandomSrcPort: Boolean read FRandomSrcPort write
FRandomSrcPort;
    property  UseRandomDstPort: Boolean read FRandomDstPort write
FRandomDstPort;
```

```
  end;

  TSenderICMP = class(TSenderIP)  // Ping Пакет передавача
  private
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
  public
    constructor Create; override;

    property  PingID: Word read FSrcPort write FSrcPort;
    property  PingSequence: Word read FDstPort write FDstPort;
    property  UseRandomPingID: Boolean read FRandomSrcPort write FRandomSrcPort;
    property  UseRandomPingSequence: Boolean read FRandomDstPort write
FRandomDstPort;
  end;

implementation

type
  // ARTCP/UDP псевдозаголовок
  // (використовується для підрахунку контрольної суми)
  //
  TPseudoHeader = packed record
    saddr, daddr: DWORD;
    zero, protocol: BYTE;
    packetlen: WORD;
  end;

procedure DelayMS(ms: Integer);
//
// DelayMS() - затримка в мілісекундах.
// можливо використання стандартної функції sleep(),
// але вона дуже довго працює й складається враження
// що додаток підвис.
// написане DelayMS() викликає Application.ProcessMessages.
//
var StartTime: LongInt;
begin
  StartTime := GetTickCount;
  repeat
    Application.ProcessMessages;
  until (LongInt(GetTickCount)-StartTime >= ms) or (Application.Terminated)
end;

procedure RaiseException(msg: String; eCode: Integer);
//
// форматуємо повідомлення для передачі по протоколу
//
  function AdditionalMessage: String;
  begin
    Result := SysErrorMessage(eCode);
    if Result <> '' then Result := ': ' + Result
  end;
begin
  if eCode = 0 then
    raise Exception.Create(msg)
  else
    raise Exception.Create('Помилка: '+msg+' [SocketError '+IntToStr(eCode)
                           +AdditionalMessage+']')
end;

{  TSenderIP }

constructor TSenderIP.Create;
begin
  FSocket := INVALID_SOCKET;
  Randomize  { використовуємо стандартний генератор випадкових чисел }
end;
```

```
destructor TSenderIP.Destroy;
begin
  CleanupWinsock(FSocket);
  inherited
end;

procedure TSenderIP.ThrowException(msg: String; eCode: Integer);
begin
  if eCode <> 0 then CleanupWinsock(FSocket);
  RaiseException(msg, eCode)
end;

procedure TSenderIP.Execute;
VAR opt: Integer;
    errStr: String;
    x: Boolean;
begin
  // ініціалізуємо WinSock. Працюємо з Winsock version 2.
  //
  errStr := InitWinsock(2,2);
  if errStr <> '' then  ThrowException(errStr, 0);

  if Assigned(FOnExecute) then
  begin
    case FProtocol of
       IPPROTO_UDP:   x := FOnExecute(Self, ptUDP);
       IPPROTO_ICMP:  x := FOnExecute(Self, ptICMP);
       else           x := FOnExecute(Self, ptARTCP);
    end;
    if NOT x then
    begin
      CleanupWinsock;
      Exit
    end
  end;

  FSocket := socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

  if (FSocket = INVALID_SOCKET) then
    ThrowException('Failed to open a RAW socket', WSAGetLastError());

  // З Winsock 2 IP_HDRINCL закриваємо сокети
  //
  // IP_HDRINCL працює на IPPROTO_IP рівні
  //
  opt := 1;
  if (setsockopt(FSocket, IPPROTO_IP, IP_HDRINCL, PChar(@opt), sizeof(opt)) =
SOCKET_ERROR) then
    ThrowException('Помилка встановленя опцій сокетів ', WSAGetLastError())
  else begin
    SendPackets; {... }
    CleanupWinsock(FSocket);
    PrintLine('Done.');
  end
end;

procedure TSenderIP.SetNumPackets(value: Word);
begin
  FNumPackets := value;
  if FNumPackets = 0 then  //
    FMaxPackets := MaxWord
  else
    FMaxPackets := FNumPackets
end;

procedure TSenderIP.SetData(value: String);
begin
  FData := value;
```

```
  FDataLen := Length(FData)
end;

procedure TSenderIP.SetDelay(value: DWORD);
begin
  // Встановлюємо мінімальну затримку 50 ms
  //
  if value < 50 then  value := 50;
  FDelay := value
end;

procedure TSenderIP.SetSrcIP(value: String);
begin
  FSrcIP := ResolveHostAddress(value);
  if FSrcIP = DWORD(-1) then
  begin
    FSourceHost := '';
    ThrowException('Невідомий хост відправника: '+ value, 0)
  end
  else begin
    FSourceHost := value;
    PrintLine('Невідомий хост відправника: '+value+'...');
  end;
end;

procedure TSenderIP.SetDstIP(value: String);
begin
  FDstIP := ResolveHostAddress(value);
  if FDstIP = DWORD(-1) then
  begin
    FDestinationHost := '';
    ThrowException('Невідомий хост одержувача: '+ value, 0)
  end
  else begin
    FDestinationHost := value;
    PrintLine('Невідомий хост одержувача: '+value+'...');
  end;
end;

procedure TSenderIP.SendPackets;
  function GenerateFakeIP: String;
  VAR a, b, c, d: Integer;
  begin
    a := random(239)+1; // a >= 240 викликає помилку сокета
    b := random(255);
    c := random(255);
    d := random(255);
    Result := Format('%d.%d.%d.%d', [a, b, c, d]);
  end;
VAR srcIP, dstIP: String;
    i: Integer;
begin
  for i := 1 to FMaxPackets do
  begin
    if FRandomSrcPort then { використовуємо вищі вихідні порти }
      FSrcPort := random(5000-1024) + 1024;

    if FRandomDstPort then { використовуємо порти }
      FDstPort := random($FFFF);

    srcIP := FSourceHost;
    dstIP := FDestinationHost;

    if FRandomSrcIP then
    begin
      srcIP := GenerateFakeIP;
      FSrcIP := ResolveHostAddress(srcIP);
    end;
```

```
    if FRandomDstIP then
    begin
      dstIP := GenerateFakeIP;
      FDstIP := ResolveHostAddress(dstIP);
    end;

    PrintLine('Відправка: '+ srcIP+ ':'+ IntToStr(FSrcPort)+ ' -> '
                          + dstIP+ ':'+ IntToStr(FDstPort));

    Application.ProcessMessages;
    SendOnePacket;
    if i < FMaxPackets then  DelayMS(FDelay);
  end;
end;

procedure TSenderIP.SendOnePacket;
VAR errCode: Integer;
    ih: THdrIP;
    id: WORD;
begin
  // Створюємо IP заголовок
  id := random($FFFF);

  FillChar(ih, sizeof(ih), 0);// очищуємо
  SetIHver(ih, 4);            // IP v 4
  SetIHlen(ih, sizeof(ih));   // IP довжина заголовка

  ih.tot_len  := htons(GetIHlen(ih) + GetLength); // загальна довжина
  ih.id       := htons(id);   // ID
  ih.ttl      := FTimeToLive; // час передачі
  ih.protocol := FProtocol;   // тип протоколу
  ih.saddr    := FSrcIP;      // джерело   IP
  ih.daddr    := FDstIP;      // розташування IP
  ih.check    := CalculateChecksum(@ih, GetIHlen(ih)); // підраховуємо IP
контрольну суму

  errCode     := SendDatagram(ih, PChar(FData), FDataLen);

  if errCode = 0 then
    UpdateSequence
  else
    ThrowException('Помилка відправки датаграми',  errCode)
end;

procedure TSenderIP.UpdateSequence;
begin
  { нічого не робить }
end;

function  TSenderIP.SendToRemote(addr, port: Integer;
                                 buffer: PChar;
                                 len: Integer): Integer;
VAR remote: TSockAddrIn;
begin
  Result := 0;

 {$ifdef DUMP_MODE}
  // dump() - допомогоа при дебазі...
  dump(buffer, len);
 {$endif}

  FillChar(remote, sizeof(remote), 0);
  remote.sin_family      := AF_INET;    { Internet домен }
  remote.sin_port        := port;
  remote.sin_addr.s_addr := addr;

  // відправляємо датаграму
  if (sendto(FSocket, buffer^, len, 0, TSockAddr(remote), sizeof(remote)) =
SOCKET_ERROR) then
```

```
    Result := WSAGetLastError()
end;

procedure TSenderIP.PrintLine(s: String);
begin
  if Assigned(FOnPrintLine) then FOnPrintLine(s)
end;

{$ifdef DUMP_MODE}
VAR howmany: Integer = 1;   // тільки дамп
procedure TSenderIP.Dump(data: PByte; len: Integer);
VAR i: Integer;
    s: String;
begin
  if howmany = 0 then Exit;
  Dec(howmany);

  PrintLine('');
  PrintLine('Дамп даних: загальна довжина: '+IntToStr(len));

  s := '';
  for i := 0 to len-1 do
  begin
    s := s + Format('%02X ', [data^]);
    if ((i+1) mod 16 = 0) then  PrintLine(s);
    Inc(data);
  end;
  PrintLine(s);
  PrintLine('');
end;
{$endif} {DUMP_MODE}


{  TSenderARTCP }

constructor TSenderARTCP.Create;
begin
  inherited;
  FProtocol := IPPROTO_ARTCP
end;

function TSenderARTCP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR ph: TPseudoHeader;
    th: THdrARTCP;
    len: Integer;
    p: PChar;
begin
  // створення ARTCP заголовку
  FillChar(th, sizeof(th), 0);    // очищення
  th.source       := htons(FSrcPort); // порт джерела
  th.dest     := htons(FDstPort); // порт розташування
  th.seq      := htonl(FSequence);// номер послідовності
  th.ack_seq  := htonl(0);        // номер запиту

  SetTHdoff(th, sizeof(th));      // прапорці $0050
  SetTHflag(th, ftFIN, FFinFlag);
  SetTHflag(th, ftSYN, FSynFlag); // прапорці $0250
  SetTHflag(th, ftRST, FRstFlag);
  SetTHflag(th, ftPSH, FPshFlag);
  SetTHflag(th, ftACK, FAckFlag);
  SetTHflag(th, ftURG, FUrgFlag);
  th.window   := htons($FFFF);

  // розрахування ARTCP контрольну суму  ...
  { ARTCP контрольна сума   включає в себе 96 біт псевдо заголовку
    Прописані в ARTCP заголовку.  Цей псевдо заголовок
    Включає Адресу Джерела, Адресу розташування, Протокол (займає октет), та
довжину ARTCP пакету. Використовується для захисту ARTCP від втрати сегментів.
```

```
  }
  ph.saddr     := ih.saddr;
  ph.daddr     := ih.daddr;
  ph.zero      := 0;
  ph.protocol  := FProtocol;
  ph.packetlen := htons(sizeof(th) + dlen);

  p := @FIpBuffer;
  Move(ph, p^, sizeof(ph));
  Inc(p, sizeof(ph));
  Move(th, p^, sizeof(th));
  Inc(p, sizeof(th));
  Move(data^, p^, dlen);

  th.check := CalculateChecksum(@FIpBuffer, sizeof(ph) + sizeof(th) + dlen);

  // заповнюється буфер посилання
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка в байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(th, p^, sizeof(th));
  Inc(p, sizeof(th));
  Move(data^, p^, dlen);
  len := len + sizeof(th)+ dlen;

  Result := SendToRemote(ih.daddr, th.dest, @FIpBuffer, len);
end;

function  TSenderARTCP.GetLength: Word;
begin
  Result := sizeof(THdrARTCP) + FDataLen
end;

procedure TSenderARTCP.UpdateSequence;
begin
  { номер послідовності задається октетом даних в датаграмі }
  Inc(FSequence, FDataLen)
end;


{  TSenderUDP }

constructor TSenderUDP.Create;
begin
  inherited;
  FProtocol := IPPROTO_UDP
end;

function  TSenderUDP.GetLength: Word;
begin
  Result := sizeof(THdrUDP) + FDataLen
end;

function TSenderUDP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR ph: TPseudoHeader;
    uh: THdrUDP;
    len: Integer;
    p: PChar;
begin
  // створюємо заголовок UDP
  FillChar(uh, sizeof(uh), 0);    // очищуємо
  uh.src_port := htons(FSrcPort); // порт джерела
  uh.dst_port := htons(FDstPort); // порт розташування
  uh.length   := htons(sizeof(uh)+dlen);

  // розраховуємо UDP контрольну суму  ...
```

```
  { UDP контрольна сума   включається в  96 бітовий псевдо заголовок який
прописаний в UDP заголовку. Розписано в  RFC 768
    for details
  }
  ph.saddr     := ih.saddr;
  ph.daddr     := ih.daddr;
  ph.zero      := 0;
  ph.protocol  := IPPROTO_UDP;
  ph.packetlen := htons(sizeof(uh)+dlen);

  p := @FIpBuffer;
  Move(ph, p^, sizeof(ph));
  Inc(p, sizeof(ph));
  Move(uh, p^, sizeof(uh));
  Inc(p, sizeof(uh));
  Move(data^, p^, dlen);

  uh.checksum   := CalculateChecksum(@FIpBuffer, sizeof(ph) + sizeof(uh) +
dlen);
  if uh.checksum   = 0 then  uh.checksum   := $FFFF;

  // заповнення буферу відправки
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка у байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(uh, p^, sizeof(uh));
  Inc(p, sizeof(uh));
  Move(data^, p^, dlen);
  len := len + sizeof(uh)+ dlen;

  Result := SendToRemote(ih.daddr, uh.dst_port, @FIpBuffer, len);
end;


{  TSenderICMP  }

type
  THdrECHO = packed record  // ICMP exo (Ping) заголовок
    IcmpType: Byte;
    IcmpCode: Byte;
    контрольну суму  : WORD;
    id      : WORD;
    sequence: WORD;
  end;

constructor TSenderICMP.Create;
begin
  inherited;
  FProtocol := IPPROTO_ICMP
end;

function  TSenderICMP.GetLength: Word;
begin
  Result := sizeof(THdrEcho) + FDataLen
end;

function TSenderICMP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR eh: THdrEcho;
    len: Integer;
    p: PChar;
begin
  // створює ICMP/Ping заголовок
  FillChar(eh, sizeof(eh), 0);    // очищення
  eh.IcmpType := 8;               // ICMP тип ECHO
  eh.IcmpCode := 0;
  eh.ID       := htons(FSrcPort);
  eh.Sequence := htons(FDstPort);
```

```
  // розраховуємо ICMP контрольну суму  ...
  p := @FIpBuffer;
  Move(eh, p^, sizeof(eh));
  Inc(p, sizeof(eh));
  Move(data^, p^, dlen);

  eh.checksum   := CalculateChecksum(@FIpBuffer, sizeof(eh) + dlen);

  // заповнення буферу відправки
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка у байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(eh, p^, sizeof(eh));
  Inc(p, sizeof(eh));
  Move(data^, p^, dlen);
  len := len + sizeof(eh)+ dlen;

  Result := SendToRemote(ih.daddr, eh.id, @FIpBuffer, len);
end;

procedure TForm1.N5Click(Sender: TObject);
begin
Form2.Show;
end;

end.
```

Файл ARTCPIPHLP.PAS – функції роботи зі стеком протоколів ARTCP/IP

```
unit ArtcpIpHlp; { ARTCP/IP хелпер }

interface

uses Windows, Winsock, SysUtils;

CONST SIO_RCVALL = $98000001;

type
  THdrIP = packed record   // IP заголовок (RFC 791)
    ihl_ver : BYTE;        // комбінована область:
                           //   ihl:4 - IP довжина заголовка divided by 4
                           //   version:4 - IP версія
    tos     : BYTE;        // IP type-of-service поле
    tot_len : WORD;        // загальна довжина
    id      : WORD;        // унікальний ID
    frag_off: WORD;        // Fragment Offset + fragmentation flags (3 bits)
    ttl     : BYTE;        // час передачі
    protocol: BYTE;        // тип протоколу
    check   : WORD;        // IP заголовок контрольна сума
    saddr   : DWORD;       // джерело   IP
    daddr   : DWORD;       // розташування IP
   {початкові опції...}
  end;
  PHdrIP = ^THdrIP;

  (* Розписуємо IP заголовок (RFC 791):

    -ih.ihl довжина заголовка у байтах поділена на 4
     Internet Довжина заголовка –довжина інтернет заголовка
     в 32 бітових словах, мінімальна довжина для правильного заголовка 5.

    -ih.tos - IP type-of-service область забезпечення якості бажаємої послуги.

    -ih.id  - визначається передавачем, для допомоги при трансляції фрагментів
датаграми.

    -ih.frag_off включає 3 бітовий флаг (RFC815).
       Bit 0: зарезервовано, дорівнює нулю
       Bit 1: (DF) 0 = може фрагментуватися,  1 = не може.
       Bit 2: (MF) 0 = останій фрагмент, 1 = поточний фрагмент.
       Bits?: показують розташування цього фрагмента в датаграмі

    -ih.protocol повідомляє IP при посилці датаграми
   *)

  THdrARTCP = packed record     // ARTCP заголовок
    джерело   : WORD;  // порт джерела
    dest    : WORD;  // порт розташування
    seq     : DWORD; // номер послідовності
    ack_seq: DWORD; // наступний номер послідовності
    flags   : WORD;  // комбіноване поле:
                     //   res1:4 – зарезервовано.дорівнює 0
                     //   doff:4 - ARTCP довжина заголовка поділена на 4
                     //   fin:1  - FIN
                     //   syn:1  - SYN
                     //   rst:1  - Reset
                     //   psh:1  - Push
                     //   ack:1  - ACK
                     //   urg:1  - Urgent
                     //   res2:2 – зарезервовано. дорівнює 0
    window : WORD;  // розмір вікна
    check  : WORD;  // контрольна сума
    urg_ptr: WORD;  // використовується для повідомлень
  end;
```

```
  PHdrARTCP = ^THdrARTCP;
  (* Деталі ARTCP заголовку

    -th.seq - номер послідовності – перший октет у сегменті

    -th.doff – зміщення даних - число 32 бітових слов в заголовку ARTCP.


    -th.ack_seq використовується, коли ACK флаг встановлений


    -th.window використовується для управління даними при передачі у любий
проміжок часу

    -th.urgent поле у яке записується конкретний октет, який буде оброблятися.
*)

  THdrUDP = packed record  // UDP заголовок (RFC 768)
    src_port: WORD;        // порт джерела
    dst_port: WORD;        // порт розташування
    length  : WORD;        // довжина, включаючи заголовок
    контрольну суму  : WORD;        // UDP контрольна сума
  end;
  PHdrUDP = ^THdrUDP;

CONST
 { Опції використовують[gs]etsockopt для рівня IPPROTO_IP }

 IP_OPTIONS                =  1; { встановити/прочитати IP опції }
 IP_HDRINCL                =  2; { заголовок, включаючий дані }
 IP_TOS                    =  3; { IP тип послуги}
 IP_TTL                    =  4; { IP час передачі  }
 IP_MULTICAST_IF           =  9; { встановити/прочитати IP груповий інтерфейс}
 IP_MULTICAST_TTL          = 10; { встановити/прочитати IP груповий  ttl }
 IP_MULTICAST_LOOP         = 11; { встановити/прочитати IP груповий  loopback }
 IP_ADD_MEMBERSHIP         = 12; { додати IP групу }
 IP_DROP_MEMBERSHIP        = 13; { частина IP групи }
 IP_DONTFRAGMENT           = 14; { не фрагмент IP датаграми }
 IP_ADD_SOURCE_MEMBERSHIP  = 15; { приєднатися до IP грпи/джерела  }
 IP_DROP_SOURCE_MEMBERSHIP = 16; { залишити IP групу/джерело   }
 IP_BLOCK_SOURCE           = 17; { блокувати IP групу/джерело   }
 IP_UNBLOCK_SOURCE         = 18; { розблокувати IP групу/джерело }
 IP_SDDCINFO                = 19; { приймає інформацію пакету для ipv4}

 { мережні інтерфейсні типи}
 IFF_UP            = $00000001; { інтерфейс запуску }
 IFF_BROADCAST     = $00000002; { підтримання широковіщання}
 IFF_LOOPBACK      = $00000004; { інтерфейс loopback }
 IFF_POINTTOPOINT  = $00000008; { інтерфейс включаючий point-to-point link}
 IFF_MULTICAST     = $00000010; { групова  характеристика підтримується }

type
  TArtcpFlagType = (ftFIN, ftSYN, ftRST, ftPSH, ftACK, ftURG);
  TEnumInterfacesEvent = procedure (value: String; iff_type: Integer) of Object;


// присвоюємо імені число
//
function  GetIPProtoName(protocol: Byte): String;
function  GetServiceName(s_port, d_port: Integer): String;
function  GetICMPType(x: Byte): String;

// вказує загальний шлях
//
procedure CleanupWinsock; overload;
procedure CleanupWinsock(VAR socket: TSocket); overload;
function  Win2KDetected: Boolean;
procedure EnumInterfaces(cb: TEnumInterfacesEvent; iff_types: Integer);
function  InitWinsock(hi_ver, lo_ver: Byte): String;
```

```
function  ResolveHostAddress(name: String): u_long;

//
procedure SetTHdoff(VAR th: THdrARTCP; value: Byte);
function  GetTHdoff(th: THdrARTCP): Word;
procedure SetTHflag(VAR th: THdrARTCP; flag: TArtcpFlagType; on: Boolean);
function  GetTHflag(th: THdrARTCP; flag: TArtcpFlagType): Boolean;
procedure SetIHver(VAR ih: THdrIP; value: Byte);
function  GetIHver(ih: THdrIP): Byte;
procedure SetIHlen(VAR ih: THdrIP; value: Byte);
function  GetIHlen(ih: THdrIP): Word;

// контрольна сума, яка використовується в IP заголовку, ARTCP, UDP, й т.д.
//
function  CalculateChecksum(addr: PWord; len: Integer): Word;

// Winsock 2 функції для імпорту
//
function getsockopt(s: TSocket; level, optname: Integer; optval: PChar; var
optlen: Integer): Integer; stdcall;
function setsockopt(s: TSocket; level, optname: Integer; optval: PChar; optlen:
Integer): Integer; stdcall;

function WSAIoctl(s: TSocket;
  dwIoControlCode: DWORD;
  lpvInBuffer: Pointer;
  cbInBuffer: DWORD;
  lpvOutBuffer: Pointer;
  cbOutBuffer: DWORD;
  lpcbBytesReturned: LPDWORD;
  lpOverlapped: Pointer;
  lpCompletionRoutine: Pointer): Integer; stdcall;

implementation

function getsockopt;   far; external 'ws2_32.dll' name 'getsockopt';
function setsockopt;   far; external 'ws2_32.dll' name 'setsockopt';
function WSAIoctl;     far; external 'ws2_32.dll' name 'WSAIoctl';

type
  TIPProto = record
    iType: word;
    iName: String;
  end;

  TWellKnownSvc = record
    port: Integer;
    svc: string[20];
  end;

CONST
  // Тип протоколу  s
  //
  IpProto: Array[1..5] Of TIPProto = (
    (iType: IPPROTO_IP;   iName: 'IP'),
    (iType: IPPROTO_ICMP; iName: 'ICMP'),
    (iType: IPPROTO_IGMP; iName: 'IGMP'),
    (iType: IPPROTO_ARTCP;  iName: 'ARTCP'),
    (iType: IPPROTO_UDP;  iName: 'UDP')
  );

  // сервіси – яким портам. Які сервіси та протоколи відповідають
  //
  WellKnownSvcs: array[1..43] of TWellKnownSvc = (
    ( port:   0; svc: 'LOOPBACK'),
    ( port:   1; svc: 'ARTCPMUX  '),   { ARTCP сервіс  порту }
    ( port:   7; svc: 'ECHO    ' ),  { exo                        }
    ( port:   9; svc: 'DISCARD ' ),  { Заборона                     }
    ( port:  13; svc: 'DAYTIME ' ),  { День та час               }
```

```
    ( port:  17; svc: 'QOTD    ' ),  { }
    ( port:  19; svc: 'CHARGEN ' ),  { генерація символів       }
    ( port:  20; svc: 'FTP_DATA' ),  { протокол Ftp                              }
    ( port:  21; svc: 'FTP_CTL ' ),  { протокол File Transfer Control Protocol}
    ( port:  22; svc: 'SSH     ' ),  { протокол SSH Remote Login Protocol     }
    ( port:  23; svc: 'TELNET  ' ),  { протокол TelNet                         }
    ( port:  25; svc: 'SMTP    ' ),  { протокол Simple Mail Transfer Protocol }
    ( port:  37; svc: 'TIME    ' ),
    ( port:  42; svc: 'NAME    ' ),  { Host Name Server              }
    ( port:  43; svc: 'WHOIS   ' ),  { WHO IS сервіс                }
    ( port:  53; svc: 'DNS     ' ),  { Domain Name Service           }
    ( port:  66; svc: 'SQL*NET ' ),  { Oracle SQL*NET                 }
    ( port:  67; svc: 'BOOTPS  ' ),  { BOOTP Server                   }
    ( port:  68; svc: 'BOOTPC  ' ),  { BOOTP Client                   }
    ( port:  69; svc: 'TFTP    ' ),  { протокол Trivial FTP                     }
    ( port:  70; svc: 'GOPHER  ' ),  { Gopher                         }
    ( port:  79; svc: 'FINGER  ' ),  { Finger                         }
    ( port:  80; svc: 'HTTP    ' ),  { протокол HTTP                            }
    ( port:  88; svc: 'KERBEROS' ),  { протокол Kerberos                        }
    ( port:  92; svc: 'NPP     ' ),  { протокол Network Printing Protocol     }
    ( port:  93; svc: 'DCP     ' ),  { протокол Device Control Protocol       }
    ( port: 109; svc: 'POP2    ' ),  { Post Office Protocol Version 2}
    ( port: 110; svc: 'POP3    ' ),  { протокол Post Office Protocol Version 3}
    ( port: 111; svc: 'SUNRPC  ' ),  { протокол SUN Remote Procedure Call     }
    ( port: 119; svc: 'NNTP    ' ),  { протокол Network News Transfer Protocol}
    ( port: 123; svc: 'NTP     ' ),  { протокол Network Time protocol         }
    ( port: 135; svc: 'LOCSVC  ' ),  { Location Service               }
    ( port: 137; svc: 'NTBNAME ' ),  { NETBIOS Name service           }
    ( port: 138; svc: 'NTBDGRAM' ),  { NETBIOS Datagram Service       }
    ( port: 139; svc: 'NTBSESSN' ),  { NETBIOS Session Service        }
    ( port: 161; svc: 'SNMP    ' ),  { протокол Simple Netw. Mgmt Protocol    }
    ( port: 162; svc: 'SNMPTRAP' ),  { протокол SNMP TRAP                     }
    ( port: 220; svc: 'IMAP3   ' ),  { протокол Interactive Mail Access Protocol
v3 }
    ( port: 443; svc: 'HTTPS   ' ),  { протокол HTTPS                         }
    ( port: 445; svc: 'MS-DS   ' ),  { Microsoft-DS                   }
    ( port:1433; svc: 'MSSQL   ' ),  { MSSQL                          }
    ( port:3306; svc: 'MYSQL   ' ),  { MySQL                          }
    ( port:5900; svc: 'VNC     ' )   { VNC - similar to PC Anywhere  }
  );

function GetIPProtoName(protocol: Byte): String;
VAR i: Integer;
begin
  Result := IntToStr(protocol);
  for i := 1 To sizeof(IPPROTO) div sizeof(TIPProto) do
    if protocol = IPPROTO[i].itype then
      Result := IPPROTO[i].iName;
end;

function GetServiceName(s_port, d_port: Integer): String;
VAR i: Integer;
begin
  Result := '';
  for i := 1 to sizeof(WellKnownSvcs) div sizeof(TWellKnownSvc) do
    if (s_port = WellKnownSvcs[i].port) OR (d_port = WellKnownSvcs[i].port) then
    begin
      Result := WellKnownSvcs[i].svc;
      break;
    end;
end;

function  GetICMPType(x: Byte): String;
begin
  Result := 'UNKNOWN';
  case x of
     0: Result := 'ECHO_R'; // повтор exa
     3: Result := 'DSTUNR'; // Розташування досягаємого
     4: Result := 'SRC_Q';  // блокування джерела повідомлень
```

```
     5: Result := 'REDIR';  // перепосилання
     8: Result := 'ECHO';   // exo
    11: Result := 'TTLX';   // перевищення часу
    12: Result := 'BADPAR'; // параметр помилки
    13: Result := 'TIME';   // часова мітка
    14: Result := 'TIME_R'; // повтор часової мітки
    15: Result := 'INFO';   // запит інформації
    16: Result := 'INFO_R'; // повтор інформації
  end
end;

function Win2KDetected: Boolean;
VAR IsNT: Boolean;
    ver: DWORD;
begin
  ver := GetVersion();

  IsNT  := ver < $80000000;
  Result := IsNT AND ((ver AND $FF) >= 5)
end;

function InitWinsock(hi_ver, lo_ver: Byte): String;
VAR versionRequested: Word;
    errorStatus: Integer;
    wd: TWSADATA;
begin
  Result := '';

  versionRequested := MAKEWORD(lo_ver, hi_ver);

  errorStatus := WSAStartup(versionRequested, wd);
  if (errorStatus <> 0) then
  begin
    Result := ' Помилка ініціалізації Winsock '+ IntToStr(errorStatus);
    Exit;
  end;

  if (LOBYTE(wd.wVersion) <> LOBYTE(versionRequested)) OR
     (HIBYTE(wd.wVersion) <> HIBYTE(versionRequested)) then
  begin
    Result := Format('Неспівпадіння версії Winsock (required: %d.%d, found:
%d.%d)',
                     [HIBYTE(versionRequested),
                     LOBYTE(versionRequested),
                     HIBYTE(wd.wVersion),
                     LOBYTE(wd.wVersion)]);
    Exit;
  end;
end;

procedure CleanupWinsock(VAR socket: TSocket);
begin
  if (socket <> 0) AND (socket <> INVALID_SOCKET) then
    CloseSocket(socket);
  socket := INVALID_SOCKET;
  WSACleanup()
end;

procedure CleanupWinsock;
begin
  WSACleanup()
end;

procedure EnumInterfaces(cb: TEnumInterfacesEvent; iff_types: Integer);
type
  TSockAddrGen = packed record
    AddressIn: sockaddr_in;
    { This record must be big enough
      to hold struct sockaddr_in6,
```

```
      which is 24 bytes long... }
    dummy: array[0..7] of char;
  end;

  TInterface_Info = packed record { see Q181520 }
    iiFlags: u_long;         // інтерфейс флагів
    iiAddress,               // інтерфейс адрес
    iiBroadcastAddress,      // передача адреси
    iiNetmask: TSockAddrGen; // маска мережі
  end;

CONST SIO_GET_INTERFACE_LIST = $4004747F;
VAR
  InterfaceList: array[0..$20] of TInterface_Info;
  NumInterfaces: Integer;
  BytesReturned: DWORD;
  AddrIn: TSockAddrIn;
  pAddr: PChar;
  flags: u_long;
  s: TSocket;
  i: Integer;
begin
  if InitWinsock(2,2) <> '' then Exit;

  s := Socket(AF_INET, SOCK_STREAM, IPPROTO_IP); { artcp сокет }
  if (s <> INVALID_SOCKET) then
  begin
    try
      if WSAIoctl(s,
                  SIO_GET_INTERFACE_LIST,
                  Nil,
                  0,
                  @InterfaceList,
                  sizeof(InterfaceList),
                  @BytesReturned,
                  Nil,
                  Nil) <> SOCKET_ERROR then
      begin
        NumInterfaces := BytesReturned div SizeOf(TInterface_Info);

        for i := 0 to NumInterfaces - 1 do
        begin
          AddrIn := InterfaceList[i].iiAddress.AddressIn;
          pAddr  := inet_ntoa(AddrIn.sin_addr);

          flags := InterfaceList[i].iiFlags;

          if (flags AND iff_types) = flags then
            if Assigned(cb) then  cb(pAddr, flags)
        end;
      end;
    except
      { do nothing }
    end;
  end;

  CleanupWinsock(s);
end;

function ResolveHostAddress(name: String): u_long;
//
// Ця функція записує заголовок імені хосту або IP адреси
//
VAR hep: PHostEnt;
    addr: u_long;
begin
  addr := inet_addr(PChar(name));

  if (addr = u_long(-1)) then
```

```
  begin
    hep := gethostbyname(PChar(name));
    if (hep <> Nil) then
    begin
      with TInAddr(addr), hep^ do
      begin
        S_un_b.s_b1 := h_addr^[0];
        S_un_b.s_b2 := h_addr^[1];
        S_un_b.s_b3 := h_addr^[2];
        S_un_b.s_b4 := h_addr^[3];
      end
    end
  end;

  Result := addr;
end;


function GetIHlen(ih: THdrIP): Word;  // IP довжина заголовка
begin

  // повертає довжину у байтах
  Result := (ih.ihl_ver AND $0F) SHL 2
end;

procedure SetIHlen(VAR ih: THdrIP; value: Byte);
begin
  // ділить значення на 4
  value := value SHR 2;
  ih.ihl_ver := value OR (ih.ihl_ver AND $F0)
end;

function GetIHver(ih: THdrIP): Byte;  // IP версія
begin
  // записує вищу частину
  Result := ih.ihl_ver SHR 4
end;

procedure SetIHver(VAR ih: THdrIP; value: Byte);
begin
  // встановлює вищу частину
  ih.ihl_ver := (value SHL 4) OR (ih.ihl_ver AND $0F)
end;

(* ARTCP заголовок містить наступні флаги
       res1:4 - зарезервовано.дорівнює 0
       doff:4 - ARTCP довжина заголовка ділиться на 4
       fin:1  - FIN
       syn:1  - SYN
       rst:1  - Reset
       psh:1  - Push
       ack:1  - ACK
       urg:1  - Urgent
       res2:2 - зарезервовано,дорівнює 0
     MSB
*)

CONST flagMask: Array[ftFIN..ftURG] of Integer = ($100, $200, $400, $800, $1000,
$2000);

function GetTHflag(th: THdrARTCP; flag: TArtcpFlagType): Boolean;
begin
  Result := Boolean(th.flags AND flagMask[flag])
end;

procedure SetTHflag(VAR th: THdrARTCP; flag: TArtcpFlagType; on: Boolean);
begin
  if on then
    th.flags := th.flags OR flagMask[flag]
```

```
  else
    th.flags := th.flags AND NOT flagMask[flag]
end;

function GetTHdoff(th: THdrARTCP): Word;
begin

  Result := (($00F0 AND th.flags) SHR 4) SHL 2;
end;

procedure SetTHdoff(VAR th: THdrARTCP; value: Byte);
VAR x: Integer;
begin
  x := value SHR 2; //
  th.flags := (x SHL 4) OR (th.flags AND $FF0F)
end;

function CalculateChecksum(addr: PWord; len: Integer): Word;
// алгоритм отримання  контрольну суму    RFCs 791,793
//
VAR sum: DWORD;
begin
  Result := 0;
  sum := 0;

  while (len > 1) do
  begin
    Inc(sum, addr^);
    Inc(addr);
    Dec(len, 2);
  end;


  if (len = 1) then
  begin
    PByte(@Result)^ := PByte(addr)^;
    Inc(sum, Result);
  end;

  sum := (sum SHR 16) + (sum AND $FFFF);{ }
  Inc(sum, sum SHR 16);                 { }
  sum := NOT sum;                       { }
  Result := Word(sum);                  { }
end;

end.
```

Файл About.PAS – довідка про програму

```
unit About;

interface

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

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

var  Form2: TForm2;

implementation
     uses Mainunit;
{$R *.dfm}

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

procedure TForm2.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи макровіртуалізації SDDC');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Цесько Сергій Романович');
Memo1.Lines.Add('                    гр. КІ-21-3СК');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Буравченко К.О.');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2024');
end;
end.
```