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
“____” _____ 2021 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-20М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Тарасенко Б.О.
«____» _____ 2021 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Володимир ПЕТРЕНЮК
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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
« 6 » вересня 2021 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Тарасенку Богдану Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6*

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

затверджені наказом вищого навчального закладу № 42-13 від 02.08.2021 року

3. Строк подання студентом роботи до захисту *10.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6*

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

1. Призначення та область використання.	*7. Економічна ефективність розробленої*
2.Перегляд аналогічних існуючих систем.	*програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	
6. Наукова новизна	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2021	14.11.2021
Охорона праці	Оришака О.В.	06.10.2021	16.11.2021

7. Дата видачі завдання « 6 » вересня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	10.12.2021 р.	

Дата видачі завдання
« 6 » вересня 2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Тарасенко Б.О. Дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи зберігання даних з використанням технології Fibre Channel 6.

Метою розробки є дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

Об'єктом дослідження є процес зберігання даних з використанням технології Fibre Channel 6.

Предметом дослідження є методи зберігання даних з використанням технології Fibre Channel 6.

Методи дослідження базуються на методах теорії телекомунікації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

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 архітектури IBM PC з ОС Windows XP/Vista/7/8/10.

Програму розроблено в середовищі Delphi 10.3.2 Rio Architect.

Ключові слова: комп'ютерна інженерія, Fibre Channel 6

ABSTRACT

Tarasenko B.O. Research and software implementation of a storage system using Fiber Channel 6 technology. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this final qualifying work for the second (master's) level of higher education, software has been developed that is designed for a data storage system using Fiber Channel 6 technology.

The purpose of development is research and software implementation of the storage system using Fiber Channel 6 technology.

The object of research is the process of data storage using Fiber Channel 6 technology.

The subject of the study are methods of data storage using Fiber Channel 6 technology.

Research methods are based on the methods of telecommunication theory, methods of mathematical statistics, methods of software development.

The result is a software implementation of a storage system using Fiber Channel 6 technology.

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

Developed user-friendly interface. Instructions for working with software are given.

The program can be used on an IBM PC with Windows XP / Vista / 7/8/10.

The program is developed in the environment Delphi 10.3.2 Rio Architect.

Keywords: computer engineering, Fiber Channel 6

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ЕОМ	–	електрона обчислювальна машина
КВ	–	коефіцієнт варіації
КЗ	–	канал зв'язку
НСД	–	несанкціонований доступ
ПС	–	програмна середа
СВВ	–	система виявлення вторгнень
СеМО	–	експонентна мережа масового обслуговування
СМО	–	система масового обслуговування
СПД	–	система передачі даних
DDoS	–	розподілена відмова від обслуговування
SAN	–	центр обробки даних
SDN	–	програмно-визначаєма мережа
VCS	–	Virtual Cluster Switching

ВСТУП

Актуальність теми. Fibre Channel залишається кращою технологією для підключення систем зберігання. Згідно надаваним IDC даним за 2019 рік за допомогою FC здійснюється доступ до більш ніж 20 тис. Пбайт даних – більше ніж за допомогою який-небудь іншої. На ринку комутаторів FC залишилося фактично два гравці – Brocade і Cisco, причому, на відміну від мережевого ринку, Cisco аж ніяк не домінуючий гравець у цьому сегменті – Brocade належить левова частина ринку (понад 80% як по обсязі продажів, так і по кількості портів, що поставляються,).

Рік назад Brocade увійшла до складу Broadcom як підрозділ Brocade Storage Networking.Всі лінійки IP- і Ethernet-устаткування були розпродані, так що тепер Brocade, як і колись, спеціалізується винятково на рішеннях Fibre Channel. В Україні все встаткування Brocade продається через OEM-партнерів – у компанії дотепер немає жодного дистриб'ютора або інтегратора, що був би авторизований на прямий продаж комутаторів під маркою Brocade.

Після різкого падіння продажів в 2015 році ринок в Україні нарешті відновився. Цього року ріст продажів триває, причому це досягається не тільки за рахунок окремих великих проектів, але й завдяки множині дрібних і середніх.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

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

– Огляд існуючих систем зберігання даних з використанням технології Fibre Channel 6.

– Дослідження системи зберігання даних з використанням технології Fibre Channel 6.

– Програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

Об'єктом дослідження є процес зберігання даних з використанням технології Fibre Channel 6.

Предметом дослідження є методи зберігання даних з використанням технології Fibre Channel 6.

Методи дослідження базуються на методах теорії телекомунікації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод зберігання даних з використанням технології Fibre Channel 6.

– Розроблено вітчизняний продукт зберігання даних з використанням технології Fibre Channel 6, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі зберігання даних з використанням технології Fibre Channel 6.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LV Науково-технічна конференція здобувачів вищої освіти «Наука – виробництву», 2021, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №12.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

За аналогією з мережевою моделлю OSI, Fibre Channel складається з п'яти рівнів. Кожний рівень забезпечує певний набір функцій.

FC-0 – рівень фізичних інтерфейсів і носіїв. Описує фізичне середовище: кабелі, конектори, HBA, трансівери, електричні й оптичні параметри.

FC-1 – рівень передачі й кодування. Тут описуються як дані будуть кодуватися перед передачею й декодуватися після. На цьому рівні визначаються три основні функції:

- Кодування / декодування.
- Ordered sets.
- Ініціалізація з'єднання (link initialization).

FC-2 – рівень кадрування й сигналів. Визначає структуру й організацію переданої інформації, а також контроль і керування її передачею. Функції, здійснювані на цьому рівні:

- Кадрування (визначення структури кадру / фрейму).
- Керування послідовностями (Sequence management).
- Керування обміном (Exchange management).
- Клас обслуговування (Class of Service).
- Керування потоком (Flow control).

FC-3 – рівень базових служб. Рівень закладений, для нових функцій, які можуть бути впроваджені в Fibre Channel у майбутньому. На цьому рівні забезпечується шифрування й стиск даних перед відправленням, а також такі речі як розщеплення потоку даних (striping) по декількох шляхах.

FC-4 – рівень відображення протоколів. Описує протоколи, які можуть використовувати FC як транспорт і, властиво, порядок використання (маппинг цих протоколів на нижні рівні FC 0-3). Стосовно до SAN цього можуть бути:

– Fibre Channel Protocol for SCSI-3 (SCSI-FCP) – проброс SCSI.

– Fibre Channel Link Encapsulation (FC-LE) – проброс TCP/IP.

А тепер докладніше про ці й інші незрозумілі словосполучення. У даному розділі розглянемо тільки нижні три рівні, як найбільш значимі при створенні й керуванні інфраструктурою FC SAN.

FC-0

Я, мабуть не буду приводити складних таблиць різновидів кабелів, передавачів і їхніх характеристик. По-перше, тому що незручно отут вставляти таблиці, по-друге, тому що ці таблиці є скрізь, де хоч щось написане про FC, по - третє (і ключове), – на мій погляд, головне зрозуміти суть, а довідкові дані знайти не проблема.

А суть у тому, що є два типи волокна: багатомодове й одномодове.

Багатомодове (Multimode Fiber, MMF) – відносно широке в перетині (50-62,5 мікрон), призначене для короткохвильових лазерних променів. «Багатомодове» виходить, що світло по каналі може проходити різними шляхами – багато раз відбиваючись від стінок волокна. Це робить кабель менш чутливим до перегину, але знижує силу і якість сигналу, що обмежує даний тип тільки невеликими дистанціями – до 500 м.

Одномодове (Singlemode Fiber, SMF) – волокно малого діаметра (8-10 мікрон), сигнал по якому передається довгохвильовим лазером, світло якого не видний людському оку. Отут світло може переміщатися єдиним шляхом – по прямій, відповідно сигнал передається швидше й точніше, але встаткування для забезпечення такого роду сигналів коштує значно дорожче, так що використовується, в основному, для зв'язку на великих відстанях (до 50 км). До перегинів і взагалі будь-яким скривленням одномодове волокно куди дошкульніше.

Варто мати через, що для з'єднання двох пристроїв використовується два кабелі. Один використовується для передачі, іншої для прийому. Тому важливо підключити їх коректно (Tx однієї сторони до Rx іншої).

Окремо хочу згадати про такий термін як **темна оптика** (**dark fiber**). Цей термін не виходить, що вона якось спеціальним образом тонована. Це просто виділені оптичні лінії зв'язку, як правило, для зв'язку на більших відстанях (між містами або далеко віддаленими будинками), які беруться в оренду, і для використання яких не потрібно додаткове встаткування посилення сигналу (його забезпечує власник). Однак, тому що це просто оптичний кабель, відданий у ваше повновладне розпорядження, доти поки ви не пустите по ньому свій світловий сигнал, він залишається «темним».

Плавний перехід від FC-0 до FC-1 і назад забезпечує **ASIC** – елемент таких пристроїв як HBA, дискових масивів і комутаторів.

ASIC (абревіатура від англ. application-specific integrated circuit, «інтегральна схема спеціального призначення») – інтегральна схема, спеціалізована для рішення конкретного завдання. На відміну від інтегральних схем загального призначення, спеціалізовані інтегральні схеми застосовуються в конкретному пристрої й виконують строго обмежені функції, характерні тільки для даного пристрою; внаслідок цього виконання функцій відбувається швидше й, в остаточному підсумку, дешевше. Прикладом ASIC може бути мікросхема, розроблена винятково для керування мобільним телефоном, мікросхеми апаратного кодування/декодування аудио- і відео-сигналів (сигнальні процесори).

В устаткуванні Fibre Channel ASIC складається з наступних функціональних елементів:

– Encoder / Decoder – забезпечує кодування кожних 8 біт переданих даних в 10-бітне подання. І декодування назад прийнятих даних.

– SERDES (Serializer / Deserializer) – перетворить паралельний потік 10-бітних порцій даних у послідовний потік 10-бітних порцій даних.

– Transceiver – перетворить електричні імпульси у світлові сигнали.

ASIC

Transceivers, трансівери або SFP – у випадку FC-комутаторів це окремі модулі, необхідні для підключення кабелю до порту. Розрізняються на короткохвильові (Short Wave, SW, SX) і довгохвильові (Long Wave, LW, LX).LW - Трансівери сумісні із багатомодовим і одномодовим волокном.SW-Трансівери – тільки із багатомодовим. І до тих і до інших кабель підключається рознімання LC.

Є ще SFP xWDM (Wave lenght Division Multiplexing), призначені для передачі даних з декількох джерел на далекі відстані єдиним світловим пучком. Для підключення кабелю до них використовується рознімання SC.

FC-1

Перше, що відбувається на цьому рівні – кодування / декодування інформації. Це досить мудрований процес, у ході якого кожні 8 біт вступник інформації перетворяться в 10-бітне подання. Робиться це з метою підвищення контролю цілісності даних, відділення даних від службових сигналів і можливості відновлення тактового сигналу з потоку даних (збереження балансу нулів і одиниць).

Це веде до помітного зниження корисної пропускної здатності, тому що як можна підрахувати, 20% потоку даних є надлишковою службовою інформацією. Але ж крім усього іншого, чималу частину цього потоку може займати службовий трафік.

Однак гарна новина в тому, що кодування 8/10 використовується в устаткуванні 1G, 2G, 4G і 8G. У частині реалізацій 10G і починаючи з 16G кодування здійснюється за принципом 64/66, що істотно збільшує корисне навантаження (до 97% проти 80% у випадку 8/10).

Ordered sets

В українській вікіпедії цей термін переведений як "упорядковані набори". У той час як на мій погляд, слово order отут варто розуміти не в значенні «порядок», а в значенні «наказ, команда».

Для початку варто згадати ще один термін, використовуваний у контексті FC – **transmission word** – мінімальна порція даних для передачі, рівна 4 байтам. Якщо передана інформація менше по обсязі, то transmission word доповнюється спеціальними байтами, що заповнюють (fill bytes), які вирізьблюються на приймачі.

Отож, **ordered sets** – це спеціальні службові transmission words. Діляться на три категорії:

– Роздільники фреймів (Start-of-Frame, SOF і End-of-Frame, EOF).

– Два базових сигнали – IDLE (порт готовий приймати або передавати дані) і R_RDY (receiver ready – порт звільнив буфер для прийому чергової порції даних).

Базові послідовності (primitive sequences):

– NOS (Not Operational) – порт виявив розрив / відсутність з'єднання.

– OLS (Offline State) – порт ініціює встановлення з'єднання, або порт одержав NOS, або порт переходить у стан off-line.

– LR (Link Reset) – ініціалізація скидання з'єднання. Відправляється у випадку одержання OLS або якихось помилок прийому-передачі (як правило, на рівні Flow Control). Порт, Що Відправив, очищає свої буфери і їхні лічильники.

– LRR (Link Reset Response) – відповідь на LR. Порт, Що Відправив, очищає свої буфери і їхні лічильники.

Ініціалізація з'єднання (Link initialization)

При встановленні фізичного з'єднання між портами A і B, між ними відбувається наступний «обмін речовин»:

FC-2

Фрейми (Кадри, Frames)

Всі дані, передані в середовищі Fiber Channel розбиваються на фрейми (кадри). Структура фрейму наступна:

– SoF – 4 байти (1 tw) – ідентифікатор початку фрейму.

– Header – 24 байта (6 tw) – заголовок. Містить таку інформацію як адреса джерела й приймача, тип фрейму (FT-0 – керуючий або FT-1 – дані), номер послідовності й порядковий номер фрейму в ній і інша службово-контрольна інформація.

– Data – 0-2112 байт (0-528 tw) – безпосередньо дані (наприклад, SCSI-команди).

– CRC – 4 байти (1 tw) – контрольна сума.

– EoF – 4 байти (1 tw) – ідентифікатор кінця фрейму.

Проміжки між фреймами заповнюються спеціальними «словами, щозаповнюють,» – fill word. Як правило, це IDLE, хоча починаючи з FC 8G було стандартизоване використання ARB(FF) замість IDLE, з метою зниження електричних перешкод у мідному встаткуванні (але по-умовчанню комутаторами використовується IDLE).

Послідовності (Sequences)

Найчастіше джерело прагне передати приймачу набагато більше інформації, ніж 2112 байт (максимальний обсяг даних одного фрейму). У цьому випадку інформація розбивається на кілька фреймів, а набір цих фреймів називається послідовністю (sequence). Щоб у логічну послідовність фреймів не вклинилося щось зайве при паралельній передачі, заголовок кожного фрейму має поля SEQ_ID (ідентифікатор послідовності) і SEQ_CNT (номер фрейму в послідовності).

Обмін (Exchange)

Одна або кілька послідовностей, відповідальних за якусь одиночну операцію, називається обміном. Джерело й приймач можуть мати кілька паралельних обмінів, але кожний обмін в одиницю часу може містити тільки одну послідовність. Приклад обміну: ініціатор запитує дані (послідовність 1), таргет повертає дані ініціаторові (послідовність 2) і потім повідомляє статус (послідовність 3). У цей набір послідовностей не може вклинитися якийсь сторонній набір фреймів.

Для контролю цього процесу заголовок кожного фрейму включає поля OX_ID (Originator Exchange ID – заповнюється ініціатором обміну) і RX_ID (Responder Exchange ID – заповнюється одержувачем у відповідних фреймах, шляхом копіювання значення OX_ID).

Класи обслуговування (Classes of Services)

Різні додатки висувають різні вимоги до рівня сервісу, гарантії доставки, тривалості з'єднання й пропускної здатності. Деяким додаткам потрібно гарантована пропускна здатність протягом їхньої роботи (бекап). Інші мають змінну активність і не вимагають постійної гарантованої пропускної здатності каналу, але їм потрібно підтвердження в одержанні кожного відправленого пакета. Для задоволення таких потреб і забезпечення гнучкості, FC визначає наступних 6 класів обслуговування.

Class 1

Для цього класу встановлюється виділене з'єднання, що резервує максимальну смугу пропущення між двома пристроями. Вимагає підтвердження про одержання. Вимагає щоб фрейми попадали на приймач у тому же порядку, що вийшли із джерела. Через те, що не дає іншим пристроям використовувати середовище передачі, використовується вкрай рідко.

Class 2

Без постійного з'єднання, але з підтвердженням доставки. Не вимагає відповідності порядку відправлених і доставлених фреймів, так що вони можуть проходити через фабрику різними шляхами. Менш вимогливий до ресурсів, ніж клас 1, але підтвердження доставки приводить до підвищеної утилізації пропускної здатності.

Class 3

Без постійного з'єднання й без підтвердження доставки. Самий оптимальний з погляду використання ресурсів фабрики, але припускає, що протоколи верхніх рівнів зможуть зібрати фрейми в потрібному порядку й перезапросити передачу зниклих фреймів. Найбільше часто використовуваний.

Class 4

Вимагає постійного з'єднання, підтвердження й порядок фреймів як і клас 1. Головна відмінність – він резервує не всю смугу пропущення, а тільки її частина. Це гарантує певне QoS. Підходить для мультимедіа й Enterprise-застосунків, що вимагають гарантованої якості з'єднання.

Class 5

Ще до кінця не описаний і не включений у стандарт. Попередньо, клас, що не вимагає з'єднання, але потребуючої негайної доставки даних у міру їхньої появи, без буферизації на пристроях.

Class 6

Варіант класу 1, але мультикастовий. Тобто від одного порту до декількох джерел.

Class F

Клас F визначений у стандарті FC-SW для використання в міжкомутаторних з'єднаннях (Interswitch Link, ISL). Це сервіс без постійного з'єднання з повідомленнями про збій доставки, що використовується для контролю, керування й конфігурування фабрики. Принцип схожий на клас 2, але той використовується для взаємодіїя між N-портами (порти нод), а клас F – для спілкування E-портів (міжкомутаторних).

Flow Control

З метою запобігання ситуації, коли відправник перевантажить одержувача надлишковою кількістю фреймів так, що вони почнуть відкидатися одержувачем, FC використовує механізми керування потоком переданих даних (Flow Control). Їх два – Buffer-to-Buffer flow control і End-to-End flow control. Їхнє використання регламентується класом обслуговування. Наприклад, клас 1 використовує тільки механізм End-to-End, клас 3 – Buffer-to-Buffer, а клас 2 – обоє ці механізми.

Buffer-to-Buffer flow control

Принцип технології – відправлення будь-якого фрейму повинна бути забезпечена наявністю кредиту на відправлення.

Всі вступники на вхід порту фрейми містяться в спеціальну чергу – буфери. Кількість цих буферів визначається фізичними характеристиками порту. Один буфер (місце в черзі) відповідає одному кредиту. Кожний порт має два лічильники кредитів:

TX BB_Credit – лічильник кредитів передачі. Після відправлення кожного фрейму, зменшується на 1. Якщо значення лічильника стало рівним нулю – передача неможлива. Як тільки від порту-приймача отримане R_RDY, лічильник збільшується на 1.

RX BB_Credit – лічильник кредитів прийому. Як тільки фрейм прийнятий і поміщений у буфер, зменшується на 1. Коли фрейм обробляється або пересилається далі, лічильник збільшується на 1, а відправникові відправляється R_RDY. Якщо значення лічильника падає до 0, то в принципі, прийом нових фреймів повинен бути припинений. На практиці, через помилки синхронізації кредитів може виникнути ситуація, що джерело надіслало ще трохи фреймів уже після того як RX BB_credit став дорівнює нулю. Дана ситуація називається buffer overflow. У більшості реалізацій порт обробляє такі ситуації «добрим-добрій-по^-доброму» – за рахунок резервних буферів. Хоча деяке встаткування в таких випадках може зініціювати Link Reset.

Звідси виходить сильний вплив відстані між портами на продуктивність. Ніж вище відстань і більше пропускна здатність, тим більше фреймів буде відправлено (читай кредитів передачі витрачений) ще перше ніж одержувач одержить хоча б перший. Ситуацію полегшує особливість архітектури FC-комутаторів. Справа в тому, що кількість буферів не закріплено жорстко за кожним портом (крім восьми обов'язкових), а є загальним для всіх. І у випадку визначення «далеких лінків» (автоматично або вручну) кількість виділюваних комутатором буферів для цього порту збільшується. Інший плюс загальної пам'яті – не потрібно ганяти буфери від одного порту до іншому усередині комутатора.

End-to-End flow control

Реалізується лічильником EE_Credit, що визначає максимум фреймів, які джерело може відправити приймачу без одержання підтвердження (Acknowledge, ACK). На відміну від BB_Credit поширюється тільки на фрейми з даними, а обмін/облік відбувається між кінцевими нодами.

1.2 Область застосування

Broadcom пропонує комутатори Fibre Channel трьох поколінь: 8 Гбіт/с (четверте покоління), 16 Гбіт/с (п'яте) і 32 Гбіт/с (шосте). Технологія 8 Гбіт/с застаріла, її життєвий цикл підходить до кінця – по даним Dell'Oro на дане встаткування доводиться зараз близько 1% продажів. У лінійці 8 -гігабітних пристроїв залишилося всього два комутатори, і найближчим часом компанія має намір припинити їхнє виробництво (офіційно про це поки не оголошено). При цьому Brocade пропонує повну лінійку пристроїв п'ятого покоління, цього року компанія значно розширила пропозицію встаткування шостого покоління.

Як показують тести, застосування комутаторів FC на 32 Гбіт/с дозволяє збільшити продуктивність систем зберігання даних (СЗД) у чотири рази. Такі результати свідчать про те, що вузьким місцем є саме мережа, і при установці сучасних СЗД необхідна модернізація мережі. Це підтверджується як ростом продажів устаткування Brocade, так і тим, що в корпоративному сегменті в Україні вже понад 60% продажів становлять комутатори шостого покоління.

Серед новинок цього року – високоплотний комутатор G630 до 128 портів на 32 Гбіт/с і лезо на 64 порту 32 Гбіт/с для комутаторів X6 директорського класу. З випуском флагманського комутатора G630 завершене формування лінійки шостого покоління -по своїх функціональних можливостях він близький до комутаторів директорського класу. Останні призначений для надвеликих мереж зберігання. З появою леза FC 32-64 ємність одного комутатора Brocade X6 Director може бути доведена до 512 портів.

За прогнозами аналітиком до 2021 року 77% всіх продаваних масивів будуть базуватися на флеш-накопичувачах. Поширення флеш-масивів породжує потреба у високошвидкісних з'єднаннях. Наступні покоління Fibre Channel зможуть забезпечити номінальну пропускну здатність до 128 Гбіт/с, однак використовуваний протокол FCP – фактично SCSI поверх Fibre Channel – успадкував всі недоліки SCSI при роботі із флеш-пам'яттю. Перспективний протокол NVMe over Fibre Channel (FC-NVMe) оптимізований для роботи з енергонезалежною пам'яттю й більш ефективно задіє FC як транспорт. Як уважають в Brocade FC-NVMe краще підходить для підключення систем зберігання й породжує менше проблем при впровадженні, ніж альтернативні види доступу NVMe over Fabics на базі Ethernet і TCP.

На ринку вже стали з'являтися перші СЗД із підтримкою FC -NVMe. Так, NetApp оголосила про підтримку FC-NVMe у своїх флеш-масивах ONTAP (на вже встановлених масивах необхідно обновити програмне забезпечення). Про намір випустити системи зберігання з підтримкою даного протоколу заявили такі компанії як Dell EMC і IBM. Комутатори Brocade п'ятого й шостого покоління вже підтримують FC-NVMe. Компанія працює над його подальшою оптимізацією – скороченням затримки й підвищенням продуктивності висновку-виводу-вводу-виводу.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Аналізатори протоколу Fibre Channel

Fibre Channel (FC) (англ. Fibre Channel – волоконний канал) – сімейство протоколів для високошвидкісної передачі даних. Стандартизацією протоколів займається Технічний комітет T11, що входить до складу Міжнародного комітету зі стандартам у сфері IT (International Committee for Information Technology Standards – INCITS). Споконвічне застосування Fibre Channel в області суперкомп'ютерів згодом практично повністю перейшло в сферу мереж зберігання даних, де Fibre Channel використовується як стандартний спосіб підключення до систем зберігання даних рівня підприємства.

Fibre Channel Protocol (FCP) – транспортний протокол (як TCP в IP-мережах), інкапсулює протокол SCSI по мережах Fibre Channel і є основою побудови мереж зберігання даних.

Нижче представлені найбільш популярні аналізатори протоколу Fibre Channel.

Компактна модульна вимірювальна платформа EXFO FTB-1 Pro

Вимір параметрів оптоволоконих ліній (магістральних, міських, мереж доступу, СКС і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 8.1 Professional. Процесор: 4 ядра.

Сенсорний екран 20,3 см. (дозвіл 1280 x 800) з підтримкою мультитач.

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0 і 1G Ethernet (стандартно), Wi -Fi і Bluetooth (опція RF), 3G/4G модем.

Внутрішня пам'ять: 128 ГБ (флеш). Тип акумулятора: Li-ion.

Маса: 1,5 кг. Габарити: 210 х 254 х 66 (96) мм. Робоча температура: від 0°С до +50°С.

Установка одного змінного модуля (можливості платформи залежать від типу модуля):

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTB-750C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTB-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTB-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTB-720C);

– аналізатор протоколів до 11,318 Гбіт/с + рефлектометр (39 дБ) (модуль FTB-730Gv2);

– аналізатор протоколів до 11,318 Гбіт/с + рефлектометр (36 дБ) (модуль FTB-720Gv2);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до **111,81 Гбіт/с** (модуль FTB-890NGE);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 111,81 Гбіт/с (модуль FTB-890);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTB-880Q);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTB-880v2);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTB-870Q);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTB-870v2);

– відеомікроскоп для оптичних конекторів (серія FIP-400B).

Модульна вимірювальна платформа EXFO FTB-2 Pro

Вимір параметрів оптоволоконих ліній (аналіз спектра, дисперсія, рефлектометрія, втрати, ORL і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 10. Процесор: 4 ядра. ОЗП: 4 ГБ.

Сенсорний екран 25,6 см. (дозвіл 1280 x 800).

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0, 1G Ethernet і Display port (стандартно), Wi-Fi і Bluetooth (опція RF).

Внутрішня флеш пам'ять: 64 ГБ (стандартно), 128 ГБ (опція). Тип акумулятора: Li-ion.

Маса: 3,0 кг. Габарити: 199 x 333 x 119 мм. Робоча температура: від 0°C до +50°C.

Два слоти для установки модулів (можливості платформи залежать від типу модулів):

– топовий аналізатор оптичного спектра (модулі FTBx-5245 і FTBx-5255);

– поліпшений аналізатор оптичного спектра (модулі FTB-5240S і FTB-5240BP);

– базовий аналізатор оптичного спектра (модулі FTB-5230S і FTB-5230S-OCA);

– аналізатор хроматичної й ПМД дисперсії (модуль FTB-5700);

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTBx-750C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTBx-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTBx-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTBx-720C);

– рефлектометр (40 дБ, одномод) **зі змінюваною довжиною хвилі** (модуль FTBx-740C);

– вимірник оптичних втрат і зворотного відбиття (модулі серії FTB-3930);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до **111,81 Гбіт/с** (модуль FTBx-88200NGE);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTBx-8880);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTBx-8870);

– відеомікроскоп для оптичних конекторів (серія FIP-400B).

Модульна вимірювальна платформа EXFO FTB-4 Pro

Вимір параметрів оптоволоконих ліній (аналіз спектра, дисперсія, рефлектометрія, втрати, ORL і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 10. Процесор: 4 ядра. ОЗП: 4 ГБ.

Сенсорний екран 25,6 см. (дозвіл 1280 x 800).

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0, 1G Ethernet і Display port (стандартно), Wi-Fi і Bluetooth (опція RF).

Внутрішня флеш пам'ять: 128 ГБ (стандартно). Тип акумулятора: Li-ion.

Маса: 4,6 кг. Габарити: 199 x 333 x 170 мм. Робоча температура: від 0°C до +40°C.

Чотири слота для установки модулів (можливості платформи залежать від типу модулів):

– топовий аналізатор оптичного спектра (модулі FTBx-5245 і FTBx-5255);

– поліпшений аналізатор оптичного спектра (модулі FTB-5240S і FTB-5240BP);

– базовий аналізатор оптичного спектра (модулі FTB-5230S і FTB-5230S-OCA);

– аналізатор хроматичної й ПМД дисперсії (модуль FTB-5700);

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTBx-750C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTBx-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTBx-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTBx-720C);

– рефлектометр (40 дБ, одномод) **зі змінюваною довжиною хвилі** (модуль FTBx-740C);

– вимірник втрат, відбиття й сертифікації волокон (модулі FTBx-940 і FTBx-945);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до **111,81 Гбіт/с** (модуль FTBx-88200NGE);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTBx-8880);

– аналізатор Ethernet, OTN, SDH і **Fibre Channel** до 11,318 Гбіт/с (модуль FTBx-8870);

– відеомікроскоп для оптичних конекторів (серія FIP-400B).

Універсальна вимірювальна платформа EXFO FTB-500

Призначена для тестування всіх існуючих типів мереж. Тестування рівнів: фізичного, оптичного, транспортного й даних.

Убудований вимірник потужності й візуальний дефектоскоп (опція).

Сенсорний екран 30,7 см. (дозвіл 800 x 600).

Процесор Intel Core 2 Duo. Операційна система Windows XP.

Широкий вибір змінних у польових умовах модулів.

Два варіанти виконання: 4 або 8 слотів для установки змінних модулів.

4 слота: вага (з акумулятором): 8,5 кг. габарити: 37 x 30 x 15 см..

8 слотів: вага (з акумулятором): 11 кг. габарити: 37 х 30 х 22 см..

Аналізатори протоколу Fibre Channel фірми Anritsu

Модульний аналізатор транспортних мереж Anritsu MT1000A

Призначений для повного тестування OTN, Ethernet, Fibre Channel, SDH, PDH.

Підтримка **двох портів** для всіх стандартів і швидкостей передачі.

Основні можливості тестування й аналізу Fibre Channel:

Тестування 1GFC, 2GFC, 4GFC, 8GFC і 10GFC з можливістю вбудовування в OTN.

Вимір затримок, тест BER, моніторинг аварій і помилок.

Основні можливості тестування й аналізу OTN (стандарт G.709):

OTU2f (**11,318 Гбіт/с)**, OTU2e, OTU2, OTU1f, OTU1e, OTU1 (2,666 Гбіт/с).

ODU0, ODUflex, ODU1 і ODU2 включаючи ODU0 – ODU2 multistage mapping.

Тестування виникнення помилок OTN відповідно до G.8201 і M.2401.

Тестування функції корекції помилок (FEC) відповідно до ITU-T O.182.

Тестування клієнтського трафіку Ethernet, Fibre Channel і SDH, **убудованого в OTN**.

Основні можливості тестування й аналізу Ethernet:

Тестування Ethernet на швидкостях **10G**, 1G, 100M і 10M.

Генерація трафіку на повній швидкості. Підтримка IPv4 і IPv6.

Тест активації сервісу Ethernet (Y.1564), тест RFC 2544, тест BER.

Тестування **синхронного Ethernet** (G.826x і IEEE 1588 v2).

Тестування MPLS, MPLS-TP і PBB/ PBB-TE.

Підтримка багатопотокового Ethernet і Stacked VLAN (Q-in-Q).

Основні можливості тестування й аналізу SDH/SONET/PDH/DSn:

Тестування SDH (STM-64, STM-16, STM-4, STM-1) і SONET (OC-192, OC-48, OC-12, OC-3).

Тестування PDH (E1, E3, E4) і DSn (DS1, DS3). Докладна статистика аварій і помилок.

Одночасний двонаправлений моніторинг ліній SDH/SONET/PDH/DSn.

Вбудовування й добування сигналів PDH/DSn з SDH/SONET.

Конструктивні характеристики аналізатора MT1000A з модулем MU100010A:

Сенсорний широкоформатний екран 22,8 см. (дозвіл 800 x 480).

Інтерфейси для підключення до ПК: USB, Ethernet, Wi-Fi, Bluetooth.

Час роботи від акумулятора: до 4 годин. Зручний дизайн корпуса.

Маса: 2,7 кг. Габарити: 257 x 164 x 77 мм. Робоча температура: від 0°C до +50°C.

Новітнє рішення для тестування мереж OTN, Ethernet, SDH, PDH і ін.

ТОВ "Техенком" пропонує як зовсім нові прилади, так і колишні у вживанні й повністю відновлені в заводських умовах. Вартість відновлених приладів може бути в 1,5 – 2 рази нижче вартості нових. Всі пропоновані відновлені прилади 100 % працездатні, відповідають технічним вимогам фірми-виробника й забезпечені гарантією 1 рік.

Нижче представлені деякі з безлічі пропонованих нами відновлених приладів.

EXFO FTB-8120NGE, FTB-8130NGE Power Blazer

SDH/SONET/OTN/PDH/DSn: від DS0/E0 до STM-**64/OTU2**.

Ethernet LAN/WAN: від 10 Мбіт/с до **10 Гбіт/с.**

Fibre Channel: 1x, 2x, 4x і 10x.

Підтримують тестування SDH/SONET наступного покоління. Сумісні зі стандартом тестування EtherSAM (ITU-T Y.156sam).

Розмір: 2 слота. Сумісні із платформами EXFO FTB-500 і FTB-200.

Універсальні модульні вимірювальні платформи для аналізу протоколів

Загальна інформація

Кілька провідних виробників вимірювальної техніки випускають **універсальні модульні вимірювальні платформи**, можливості яких наращуються за рахунок установки змінних модулів. Як правило до кожної вимірювальної платформи випускається кілька десятків модулів для виміру різних параметрів волоконно-оптичних мереж, аналізу комунікаційних протоколів (Transport and Datacom, скорочено T&D) і тестування мереж доступу (xDSL, xPON/FTTx, Ethernet).

ТОВ "Техенком" здійснює поставку всіх типів сучасних моделей модульних вимірювальних платформ провідних виробників: від мініатюрних, що вміщаються в одній руці, до габаритних, багатослотових, здатних виконувати повний набір тестів телекомунікаційної інфраструктури.

У цей час випускаються модулі для аналізу таких комунікаційних протоколів: Ethernet / IP / MPLS / VPN, IPTV / VoIP, SDH / SONET / OTN, Fibre Channel, PDH / DSn, VDSL2/ADSL1/2/2+, CPRI і ін. Залежно від типу модульної системи, у неї можна встановити від одного до восьми змінних модулів.

Універсальні модульні вимірювальні платформи й модулі для T&D фірми EXFO

Компактна модульна вимірювальна платформа EXFO FTB-1v2 Pro

Вимір параметрів оптоволоконих ліній (магістральних, міських, мереж доступу, СКС і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 10. Процесор: 4 ядра.

Сенсорний екран 20,3 см. (дозвіл 1280 x 800) з підтримкою мультитач.

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0 і 1G Ethernet (стандартно), Wi -Fi і Bluetooth (опція RF).

Флеш пам'ять: 128 ГБ. Акумулятор: Li-ion. Робоча температура: від 0°С до +50°С.

Маса: від 2 до 3,2 кг (залежить від модифікації). Габарити: 210 х 254 х 66, (96) або (122) мм.

Установка одного або двох модулів (можливості платформи залежать від типу модулів):

Платформа EXFO FTB-1v2 Pro сумісна з такими модулями для T&D:

– аналізатори Ethernet, OTN, SDH і CPRI до 11,318 Гбіт/с (модулі FTBx -8880 і FTBx-8870);

– аналізатори Ethernet, OTN, SDH до **111,81 Гбіт/с** (модулі FTBx-88200NGE і FTBx-88260)

Платформа EXFO FTB-1v2 Pro сумісна з такими модулями для ВОЛЗ:

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTBx-750C);

– рефлектометр (40 дБ, одномод) **зі змінюваною довжиною хвилі** (модуль FTBx-740C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTBx-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTBx-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTBx-720C);

– вимірник втрат, відбиття й сертифікації волокон (модулі FTBx-940 і FTBx-945);

– аналізатор оптичного спектра (модуль FTBx-5235);

– відеомікроскоп для оптичних конекторів (серія FIP-400B)

Модульна вимірювальна платформа EXFO FTB-2 Pro

Вимір параметрів оптоволоконих ліній (аналіз спектра, дисперсія, рефлектометрія, втрати, ORL і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 10. Процесор: 4 ядра. ОЗП: 4 ГБ.

Сенсорний екран 25,6 см. (дозвіл 1280 x 800).

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0, 1G Ethernet і Display port (стандартно), Wi-Fi і Bluetooth (опція RF).

Внутрішня флеш пам'ять: 64 ГБ (стандартно), 128 ГБ (опція). Тип акумулятора: Li-ion.

Маса: 3,0 кг. Габарити: 199 x 333 x 119 мм. Робоча температура: від 0°C до +50°C.

Два слота для установки модулів (можливості платформи залежать від типу модулів).

Платформа EXFO FTB-2 Pro сумісна з такими модулями для T&D:

– аналізатори Ethernet, OTN, SDH до **111,81 Гбіт/с** (модулі FTBx-88200NGE і FTBx-88260);

– аналізатори Ethernet, OTN, SDH і CPRI до 11,318 Гбіт/с (модулі FTBx - 8880 і FTBx-8870)

Платформа EXFO FTB-2 Pro сумісна з такими модулями для ВОЛЗ:

– топовий аналізатор оптичного спектра (модулі FTBx-5245 і FTBx-5255);

– базовий аналізатор оптичного спектра (модулі FTBx-5235);

– аналізатор хроматичної й ПМД дисперсії (модуль FTB-5700);

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTBx-750C);

– рефлектометр (40 дБ, одномод) **зі змінюваною довжиною хвилі** (модуль FTBx-740C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTBx-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTBx-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTBx-720C);

– вимірник втрат, відбиття й сертифікації волокон (модулі FTBx-940 і FTBx-945);

– відеомікроскоп для оптичних конекторів (серія FIP-400B)

Модульна вимірювальна платформа EXFO FTB-4 Pro

Вимір параметрів оптоволоконих ліній (аналіз спектра, дисперсія, рефлектометрія, втрати, ORL і ін.), а також тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.).

Операційна система: Windows 10. Процесор: 4 ядра. ОЗП: 4 ГБ.

Сенсорний екран 25,6 см. (дозвіл 1280 x 800).

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0, 1G Ethernet і Display port (стандартно), Wi-Fi і Bluetooth (опція RF).

Внутрішня флеш пам'ять: 128 ГБ (стандартно). Тип акумулятора: Li-ion.

Маса: 4,6 кг. Габарити: 199 x 333 x 170 мм. Робоча температура: від 0°C до +40°C.

Чотири слота для установки модулів (можливості платформи залежать від типу модулів).

Платформа EXFO FTB-4 Pro сумісна з такими модулями для T&D:

– аналізатори Ethernet, OTN, SDH до **111,81 Гбіт/с** (модулі FTBx-88200NGE і FTBx-88260);

– аналізатори Ethernet, OTN, SDH і CPRI до 11,318 Гбіт/с (модулі FTBx - 8880 і FTBx-8870)

Платформа EXFO FTB-4 Pro сумісна з такими модулями для ВОЛЗ:

– топовий аналізатор оптичного спектра (модулі FTBx-5245 і FTBx-5255);

– базовий аналізатор оптичного спектра (модулі FTBx-5235);

– аналізатор хроматичної й ПМД дисперсії (модуль FTB-5700);

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTBx-750C);

– рефлектометр (40 дБ, одномод) **зі змінюваною довжиною хвилі** (модуль FTBx-740C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTBx-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTBx-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTBx-720C);

– вимірник втрат, відбиття й сертифікації волокон (модулі FTBx-940 і FTBx-945);

– відеомікроскоп для оптичних конекторів (серія FIP-400B)

Компактна вимірювальна платформа EXFO FTB-1 Pro

Тестування протоколів (Ethernet, OTN, SDH/PDH, CPRI і ін.), а також вимір параметрів оптоволоконих ліній (магістральних, міських, мереж доступу, СКС і ін.).

Операційна система: Windows 8.1 Professional. Процесор: 4 ядра.

Сенсорний екран 20,3 см. (дозвіл 1280 x 800) з підтримкою мультитач.

Убудований вимірник оптичної потужності й просвітлення VFL (опція VPM2X).

Інтерфейси: USB 3.0 і 1G Ethernet (стандартно), Wi -Fi і Bluetooth (опція RF), 3G модем.

Внутрішня пам'ять: 128 ГБ (флеш). Тип акумулятора: Li-ion.

Маса: 1,5 кг. Габарити: 210 x 254 x 66 (96) мм. Робоча температура: від 0°C до +50°C.

Установка одного змінного модуля (можливості платформи залежать від типу модуля).

Платформа EXFO FTB-1 Pro сумісна з такими модулями для T&D:

– аналізатор Ethernet, OTN, SDH/PDH, FC і CPRI до **111,81 Гбіт/с** (модуль FTB-890NGE);

– аналізатор Ethernet, OTN, SDH, FC і CPRI до 111,81 Гбіт/с (модуль FTB-890);

– аналізатор Ethernet, OTN, SDH/PDH, FC і CPRI до 11,318 Гбіт/с (модуль FTB-880Q);

– аналізатор Ethernet, OTN, SDH/PDH, FC і CPRI до 11,318 Гбіт/с (модуль FTB-880v2);

– аналізатор Ethernet, OTN, SDH/PDH, FC і CPRI до 11,318 Гбіт/с (модуль FTB-870Q);

– аналізатор Ethernet, OTN, SDH/PDH, FC і CPRI до 11,318 Гбіт/с (модуль FTB-870v2);

– аналізатор протоколів до 11,318 Гбіт/с + рефлектометр (39 дБ) (модуль FTB-730Gv2);

– аналізатор протоколів до 11,318 Гбіт/с + рефлектометр (36 дБ) (модуль FTB-720Gv2)

Платформа EXFO FTB-1 Pro сумісна з такими модулями для ВОЛЗ:

– рефлектометр (46 дБ, одномод) для магістралей (модуль FTB-750C);

– рефлектометр (42 дБ, одномод) для міських мереж (модуль FTB-735C);

– рефлектометр (39 дБ, одномод) для PON, FTTx / MDU (модуль FTB-730C);

– рефлектометр (36 дБ, одномод + багатомод) для мереж доступу й СКС (модуль FTB-720C);

– відеомікроскоп для оптичних конекторів (серія FIP-400B)

Модульний аналізатор транспортних мереж Anritsu MT1000A

Призначений для повного тестування OTN, Ethernet, Fibre Channel, SDH, PDH.

Підтримка **двох портів** для всіх стандартів і швидкостей передачі.

Основні можливості тестування й аналізу OTN (стандарт G.709):

OTU2f (**11,318 Гбіт/с)**, OTU2e, OTU2, OTU1f, OTU1e, OTU1 (2,666 Гбіт/с).

ODU0, ODUflex, ODU1 і ODU2 включаючи ODU0 – ODU2 multistage mapping.

Тестування виникнення помилок OTN відповідно до G.8201 і M.2401.

Тестування функції корекції помилок (FEC) відповідно до ITU-T O.182.

Тестування клієнтського трафіку Ethernet, Fibre Channel і SDH, **убудованого в OTN**.

Основні можливості тестування й аналізу Ethernet:

Тестування Ethernet на швидкостях **10G**, 1G, 100M і 10M.

Генерація трафіку на повній швидкості. Підтримка IPv4 і IPv6.

Тест активації сервісу Ethernet (Y.1564), тест RFC 2544, тест BER.

Тестування **синхронного Ethernet** (G.826x і IEEE 1588 v2).

Тестування MPLS, MPLS-TP і PBB/ PBB-TE.

Підтримка багатопотокового Ethernet і Stacked VLAN (Q-in-Q).

Основні можливості тестування й аналізу SDH/SONET/PDH/DSn:

Тестування SDH (STM-64, STM-16, STM-4, STM-1) і SONET (OC-192, OC-48, OC-12, OC-3).

Тестування PDH (E1, E3, E4) і DSn (DS1, DS3). Докладна статистика аварій і помилок.

Одночасний двонаправлений моніторинг ліній SDH/SONET/PDH/DSn.

Вбудовування й добування сигналів PDH/DSn з SDH/SONET.

Основні можливості тестування й аналізу Fibre Channel:

Тестування 1GFC, 2GFC, 4GFC, 8GFC і 10GFC з можлив істю вбудовування в OTN.

Вимір затримок, тест BER, моніторинг аварій і помилок.

Конструктивні характеристики аналізатора MT1000A з модулем MU100010A:

Сенсорний широкоформатний екран 22,8 см. (дозвіл 800 x 480).

Інтерфейси для підключення до ПК: USB, Ethernet, Wi-Fi, Bluetooth.

Час роботи від акумулятора: до 4 годин. Зручний дизайн корпуса.

Маса: 2,7 кг. Габарити: 257 x 164 x 77 мм. Робоча температура: від 0°C до +50°C.

Портативна платформа Anritsu MT9090A Network Master

Призначена для вимірів на мережах **PON FTTx, CWDM і Ethernet**.

Легка й міцна конструкція, відсутність вентилятора. Інтуїтивний користувальницький інтерфейс дозволяє технічному персоналу з різним рівнем кваліфікації легко працювати із платформою.

Убудований вимірник потужності й візуальний дефектоскоп (опція). Широкоформатний екран 11 см. (дозвіл 480 x 272). Інтерфейси: USB.

Внутрішня пам'ять: 8 ГБ (флеш). Час автономної роботи: 3,5 години.

Вага (з акумулятором): 0,8 кг. Габарити: 19 x 10 x 5 см..

Один слот для установки змінного модуля.

Anritsu MU909060A1/A2/A3

Ethernet: від 10 Мбіт/с до **1 Гбіт/с.**

Порти: два оптичних SFP, два провідних RJ-45 або

один оптичний SFP і один провідний RJ-45.

Тест **RFC 2544**. Можливість роботи в прозорому режимі (два порти).

Підтримка **VoIP, IPTV, Stacked VLAN (Q-in-Q), MPLS, IPv4 і IPv6**.

Перевірка фізичного рівня: оптична потужність і перевірка кручений пари.

Сполучимо з портативною платформою Anritsu MT9090A.

Аналізатори протоколів Ethernet / IP:

Сучасні аналізатори (тестери) мереж Ethernet дозволяють проводити повну перевірку параметрів роботи мереж на швидкостях від 10 Мбіт/с до 100 Гбіт/с.

Перелік проведених вимірів може включати всі рівні стека протоколів: від фізичного (крученої пари й оптоволокно) до рівня застосунків і сервісів (IPTV / VoIP і інші). При цьому більш ість таких приладів виконано в компактних портативних корпусах з акумуляторним живленням.

Аналізатори мереж 1G і 10G серії EXFO MAX-800

Аналіз **Ethernet до 1 Гбіт/с** (модель MAX-860).

Аналіз **Ethernet до 10,3 Гбіт/с** (модель MAX-860G).

Аналіз **Ethernet, SDH/PDH і OTN до 10,7 Гбіт/с** (модель MAX-880).

Призначені для базового тестування Ethernet (LAN/WAN), SDH/PDH і OTN.

Стандартні тести: EtherSAM (Y.1564), RFC 2544, BERT, генерація трафіку.

Опції: IPv6, MPLS, SDH (до STM -64), OTN (до OTU2), PDH, два порти, наскрізний режим.

Інтерфейси для підключення до ПК: USB, 1G Ethernet. Опціонально: Wi-Fi, Bluetooth.

Сенсорний екран 20,3 см. (дозвіл 1280 x 800). Акумулятор: Li-ion.

Маса: до 2,6 кг. Габарити: 210 x 254 x 66 мм. Робоча температура: від 0°C до +50°C.

Модульний аналізатор транспортних мереж Anritsu MT1000A

Призначений для повного тестування OTN, Ethernet, Fibre Channel, SDH, PDH.

Підтримка **двох портів** для всіх стандартів і швидкостей передачі.

Основні можливості тестування й аналізу Ethernet:

Тестування Ethernet на швидкостях **10G**, 1G, 100M і 10M.

Генерація трафіку на повній швидкості. Підтримка IPv4 і IPv6.

Тест активації сервісу Ethernet (Y.1564), тест RFC 2544, тест BER.

Тестування **синхронного Ethernet** (G.826x і IEEE 1588 v2).

Тестування MPLS, MPLS-TP і PBB/ PBB-TE.

Підтримка багатопотокового Ethernet і Stacked VLAN (Q-in-Q).

Основні можливості тестування й аналізу OTN (стандарт G.709)

OTU2f (**11,318 Гбіт/с)**, OTU2e, OTU2, OTU1f, OTU1e, OTU1 (2,666 Гбіт/с).

ODU0, ODUflex, ODU1 і ODU2 включаючи ODU0 – ODU2 multistage mapping.

Тестування виникнення помилок OTN відповідно до G.8201 і M.2401.

Тестування функції корекції помилок (FEC) відповідно до ITU-T O.182.

Тестування клієнтського трафіку Ethernet, Fibre Channel і SDH, **убудованого в OTN**.

Основні можливості тестування й аналізу SDH/SONET/PDH/DSn

Тестування SDH (STM-64, STM-16, STM-4, STM-1) і SONET (OC-192, OC-48, OC-12, OC-3).

Тестування PDH (E1, E3, E4) і DSn (DS1, DS3). Докладна статистика аварій і помилок.

Одночасний двонаправлений моніторинг ліній SDH/SONET/PDH/DSn.

Вбудовування й добування сигналів PDH/DSn з SDH/SONET.

Основні можливості тестування й аналізу Fibre Channel

Тестування 1GFC, 2GFC, 4GFC, 8GFC і 10GFC з можливістю вбудовування в OTN.

Вимір затримок, тест BER, моніторинг аварій і помилок.

Конструктивні характеристики аналізатора MT1000A з модулем MU100010A

Сенсорний широкоформатний екран 22,8 см. (дозвіл 800 x 480).

Інтерфейси для підключення до ПК: USB, Ethernet, Wi-Fi, Bluetooth.

Час роботи від акумулятора: до 4 годин. Зручний дизайн корпуса.

Маса: 2,7 кг. Габарити: 257 x 164 x 77 мм. Робоча температура: від 0°C до +50°C.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero RAD Studio Delphi 10.3.2 Rio Architect – це найшвидший спосіб створювати й обновляти інтенсивно працюючі з даними, сильно взаємодіючі застосунки з візуально насиченим користувальницьким інтерфейсом для Windows 10, Mac, мобільних пристроїв, IoT і інших платформ за допомогою Object Pascal і C++. Широкий вибір функцій підтримки Windows 10, у тому числі нові компоненти VCL для Windows 10, стилі для VCL і FMX, а також служби UWP (універсальної платформи Windows), наприклад повідомлення, дозволяють легко й швидко перенести застосунки в Windows 10, зберігши користувачів. Нова платформа дозволяє підтримувати великі проекти на більшому числі платформ із подвоєним обсягом пам'яті в середовищі розробки й удвічі більшим розміром підтримуваних проектів. Крім того, підтримка декількох моніторів і десятки нових функцій середовища розробки, призначених для прискорення створення коду, зроблять роботу як ніколи ефективною. За допомогою RAD Studio 10 розроблювачі зможуть створювати застосунки в 5 разів швидше в порівнянні з іншими інструментами, а розробка застосунків для декількох настільних, мобільних, хмарних платформ і платформ баз даних, включаючи 32і 64 -розрядні версії Windows 10, Mac OS X, iOS і Android, стане ще швидше.

Зміни у версії 10.3 Rio:

– Створюйте міжплатформені застосунки. 80% інтернет користувачів мають смартфони й застосунки доступу, а також дані з мобільного пристрою й ноутбука / настільного комп'ютера, саме тому так важливо в цей час, щоб застосунки працювали в будь-якому пристрої.

– У всіх версіях Professional, Enterprise і Architect RAD Studio 10.3 надається підтримка процесу розробки застосунків для мобільних пристроїв. Розроблювачі RAD Studio кодують лише один раз, компілюють споконвічно для

кожної платформи, що скорочує час і трудозатрати на вивчення декількох мов і дозволяє паралельно управляти циклами розробки.

– Підтримка Android API26, відповідність вимогам Google Play Store відносно нових застосунків із серпня 2018 року й відновлення застосунків з листопада 2018 року.

– Власні елементи керування Android і стилізовані елементи керування FMX в одній і тій же формі Android, включаючи тему матеріального дизайну для Android 5.0 або вище.

– Підтримка iOS 12 (32і 64-біт) для створення App Store і корпоративних застосунків.

– Підтримка смайликів Юнікод.

– Програмуйте по-своєму. Завдяки двом новим темам самостійне налаштування інтегрованого середовища розробки для відповідності вашому стилю кодування ще ніколи не була настільки простій.

– Темне й світле оформлення Незалежно від того чи волієте ви кодувати вночі або у світлий час доби, завдяки темному й світлому оформленню RAD Studio ви можете вибрати потрібний вам стиль. Було доведено, що темне оформлення допомагає знизити зорову напругу в умовах низького освітлення, дозволяючи вам працювати більш продуктивно вночі. Немає нічого простіше, ніж перейти від темного до світлого оформлення й навпаки за допомогою меню панелі інструментів.

– Виконаєте користувальницьке налаштування свого середовища розробки Поліпшена програма установки інтерфейсу користувача й менеджера ліцензій інтерфейсу користувача дозволяє визначити ті можливості, які необхідні й опустити непотрібні, незалежно від того чи розробляєте ви застосунки для декількох платформ або всього однієї.

– Чистий, оновлений інтерфейс користувача інтегрованого середовища розробки Знайдіть потрібні можливості. Швидко. Головне вікно інтегрованого середовища розробки відцентровано й відрізняється високим ступенем читаності.

Ви з легкістю визначите, де перебуває область фокусування клавіатури з оновленими змінами фонових квітів фокуса. Вкладки редактора більше, що полегшує читання шрифтів, тому ви можете швидко внести зміни й зберегти кодування.

– Чудові застосунки Windows з VCL. Бібліотека візуальних компонентів (Visual Component Library, VCL) пропонує просту й візуальну розробку користувальницького інтерфейсу застосунки, у версії 10.3 представлені нові відновлення, які дозволять вашим додаткам виглядати сучасними й свіжими.

– Розширена підтримка HighDPI. Завдяки новому елементу керування VCL High DPI ImageList у версії 10.3 розроблювачі, що створюють нові застосунки VCL для Windows або оновлюючі існуючі застосунки для High DPI дисплеїв, можуть повністю підтримувати зроблені до рівня пікселів зображення зі змінною розв'язною здатністю на всіх елементах керування, а також будь-яке користувальницьке креслення, що вимагає масштабованих зображень для моніторів з різною розв'язною здатністю.

– Підтримка Per Monitor V2. Переконаєтеся, що ваш додаток масштабується правильно для всіх типів масштабування в Windows, реагуючи на зміни масштабування DPI на різних екранах під час виконання.

– Розширена підтримка Windows 10 і WinRT API. Сюди відноситься ряд ключових API-інтерфейсів WinRT і останні API-інтерфейси Windows 10, включаючи готові до використання компонента для убудованих у застосунки покупок і випробувань у магазині Windows 10 Store.

– Розгортання застосунків на основі служб за допомогою RAD Server. Продуктивність RAD Server була значно поліпшена завдяки десятикратному збільшенню потужності відносно простих операцій.

– Нові компоненти обробки JSON допоміжного засобу.

– Розширена підтримка RAD Server для клієнта Ext JS. Об'єднайте зовнішній інтерфейс javascript і веб-службу, підтримувану REST Server REST. (У версії Architect тепер включена ліцензія ExtJS Professional).

– Версії Enterprise включають ліцензію для одиничного розгортання RAD Server.

– Версії Architect включають ліцензію для розподіленого розгортання RAD Server.

– Що нового в C++? Підтримка C++17 Win32 збільшує продуктивність, поліпшує роботу компілятора й прискорює процес кодування. Були оновлені RTL і STL.

– Нова версія STL/Dinkumware 2018 для Win32 і Win64.

– Поліпшене автодоповнення коду Автодоповнення коду для даного компілятора тепер асинхронне, швидше й із кращими результатами, ніж у попередньому автодоповненні коду C++. Уведення тексту не буде припинятися, поки виконується розрахунок.

– Тепер є підтримка налагодження для оптимізації компонувань.

– 2X швидкість математичної продуктивності для Win64.

– Нові додаткові лабораторії C++ в GetIt.

– Нові й поліпшені можливості роботи з базами даних. InterBase 2017 / IBToGo 2017 в RAD Studio. Версії Professional включають ліцензію розроблювача InterBase 2017, у той час як версії Enterprise і Architect містять у собі ліцензії InterBase ToGo. InterBase ToGo доповнена можливістю шифрування, функціями зміни подань, призначених для простої синхронізації даних застосунку по підписці без обмежень за розміром файлу бази даних.

– Поліпшена й оновлена підтримка для популярних баз даних, включаючи mySQL v8.0, MariaDB 10.3, SQL Server 2017, PostgreSQL v10, Firebird v3.0, MongoDB, InterBase, SQLite 3.23.1, SQL Anywhere і багатьох інших.

– Удосконалення DataSnap.

– Поліпшення REST. Підтримка додаткових родинних REST методів, типів і властивостей.

– Повністю оновлений модуль живлення версії Architect. Одержіть більше від версії Architect, включаючи ці ліцензії сімейства Idera.

– Ліцензія Sencha ExtJS Professional: Створіть свій ідеальний мережевий вхідний інтерфейс за допомогою javascript і ExtJS.

– Ліцензія на розгортання InterBase ToGo. Додайте сховище даних у свої застосунки за допомогою цієї гнучкої, зашифрованої бази даних, що вбудовується.

– Ліцензія для розподіленого розгортання RAD Server. Ідеально підходить для серверного застосунку архітектури мікросервісів.

– Ліцензія AquaData Studio. Вражаючий аналіз бази даних.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи зберігання даних з використанням технології Fibre Channel 6.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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

Чому варто розглянути використання Fibre Channel? Fibre Channel як і раніше є найбільш безпечним, надійним, економічно ефективним і масштабованим протоколом для з'єднання серверів і сховищ, а також єдиним протоколом, спеціально призначеним для передачі трафіку сховища.

Всі ми знаємо, що обс яг даних продовжує рости в геометричній прогресії, і що самі дані є тією новою валютою, на яку розраховують підприємства.
Здатність вчасно реагувати на ці дані може вплинути на конкурентоспроможність бізнесу на ринку. Тому швидкий і надійний доступ до даних має першорядне значення, а базова інфраструктура, що зв'язує користувача із системами зберігання даних, є більше важливої, ніж коли-або колись.

У сучасному центрі обробки даних архітектори можуть вибирати з безлічі різних варіантів підключення, але Fibre Channel був і залишиться джерелом життєвої сили для підключення до загальних сховищ. Це пов'язане з тим, що Fibre Channel є найбільш безпечним, надійним, економічно ефективним і масштабованим протоколом для з'єднання серверів і сховищ, а також єдиним протоколом, спеціально призначеним для передачі трафіку сховища.

Fibre Channel існує вже кілька десятиліть і як і раніше є основним вибором для підключення до загального сховища в центрі обробки даних. За допомогою Fibre Channel створюється виділена мережа зберігання, а команди зберігання SCSI направляються між сервером і пристроями зберігання із пропускною здатністю до 28,05 Гбіт/с (32GFC) і з IOPS, що перевищує один мільйон.
Оскільки Fibre Channel споконвічно був розроблений для трафіку сховищ, він працює дуже надійно й забезпечує високопродуктивний зв'язок. Адаптери HPE StoreFabric 16GFC і 32GFC і інфраструктура комутації забезпечують пропускну

здатність, кількість операцій вводу-виводу в секунду й низьку затримку, необхідні в центрах обробки даних сьогодні й на роки вперед.

Досягнення в технології Fibre Channel тримають його на випередження, коли справа доходить до підключення.

Наприклад, інфраструктура HPE StoreFabric 16GFC і 32GFC уже здатна підтримувати трафік зберігання NVMe, навіть до того, як власні масиви зберігання NVMe стануть масовими. Інші розширені можливості включають розширену діагностику, спрощене розгортання й оркестровку й підвищену надійність, таку як T-10 PI, двопортова ізоляція й багато чого іншого.

Інший популярний варіант підключення до сховища – iSCSI. З iSCSI, команди зберігання в стандартній мережі TCP / IP, і це відмінно підходить для систем низького й середнього рівня, де продуктивність і безпека не є основними вимогами. Розповсюджена омана про Fibre Channel полягає в тому, що, оскільки він використовує виділену мережу зберігання даних, він дорожче, ніж iSCSI. Хоча iSCSI може працювати в тій же мережі Ethernet, що й весь звичайний мережевий трафік, для забезпечення продуктивності, необхідної більшості клієнтів від своїх систем зберігання, iSCSI повинен працювати в сегментованій або виділеній мережі Ethernet, ізольованої від звичайного мережевого трафіку. Це означає складні конфігурації VLAN і політики безпеки або повністю виділену мережу Ethernet. Так само, як Fibre Channel.

Єдина реальна різниця у вартості між FC і iSCSI – це коли DAC – кабелі використовуються в реалізаціях iSCSI. Але з обмеженням відстані 5 метрів, використовуючи DAC – кабелі. Це може нормально працювати для клієнтів малого й середнього бізнесу, що мають тільки один масив зберігання, але DAC – кабелі погано працюють у великомасштабному центрі обробки даних.

Коли ви дивитеся на топологію мережі зберігання даних, кращі практики ідентичні для iSCSI і Fibre Channel. Для забезпечення відказостійкості й усунення простоїв у проекті мережі зберігання даних (SAN) передбачено два ідентичних мережевих шляхи між серверами й сховищем.

Однак одна істотна відмінність полягає в тому, що мережі Fibre Channel не так піддані порушенням безпеки, як Ethernet. Коли ви востаннє чули про злом мережі Fibre Channel? Ніколи? Як щодо мережі Ethernet?

Безпека є однією з головних причин того, що Fibre Channel буде залишатися опорою в центрі обробки даних протягом багатьох років.

Як ми очікуємо, набір команд SCSI буде замінений командами Non-Volitile Memory Express або NVMe. NVMe – це оптимізований набір команд, розроблений для SSD і пам'яті класів зберігання, що набагато ефективніше, ніж SCSI. Крім того, NVMe являє собою багаторядну архітектуру із чергами вводу-виводу до 64 КБ, причому кожна черга вводу-виводу підтримує до 64 КБ команд. У порівнянні з SCSI з однією чергою й 64 командами, NVMe може забезпечити значно більше високу продуктивність.

Сучасна інфраструктура HPE StoreFabric 16GFC і 32GFC, що підтримує команди SCSI, також може запускати команди NVMe у мережі SAN або в структурі, як вона називається. При використанні Ethernet клієнтам буде потрібно впровадити RDMA з низькою затримкою в порівнянні з конвергентним Ethernet або RoCE, щоб повною мірою використовувати переваги NVMe. Однак цей підхід вимагає складної реалізації Ethernet без втрат з використанням мостів центрів обробки даних (DCB) і керування пріоритетними потоками (PFC). Складність мережі для NVMe через Ethernet буде величезним бар'єром для більшості клієнтів, особливо коли розгорнута сьогодні FC SAN прекрасно працює зі сховищем NVMe завтрашнього дня.

Fibre Channel – зверхшвидкістна (до 1 Гбіт/с і вище) схема повнодуплексної передачі даних. Технологія забезпечує передачу даних з малою затримкою (10-30 мкс) на відстані до 10 кілометрів.

У назві Fibre Channel (якщо переводити буквально, те 'волоконний канал') криється підступ, оскільки оптичне волокно зовсім не при чому. Середовищем передачі даних може бути крім оптоволокна й кручена пари, і коаксіал. Архітектура Fibre Channel являє собою суміш канальної й мережевої топології.

Варіанти топології Fibre Channel

Топологія Fibre Channel.

Усього в Fibre Channel існують три варіанти топології.

Найпростішою топологією є, мабуть, 'точка-точка'. У ній два пристрої Fibre Channel з'єднані прямим з'єдн анням між собою. При цьому передавач одного пристрою з'єднується із приймачем другого, і, відповідно, навпаки. Пристрої, що з'єднуються, повинні працювати на одній швидкості, при цьому їм доступна вся пропускна здатність з'єднання.

Більше розповсюдженим варіантом топології є арбітражна петля (FC-AL). При цьому способі з'єднання можливе підключення до 127 пристроїв без використання комутаторів. Хоча топологія й називається петлею, по суті це ланцюжок пристроїв, що одним кінцем може бути підключена до комутатора (а може й не бути). При з'єднанні пристроїв арбітражною петлею пропускна здатність є поділюваної, тобто в один конкретний момент часу тільки два пристрої можуть взаємодіяти один з одним. Також зберігається вимога, відповідно до якого пристрої, що з'єднуються, повинні працювати на одній швидкості.

Третій варіант топології – з'єднання з комутуючою структурою. За рахунок каскадного застосування комутаторів можливе з'єднання дуже великої кількості пристроїв – понад 16 мільйонів. Обмежень на відповідність швидкостей пристроїв, що з'єднуються, у цьому випадку немає.

Позначення портів

Порти Fibre Channel діляться на кілька типів залежно від типу пристрою, його призначення й підтримуваної топології. Для зручності всі вони мають літерні позначення.

– N (Node Port, N_Port) – порт Fibre Channel на кінцевому пристрої (сервері, дисковому масиві, принтері й т. П.). Node Port у перекладі означає 'вузловий порт'.

– F (Fabric Port, F_Port) – порт на комутаторі (або 'комутуючий порт'), до якого підключається вузловий порт.

– NL_Port і FL_Port . Якщо перераховані вище вузлові або комутуючі порти можуть підключатися до арбітражної петлі, то вони маркіруються додатково буквою L від англійського loop, тобто 'петля'.

– E (Expansion Port, E_Port) – порт на комутаторі для підключення одного комутатора до іншого.

– G (Generic Port, G_Port) – універсальний порт, до якого може бути підключений не тільки інший комутатор, але й вузол.

– GL – універсальний порт із підтримкою підключення арбітражної петлі.

Різновиди устаткування

Концентратори

У топологічної схеми «арбітражна петля» крім поділу пропускної здатності є й інші недоліки. Наприклад, при відмові адаптера на якому-небудь пристрої або розриві в з'єднуючому кабелі петля виявляється повністю непрацездатною. При зміні набору пристроїв у петлі (зокрема, при додаванні нового пристрою) вся петля повинна бути ініціалізована заново (щоб підключений пристрій міг одержати адресу), причому ця процедура може займати досить багато часу.

Для рішення цих і подібних проблем використовуються концентратори Fibre Channel. Одночасно із цим, їхнє застосування дозволяє використовувати більше зручну для підключення нових пристроїв фізичну топологію «зірка», хоча логічно топологія залишається кільцем. Звичайно концентратори мають не більше 10 портів. Однак це обмеження легко перебороти за рахунок каскадного підключення концентраторів.

Концентратори підвищують надійність використання арбітражної петлі за рахунок застосування схеми обходу портів (Port Bypass Circuit, PBC). PBC виконує дві функції: по-перше, автоматично виявляє наявність вузла й включає його в петлю, по-друге, виявляє відмова вузла й виключає його з петлі. Найбільш

просунуті концентратори підтримують віддалене керування й інші розвинені функції.

Комутатори

Комутатори Fibre Channel є істотно більше дорогими пристроями, ніж концентратори Fibre Channel. На відміну від концентраторів, вони дозволяють надати вузлу виділену пропускну здатність і створювати топології з незрівнянно більшим числом вузлів (224). Крім того, комутатори можуть мати порти з підтримкою різних швидкостей і середовищ передачі.

Комутатори Fibre Channel прості в установці й використанні завдяки наявності функцій самоконфігурації й самоврядування. Всі операції конфігурування здійснюються автоматично. Наприклад, при підключенні вузла до комутатора він реєструється на комутаторі й погоджує з ним взаємоприйняті параметри. При підключенні комутатора до комутатора вони визначають конфігурацію й адреси. У випадку універсального порту (GL_Port) комутатор також сам установлює, до чого він підключений – до іншого комутатора, до петлі або до вузла. З огляду на цінові фактори, для організації взаємодії між пристроями в декількох петлях вигідніше використання комутуючих концентраторів замість більше дорогих комутаторів.

Маршрутизатори

Маршрутизатори Fibre Channel дозволяють підключити мережа Fibre Channel до іншого середовища передачі, наприклад до SCSI або Ethernet.

Адаптери Fibre Channel

Дотепер ми говорили про, так сказати, структуроутворюючих пристроях Fibre Channel. Однак найпоширенішими пристроями є, природно, адаптери Fibre Channel. Без них ніякий вузол не зміг би взаємодіяти з комутуючою структурою Fibre Channel. Ті самі адаптери можуть служити для з'єднання як з локальною мережею (іншими вузлами), так і з периферією. Це дозволяє, зокрема, скоротити число необхідних слотів вводу/виводу. Більшість адаптерів випускається для шини PCI. Часто разом з адаптерами використовуються 'гігабітні перехідники'

(GigaBit Interface Converter). Вони служать для перетворення оптичних сигналів в електричні й назад.

Класи сервісу

Комутатори й вузли можуть підтримувати один або більше видів сервісу. Загальні підтримувані комутаторами й вузлами сервіси визначаються під час процедури реєстрації пристроїв. Ручне налаштування при цьому не потрібно.

Клас 1 відповідає сервісу із установленням з'єднання й гарантованою доставкою. Виділене з'єднання через комутуючу структуру (сукупність комутаторів) установлюється за кілька мікросекунд. Оскільки з'єднання є виділеним, ніяке інший пристрій не може зв'язатися з портами одержувача й відправника, до його завершення. Пристрій-Одержувач підтверджує одержання кожного кадру устрою-передавачу. У такий спосіб гарантується доставка кадрів. Цей клас сервісу підходить для обміну більшими обсягами даних, зокрема для резервного копіювання.

Клас 2 іноді називають мультиплексним. При його використанні комутація кадрів виробляється незалежно друг від друга, тому кадри можуть доставлятися не в тому порядку, у якому були відправлені. З'єднання між пристроями не встановлюється. Доставка кадрів гарантується шляхом використання підтверджень. Цей вид сервісу схожий на організацію трафіку в локальних мережах.

Клас 3 аналогічний Класу 2, за винятком того, що він не використовує підтвердження одержання, тому доставка кадрів не гарантується. За рахунок цього реальна пропускна здатність збільшується. Щонайкраще цей клас сервісу підходить для багатоадресного і широкомовного розсилання.

Інші класи часто не виділяються в самостійні, а вважаються підвидами перерахованих. Вони відрізняються від перерахованих вище, наприклад, використанням не повної, а часткової пропускної здатності каналу.

Характеристики Fibre Channel

Fibre Channel дозволяє підтримувати самі різні швидкості – від 133 Кбіт/с до 4,252 Мбіт/с і навіть більше. Одна із цілей розробки Fibre Channel складалася, зокрема, у підтримці HIPPI на 100 Мбайт/с. Тому основною швидкістю передачі даних – так званою повною швидкістю – є 100 Мбайт/с (інші швидкості вказуються часто в частках від основної швидкості – одна восьма, четверта, друга, подвійна, учетверена). Однак, з урахуванням накладних витрат на кодування 8B/10B, заголовки кадрів і т.д., швидкість передачі властиво бітів становить 1,063 Мбіт/с. Таким чином, виробники приводять, як правило, дві швидкості – 'корисну', у байтах за секунду, і 'чисту', у бітах за секунду.

Як і в інших мережевих технологіях, підтримувані відстані й швидкості передачі залежать від типу використовуваного середовища передачі й генераторів сигналу. Fibre Channel може функціонувати як по оптичної, так і по мідному середовищу передачі.

Найбільші швидкості (до 4 Гбіт/с) і відстані (до 10 км) досягаються у випадку застосування одномодового оптичного волокна й низькочастотних лазерів. Багатомодове волокно здатне підтримувати такої ж швидкості, але на набагато менших відстанях, зокрема 100 Мбайт/с на відстанях до 500 м у випадку багатомодового волокна 50/125 мкм із високочастотним лазером. Мідне середовище передачі дозволяє підтримувати швидкості не вище основний на невеликих відстанях (100 м і менш).

3.2 Розробка структурної схеми

Дуже утрированно ідею Fibre Channel можна викласти так – аналог SCSI інтерфейсу для роботи з послідовних високошвидкісних каналів з можливістю комутації й маршрутизації потоків даних подібно звичайним Ethernet мережам і роботою на більших відстанях (до десятків кілометрів). Повторюємо, це дуже спрощене формулювання, але основну суть інтерфейсу Fibre Channel вона

відбиває. Часто багато з людей плутають Fibre Channel з оптичною реалізацією Ethernet, уважаючи що раз це оптика з такими ж оптичними кабелями, те й призначення те саме. Зрозуміло, це далеко жодне й те ж. Fibre Channel по суті поєднує високу швидкість SCSI інтерфейсу й переваги мережевої топології.

Технічні характеристики**:**

– Швидкість передачі даних – 1 Gbit/s, 2 Gbit/s, 4 Gbit/s і 8 Gbit/s. С обліком того, що для з'єднання пристроїв застосовуються два оптичних кабелі, кожний з яких працює в одному напрямку, при збалансованому наборі операцій запис/читання швидкість обміну даними подвоюється, тобто Fibre Channel працює в повнодуплексному режимі. У перерахуванні на мегабайтів паспортна швидкість Fibre Channel становить відповідно 100 MByte/s, 200 MByte/s, 400 MBytes/s, 800 Mbytes/s. Реально при 50% співвідношенні операцій запису/читання швидкість інтерфейсу досягає 200 MByte/s, 400 MByte/s і 800 MBytes/s. У майбутньому повинен з'явитися варіант на 16 Gbit/s або 1600 Mbyte/s. Найбільш популярні рішення Fibre Channel на 4 Gbit/s, оскільки вони мають краще співвідношення ціна/якість.

– Як і в SCSI інтерфейсі, протоколи сумісні – пристрій з FC-AL на 4 Gbit/s буде працювати з контролером на 1 Gbit/s і навпаки. Зрозуміло, швидкість передачі даних у подібній парі буде визначатися по самому повільному із пристроїв.

– Дальність роботи – до 300 метрів на оптичних багатомодових кабелях і повній швидкості інтерфейсу, до 10 кілометрів на одномодовом кабелі. Для переходу із багатомодового на одномодовий кабель потрібні спеціальні перетворювачі вартістю від $600 за пару.

– Реалізація фізичного каналу – або мідь, але тільки на швидкостях не вище 1 Gbit/s, або оптоволокно багатомодове 50/125 мкм і 62,5/125 мкм із з'єднувачами типу SC або, що частіше зустрічається, LC.

Топологія

Топологія Fibre Channel. Незважаючи на те, що формально Fibre Channel не є мережевим інтерфейсом, його топологія має багато схожа з мережевою топологією. Отже, топологія Fibre Channel може бути:

Arbitrated Loop (Петля з арбітражем, скорочено AL)

Це послідовне з'єднання пристроїв у кільце. Вихід одного пристрою з'єднується із входом наступного, його вихід у свою чергу із входом наступні й т.д. Усього в такій петлі може брати участь до 127 пристроїв. На **AL** топологію на момент створення стандарту покладали більші надії. Оскільки в Fibre Channel використовується абсолютна адресація пристрою, те більших втрат і затримок у петлі не виникає, при цьому **AL** дозволяє уникнути покупки дорогого Fibre Channel комутатора. Але з ростом швидкостей обміну даними **AL** стала негативно позначатися на продуктивності систем. До того ж в **AL** є два істотних недоліки: вихід з ладу хоча б одного пристрою приводить до відмови всієї системи й додавання/ виключення пристрою вимагає зупинки роботи всієї системи хоча б на короткий час.

Point-To-Point (Точка – точка)

У цьому варіанті робота з Fibre Channel практично нічого не відрізняється від SCSI, хіба що відстані між пристроями можуть бути на порядки більше.

Switched Fabric (Комутуєма структура, комутуєма матриця)

Сама популярна топологія Fibre Channel зараз і з високою часткою ймовірності в майбутньому.

Ця топологія близька до зрозумілої багатьом топології звичайної мережі – є комутатор і пристрої, до комутатора підключені. Незважаючи на подібність **Switched Fabric** на мережу, реально ця топологія функціонує трохи по іншому, про що піде мова нижче. Помітьте, що й назва говорить саме за себе – перемикається матриця, що, ніяк не асоціюється з хабом або маршрутизатором звичайного Ethernet. Сучасні Switched Fabric, такі як QLogic SANBox 5800 на 20 FC портів, мають агреговану смугу пропущення 544 Gbits/s, тобто 20 портів на

8+8 Gbits/s плюс смуга, необхідна для каскадування (підключення до інший Switched Fabric).

Види портів Fibre Channel**:**

– NL_port – (Node Loop – Вузол петлі) порт на пристрої, через який пристрій підключений до Arbitrated Loop (Петля з арбітражем).

– FL_port – (Fabric Loop – Порт у петлі) порт на зовнішньому пристрої, що з'єднується з ком утирується матрицей, що (fabric). Зрозуміло, топологія підключення в цьому випадку Arbitrated Loop (Петля з арбітражем).

– L_port (Loop port – Порт у петлі) просто термін, що поєднує й FL_port і NL_port.

– N_port (Node – Вузол) порт на пристрої, через який пристрій підключений по Point-To-Point (Точка – точка) топології.

– F_port (Fabric port – Порт матриці) порт на комутирується матрице, що, у топології Switched Fabric (Комутирується структура, що, що комутирується матриця).

– E_port (Expansion port – Порт розширення) порт, що з'єднує дві матриці, що комутуються між собою. Зв'язок, що з'єднує два E_port, звичайно називається ISL.

– TE_port (Trunking Expansion – З'єднання для розширення) – цей термін використовується для опису декількох портів, об'єднаних разом для збільшення смуги пропущення.

– G_port (Generic – Загальний, найпростіший) – порт, емулюючий F_port або E_port.

Безумовно, технічні дані про Fibre Channel цікаві самі по собі, але не зрозумівши практичного змісту у використанні Fibre Channel, не можна по достоїнству оцінити всі переваги цього чудового інтерфейсу. Тому ми плавно переходимо до прикладів практичної реалізації систем на Fibre Channel.

Отже, що ми маємо зараз у переважній більшості організацій? Залежно від розміру організації, кількості оброблюваних даних, кількості користувачів і т. П.

ми бачимо **n** серверів різного рівня й продуктивності, кожний зі своїм дисковим RAID масивом і всі обміни даними в організації здійснюються через локальну мережу. Згодом сервера утворять своєрідний зоопарк різних моделей, різної продуктивності й різних по можливостях. Під кожне нове завдання або для кращого рішення старої, як правило, купується новий сервер знову ж зі своїм дисковим масивом, потім він перестає справлятися зі своїми обов'язками, купується наступний сервер і процес повторюється.

Проблеми такий, можна сказати, класичної архітектури побудови мережі підприємства, зовсім очевидні:

– Кожний сервер коштує досить великих грошей, оскільки в складі сервера обов'язково присутній недешевий RAID масив.

– Перенос програмного забезпечення й даних зі старого сервера на новий трудомісткий і особливо складний у випадку неможливості зупинки старого сервера.

– Створення кластера (кластерів) вимагає покупки спеціального комплекту (комплектів) устаткування.

– Весь обмін даними в організації йде через локальну мережу. Які би гарні маршрутизатори не застосовувалися, з ростом обсягу даних мережа регулярно стає вузьким місцем і продуктивність всієї комп'ютерної системи організації знижується. Установка нового сервера (серверів) і нових комутаторів дозволяє якось "розшити" вузькі місця, але згодом, якщо організація збільшується або її обсяги ростуть, усе повторюється спочатку. До того ж реальна пропускна здатність навіть гігабітної мережі невелика, оскільки навіть досить дорогі мережеві комутатори не забезпечують сумарну смугу пропущення рівній сумі всіх потоків через комутатор.

– Неможливий перерозподіл ресурсів дискової пам'яті між серверами. Таке завдання рівносильне заміні цілком сервера, що, до речі, часто й робляться.

– Адміністрування системи, що складає з різномастого встаткування й не має програмних засобів для керування всією системою в цілому занадто залежить від конкретних людей.

– Сервера встановлюються в одній або декількох кімнатах, що при техногенній аварії (прорив води наприклад, або опалення, пожежі, нарешті) приводить до величезних втрат – і дані губляться й сервера виходять із ладу. Відновлення системи може зажадати масу часу й грошей.

Багато просунутих керівників IT підрозділів почали використовувати зовнішні SCSI to IDE/SATA системи зберігання даних. Це дозволяє "відв'язати" хоча б дискову пам'ять від конкретного сервера і якщо буде потреба міняти тільки один компонент, яким як правило, є сервер, обчислювальних можливостей якого часто не вистачає. У цьому варіанті можна купувати сервера тільки виходячи з їхніх обчислювальних можливостей і не платити щораз за великий убудований дисковий масив.

Безумовно, така архітектура більше зручна, ніж класична, але вона вирішує тільки малу частину проблем і не ліквідує головні п роблеми – високе навантаження на мережу й труднощі адміністрування системи. Тому далі ми розповімо про зовсім інший варіант побудови серверної системи підприємства.

Використання переваг Fibre Channel як високошвидкісної спеціалізованої мережі для передачі даних дозволяє принципово змінити архітектуру обчислювальної мережі організації. Fibre Channel дає можливість відокремити всі потоки даних між серверами підприємства, архівування даних і т. П. від локальної мережі користувачів.

У цьому варіанті можливості з конфігурування величезні – будь-який сервер може звертатися до будь-яким, дозволеним адміністратором системи дисковому ресурсу, можливий доступ до тому самому диска декількох пристроїв одночасно, причому з високою швидкістю, що не йде ні в яке порівняння зі швидкістю передачі даних по Ethernet. Не забувайте, що для кожного комп'ютера, підключеного до дискового ресурсу цей ресурс представляється як локальний. У

цьому варіанті й backup даних стає легким і прозорим завданням. У будь -який момент можна створити кластер, визволивши під нього ресурси на кожній з FC систем зберігання. Масштабування також досить наочно й зрозуміло – залежно від того, недостача яких можливостей виникла, можна або додати сервер, що буде куплений виходячи винятково з його обчислювальних можливостей, або додати нову систему зберігання.

Існує досить багато різного програмного забезпечення, яке дозволяє фактично управляти всіма системами зберігання як єдиним масивом + адміністрування, у результаті чого й виходить SAN (Storage Area Network). Зрозуміло, що в цій схемі є очевидний елемент ненадійності – єдина комутуєма матриця, для всіх серверів і систем зберігання. Цей недолік легко вбирається установкою дублюючої другої матриці. Завдяки продуманій архітектурі Fibre Channel, друга матриця може працювати "паралельно" з першої й у випадку виходу будь-якої матриці з ладу система в цілому цього просто не помітить.
Правильні системи зберігання, у свою чергу, мають 2 канали Fibre Channel (4 порти), що також уможливлює розпаралелювання процесів за допомогою другої матриці. На рисунку 3.1 ви можете бачити структурну схему системи у вигляді спрощеної схеми системи з дублюванням.

Не можна не розповісти ще про одну досить важливу й потрібну особливість Fibre Channel – можливості сегментування або, як прийнято в термінології Fibre Channel, зонування системи. Поділ на зони подібно поділу на віртуальні мережі (VLAN) у локальній мережі – пристрої, що перебувають у різних зонах, не можуть "бачити" один одного. Поділ на зони можливо або за допомогою комутирується матрици, що (Switched Fabric) або на основі вказівки адреси WWN (World Wide Name). Адреса WWN подібна MAC адресі в мережах Ethernet, кожний FC контролер має свою унікальну WWN адресу, що привласнює йому виробник, а будь-яка правильна система зберігання даних дозволяє ввести адреси тих контролерів або портів матриць, з якими цьому пристрою дозволений працювати. Поділ на зони призначено в першу чергу для підвищення безпеки й

продуктивності мереж зберігання даних. У відмінності від звичайної мережі, із зовнішнього миру не можна "проломити" зони й одержати доступ до закритого для даної зони пристрою.

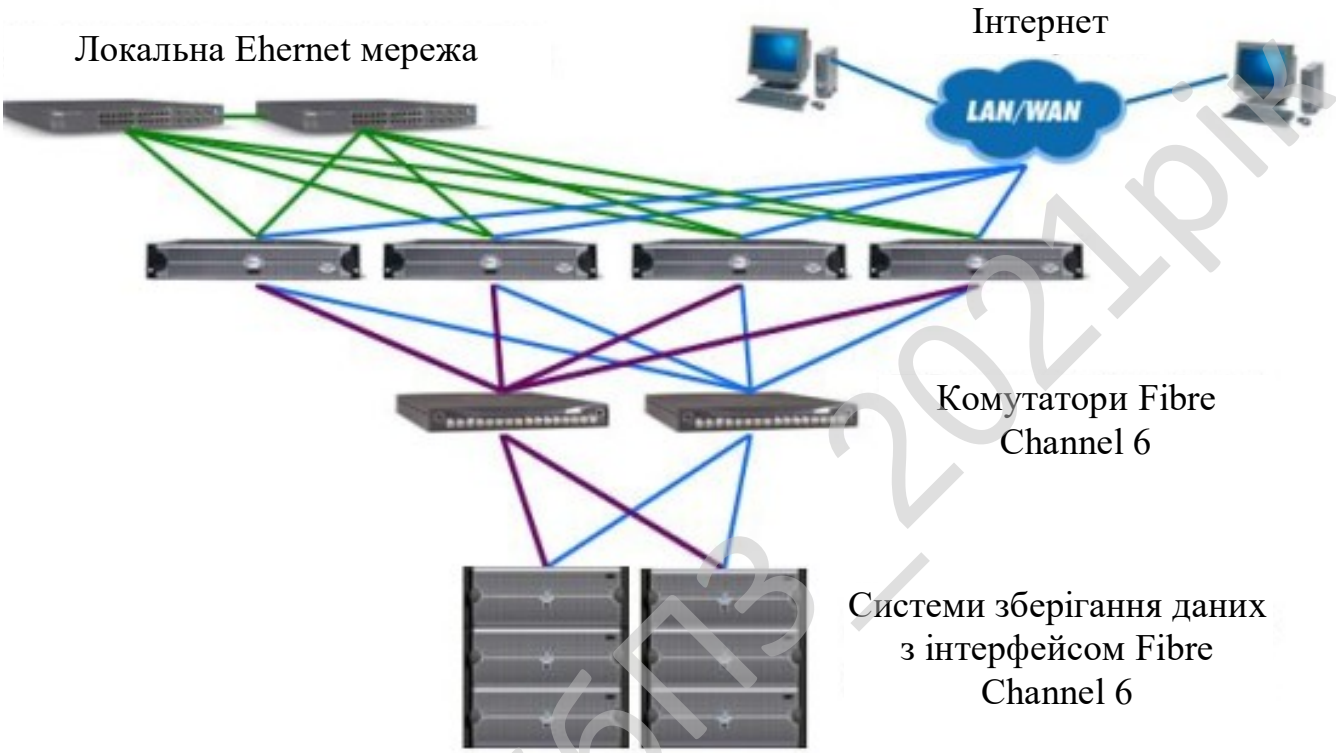

Рисунок 3.1 – Структурна схема системи

Зрозуміло, сфера застосування Fibre Channel не обмежується тільки серверними системами підприємств. Один із самих цікавих напрямків використання – робота з відео й цифровим кіно (Digital Cinema). Історично в Україні компанії, що працюють із кіно/відео матеріалом були одними з перших споживачів зовнішніх систем зберігання даних. Причина очевидна – кіно/відео матеріал вимагає величезних обсягів дискової пам'яті й без зовнішніх дискових масивів у багатьох випадках було просто неможливо працювати над проектами. Зараз в Україні є чимало кіно-, відео-компаній, у яких кількість систем зберігання даних обчислюється десятками.

Але, з ростом ринку кіно й телебачення в Україні зросла спеціалізація співробітників, що займаються комп'ютерною обробкою кіно- відео- матеріалів і виникла серйозна проблема – як обробляти матеріал послідовно декільком фахівцям? Або треба пересаджувати їх з однієї робочої станції на іншу й тим самим пропустити всіх необхідних людей через станцію, до якої підключений зовнішній дисковий масив з конкретним фільмом, або переганяти матеріал по мережі з одного масиву на інший, або (не варто сміятися, ми це бачили на власні очі), перетаскувати систему зберігання від одного комп'ютера до іншого. Система зберігання, до речі, може важити від 15 до 40 кілограмів. Всі ці способи досить помітно збільшують час роботи над проектом, оскільки навіть перекачування по мережі з однієї робочої станції на іншу терабайтів даних процес тривалий. Зрозуміло, що у відмінності від бізнес-застосунків, обробка відео по локальній мережі неможлива, занадто мала швидкість обміну даними.

Тому для компаній, що займаються post -production, застосування Fibre Channel є самим логічним виходом із цієї ситуації й ряд компаній уже успішно застосовують у своїй роботі Fibre Channel. Структура системи в цьому випадку не відрізняється від раніше розглянутої, просто замість серверів у систему входять робочі станції. У цьому варіанті всі проблеми відпадають – по завершенні чергового етапу інший співробітник починає займатися тим же самим фільмом, нікуди не переміщаючи файли й анітрошки не втрачаючи у швидкості доступу до даних. Використовуючи спеціалізоване програмне забезпечення, цілком можлива й одночасна робота декількох користувачів над одним проектом, що істотно прискорює процес. При обробці серіалів такі переваги Fibre Channel просто життєво необхідно використовувати.

Є ще одна серйозна причина для переходу на Fibre Channel – шум. Системи зберігання даних містять у собі, як правило, до 16 жорстких дисків, кілька блоків живлення й вентилятори охолодження. Все це технічна пишнота, на жаль, шумить і шумить помітно. Якщо для бізнес -застосунків це не настільки важливо, оскільки сервера й системи зберігання звичайно перебувають у

спеціальних приміщеннях, у яких персонал постійно відсутній, то для людей, що займаються творчістю з ранку до вечора шум є досить серйозною проблемою. Працюючи з SCSI системами зберігання, компанії йдуть на різні хитрування для ліквідації шуму. Наприклад, забирають робочі станції разом із системами зберігання в ізольовані приміщення, подовжуючи кабелі до миші, клавіатурі й монітору. Є й інші, не менш заморочливі варіанти. Застос ування ж Fibre Channel ліквідує проблему шуму як клас. Припустимого для FC інтерфейсу відстані в сотні метрів від комп'ютера з лишком вистачить для видалення системи зберігання від робочої станції в будь-якому будинку.

Купили, підключили, але як всім цим управляти?

Питання правильний і своєчасний. Отже, уявимо собі, що ми купили декілька Fibre Channel систем зберігання, підключили до нього кілька серверів у корпоративній системі або робітники станції для групової обробки відео в кінокомпанії. Що далі? Проблема в тому, що Fibre Channel дискова система зберігання з погляду комп'ютера є звичайний внутрішній локальний диск, підключений через звичайний дисковий контролер. Відповідно, якщо до цьомго FC диску буде мати доступ кілька комп'ютерів, то зовсім незрозумі ло, яким образом і хто буде стежити за тим, щоб один комп'ютер не записував у ті самі місця диска одночасно з іншим комп'ютером? Та й просто як одному комп'ютеру хоча б побачити, що хтось чужий таємничим образом змінив уміст його внутрішнього диска?

Рішення, а точніше спосіб не рішення цього завдання істотно залежить від сфери застосування**.**

Бізнес

Найчастіше в бізнес застосуваннях не потрібний поділ одного локального дискового простору між декількома комп'ютерами або такий поділ штатно підтримується операційною системою (типовий приклад – кластер). У багатьох бізнес застосуваннях адміністратори цілком задовольняються гнучкими можливостями Fibre Channel систем по перерозподілі дискового простору між

комп'ютерами, легкістю конфігурування, можливостям по масштабуванню й т. П. Тому в бізнес застосуваннях можна обмежитися штатними можливостями адміністрування, які надають як Fibre Channel матриці, так і самі системи зберігання.

Цифровий кінематограф і телебачення

Тут теж часто застосовується простий підхід – віртуальна "перекидання" дисків від одного користувача до іншого залежно від етапів роботи над проектом, або елементарне тверде закріплення певного дискового простору за конкретним комп'ютером (користувачем). До того ж кваліфікація системних адміністраторів у кіно- відео- компаніях найчастіше буває набагато нижче, ніж у великих комерційних структурах, банках і т. П. і саме по собі застосування Fibre Channel у кіно- відео- багатьом їхнім керівникам і технічному персоналу здається, на жаль, непотрібним. На щастя, останнім часом Fibre Channel стає усе більше й більше популярним навіть у невеликих студіях.

Зрозуміло, варіанти "вижимання" з Fibre Channel систем усього, що вони вміють і можуть, є, але проблема в масовому незнанні цих можливостей. Якщо перебільшувати проблему, то її можна сформулювати в такий спосіб – як, зберігаючи величезні швидкості обміну даними, можливості гнучкого апаратного налаштування, масштабування й т. П. управляти SAN як локальною мережею?

Програма дозволяє набудовувати й управляти практично всіма можливостями SAN, додаючи до них такі корисні речі як перемикання трафіку на LAN у випадку обриву кабелю або виходу з ладу FC контролера, керування смугою пропущення й т. П. П ереваги – великі можливості з адміністрування, відсутність виділеного сервера метаданих, гарний захист від збоїв, немає прив'язки до конкретних типів матриць – систем зберігання – FC контролерів. До переваг варто віднести підтримку крім Windows також Mac OS і Linux Red Hat. Недоліки – відносно висока вартість рішення, складова $1100 на комп'ютер, підключений до SAN.

Ціна питання

Ще кілька років тому назад Fibre Channel могли застосовувати тільки дуже небідні організації. Комутатори (Switched Fabric) коштували до десятків тисяч доларів, контролери по кілька тисяч. Ціни на системи зберігання теж не радували – FC системи зберігання коштували на $ 1500-2000 більше, ніж точно такої ж системи з SCSI інтерфейсом. Але останнім часом, завдяки популярності, що розширюється, Fibre Channel ціни стали помітно знижуватися. На весну 2020 року ситуація така:

- Контролери PCI Express – $900 за порт.
- Комутатори (Switched Fabric) – $450 за порт
- Системи зберігання – на $400 дорожче аналогічні з SAS інтерфейсом.

Недешево, скаже потенційний покупець, який придивлявся до можливостей Fibre Channel. На перший погляд це дійсно так. Але, за рахунок зовсім іншої конфігурації системи застосування Fibre Channel дозволить заощадити значні суми на відмові від застосування в серверах дорогих RAID контролерів і дисків, на значно менші витрати на розширення системи і її адміністрування. Звичайно, застосування Fibre Channel для двох-трьох серверів або робочих станцій економічно невиправдано – але для більших або систем, що розвиваються, їсти зміст відразу використовувати Fibre Channel, оскільки це дасть істотну економію при масштабуванні

3.3 Розробка функціональної схеми

Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, – це операційна система для керування файловими й блоковими системами зберігання даних, що містить функції для створення мережевих сховищ (NAS), а також мереж зберігання даних (SAN) на основі iSCSI, InfiniBand і Fibre Channel.

Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, забезпечує найвищу продуктивність, безпеку й масштабованість, а її вартість значно нижче, ніж в альтернативних рішень зберігання даних. До числа розширених функцій програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6, відноситься реплікація даних і томів, створення моментальних знімків томів, автоматичне перемикання на інші iSCSI-томи, підтримка технології однократного запису WORM і протоколу NDMP. Крім того, програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує самі різні апаратні RAID-платформи від провідних представників галузі. Нарешті, програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, винятково багатофункціональна: вона не тільки дозволяє користувачам сполучати мережеві сховища з мережами iSCSI SAN, але й містить убудовані засоби антивірусного захисту, а також підтримує підключення самих різних мережевих плат, включаючи Gigabit Ethernet, 10 Gigabit Ethernet, Fibre Channel і InfiniBand. Для зручності доступу програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, можна завантажити на будь-який апаратний пристрій.

Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з NAS і iSCSI, а також цілий ряд додаткових функцій:

– установка на будь-який носій: убудований жорсткий диск, RAID-масив, диски USB DOM, SATA DOM і IDE DOM і будь-які інші завантажувальні носії;

– поліпшений графічний інтерфейс користувача;

– агент резервного копіювання Symantec Backup Exec (RALUS) з підтримкою версій 11, 12 і 12.5;

– базові налаштування через API-інтерфейс;

– підвищена швидкість виконання довільних операцій читання й записи iSCSI;

– підтримка функцій кластеризації Windows Server 2008 (постійне резервування SCSI-3);

– підтримка апаратних RAID;

– перехід при відмові iSCSI;

– реплікація томів і даних;

– знімки томів;

– ініціатор iSCSI / кінцевий пристрій iSCSI;

– функції резервного копіювання;

– підтримка протоколу NDMP 3.0;

– підтримка технології однократного запису WORM;

– тіньове копіювання ОС для відновлення системи;

– підтримка декількох мережевих карт (NIC);

– об'єднання / створення груп мереж;

– механізм завдань і запуск за розкладом;

– підтримка мережевих ИБП;

– підтримка Fibre Channel;

– підтримка багатопроцесорних систем;

– підтримка IP-SEC.

Вимоги до апаратного забезпечення мінімальні:

– ПК, сумісний з архітектурою x86.

– 2 GB RAM (рекомендується 4 GB RAM).

– ЦП (рекомендується частота, що, – 1 ГГц або вище).

– Жорсткий диск (SATA, SAS, SCSI, ATA) / RAID / Fibre Channel / iSCSI.

– Мережева карта (для використання функції об'єднання рекомендується кілька мережевих карт 1 Гб/с).

Функціонально програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6 складається з наступних блоків:

– Інтерфейс адміністрування.

– Мережеві функції.

– Керування системами зберігання даних.

– Моніторинг.

– Підтримка апаратного забезпечення.

– Функції для роботи з мережевими сховищами (NAS).

– Функції для роботи з iSCSI.

– Функції резервного копіювання.

– Інші функції.

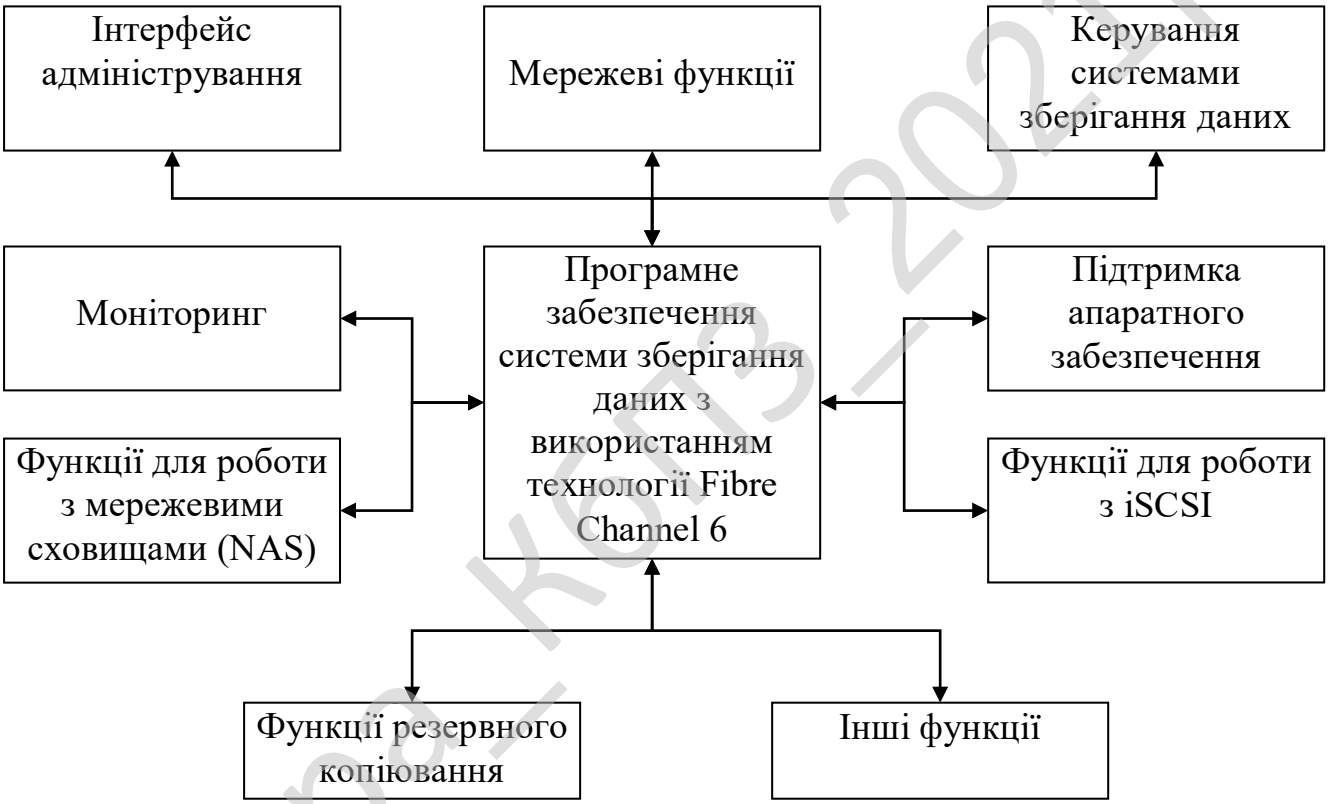

Рисунок 3.2 – Функціональна схема системи

Розглянемо основні характеристики й особливості програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6.

Інтерфейс адміністрування

– Потужний, інтуїтивно зрозумілий веб-інтерфейс. Веб-интерфейс програмного забезпечення системи зберігання даних з використанням технології

Fibre Channel 6, спрощує віддалене адміністрування пристроїв зберігання даних і підтримує керування системою з будь-якого комп'ютера в мережі.

– Безпечний доступ адміністратора. Для налаштування програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6, і роботи з рішенням використовується веб-інтерфейс, доступ до якого захищений паролем і здійснюється по безпечному протоколі SSL.

– Інструменти консолі. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримується керування за допомогою інструментів консолі. З метою безпеки деякі найважливіші функції адміністрування (наприклад, видалення й відновлення томів або перевірка й виправлення файлової системи) доступні тільки при роботі в консолі. За допомогою інструментів консолі адміністратор відслідковує стан мережі й рівень використання пам'яті, установлює драйвери й устаткування й виконує безліч інших завдань.

– Віддалений доступ до консолі. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує віддалений доступ до інструментів консолі, що дозволяє управляти сервером з будь-якого комп'ютера в мережі.

– Засоби налаштування. Засоби налаштування дозволяють змінювати параметри дисків, мережі й файлової системи, щоб підвищити продуктивність сервера програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6.

– Багаторівневе керування. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримується трирівнева система прав доступу: повний доступ, обслуговування й адміністрування.

– Відновлення системи (локально або через Інтернет). Нова функція відновлення системи дає можливість перезавантажитися зі старої системи відразу в нову версію, що скорочує тривалість простою.

Мережеві функції

– Диспетчер статичних маршрутів. У новій версії рішення адміністратор може визначати пріоритет правил маршрутизації, змінивши їхній порядок у списку. Ця функція буває необхідна в тих випадках, коли в одній підмережі використовується кілька мережевих інтерфейсів.

– Підтримка стандартного мережевого шлюзу. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, може використовуватися в структурованих корпоративних мережах і підтримує доступ до даних через Інтернет і інтранет.

– DHCP-клієнт. Доступний підтримка протоколу DHCP, що забезпечує адміністрування й автоматизацію призначення IP-адрес.

– Підтримка декількох мережевих карт (NIC). Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з декількома мережевими картами для підключення до різних підмереж або для підвищення пропускної здатності. Крім того, адміністратор може вибирати, які сервіси будуть включені на тім або іншому мережевому інтерфейсі.

– Відказостійкість адаптерів. Функція відказостійкості адаптерів (AFT) підвищує надійність системи. У випадку збою в основному мережевому адаптері підключення автоматично переводиться на вторинний.

– Адаптивне балансування навантаження. Якщо вимоги застосунків міняються, функція адаптивного балансування навантаження (ALB) автоматично перенаправляє дані по іншому маршруті, що підвищує пропускну здатність мережі.

– Інтерфейс 10 Gb Ethernet з механізмом розвантаження TCP/IP. Програмне забезпечення підтримує інтерфейси 10 Gb, які мають підвищену пропускну здатність, збільшують загальну продуктивність системи й скорочують затримку при передачі повідомлень між підключеннями. Крім того, програмне забезпечення системи зберігання даних з використанням технології Fibre

Channel 6, працює з деякими мережевими картами 10Gb з механізмом розвантаження TCP/IP.

– Підтримка інтерфейсу InfiniBand. В програмному забезпеченні підтримуються плати InfiniBand, які мають підвищену пропускну здатність, збільшують загальну продуктивність системи й скорочують затримку при передачі повідомлень між підключеннями.

– Налаштування проксі-сервера. Програмне забезпечення дозволяє набудовувати проксі-сервери для підключення до Інтернету.

– Робота із протоколом IPsec. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з безпечного протоколу IPsec, що запобігає можливість перехоплення або модифікації конфіденційних даних при передачі по мережі.

– Налаштування шлюзу за замовчуванням. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, може працювати з декількома мережевими інтерфейсами й дозволяє вибирати шлюз за замовчуванням.

– Статична маршрутизація. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, доступно потужний засіб налаштування статичних маршрутів, що дозволяє встановлювати налаштування маршруту за замовчуванням.

Керування системами зберігання даних:

– Автоматичний перехід при відмові томів NAS із протоколом NFS (опціонально).Для підвищення відказостійкості системи програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує синхронну реплікацію томів NAS для загальних ресурсів NFS. Якщо ця функція встановлена, то одночасно із записом даних на основний сервер система автоматично створює їхню дзеркальну копію на вторинному сервері. У випадку збою основного сервера програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, автоматично переводить всі завдання

на інший і перепризначує йому всі блокування NFS. Для використання цієї функції необхідно придбати пакет додаткових компонентів.

– Можливість динамічного перемикання між ініціатором і цільовим об'єктом FC у платах QLogic 2/4/8 Гбіт/с без перезапуску системи. Режими портів можна перемикати в динамічному режимі.Тепер режим плати FC QLogic можна перемикати скільки завгодно раз (з ініціатора в цільовий об'єкт і назад) без перезапуску сервера. Це безцінна можливість для тих, хто впроваджує у своєму середовищі програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, і планує випробувати різні режими ініціатора / цільового об'єкта. Крім того, режим можна міняти не тільки для самого адаптера, але й для окремих портів (якщо використовується багатопортова мережева плата).

– Підтримка автоматичного переприсвоєння сеансів (ASR) для протоколів Fibre Channel і iSCSI. Автоматичне переприсвоєння сеансів дозволяє перейменовувати, видаляти групи й привласнювати WWN-ідентифікатори без переривання (блокування) інших сеансів. Така функціональність забезпечує безперебійну роботу сервісів iSCSI і Fibre Channel під час зміни їхніх конфігурацій.

– Додаткові параметри для роботи із загальними ресурсами Samba. Операційна система підтримує параметри «Спадкування власника», «Спадкування контролю доступу», «Спадкування прав», «Зіставлення спадкування контролю доступу», «Блокування».

– Перехід при відмові iSCSI. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, забезпечує відказостійкість систем шляхом реплікації томів iSCSI. Копіювання даних на основний сервер здійснюється в режимі реального часу, при цьому кожна зміна негайно відбивається на допоміжному сервері. У випадку збою основного сервера програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, автоматично перемикається на роботу з допоміжним сервером.

– Перехресна реплікація даних. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує синхронізацію файлів і каталогів з одного NAS-серверу на інший. При аварійному відновленні або синхронізації резервних копій з диска на диск використовується блокове перенесення даних, що дозволяє зменшити навантаження на мережу. Синхронізація даних може виконуватися в обох напрямках: при переносі файлів система може одночасно бути як джерелом, так і об'єктом призначення, забезпечуючи перехресне копіювання декількох серверів.

– Постійне резервування SCSI-3. Підтримка команд постійного резервування. Ця можливість забезпечує підтримку Windows 2008 і кластерів на базі Windows.

– Апаратний і програмний ініціатор iSCSI. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, не тільки підтримує роботу з апаратними ініціаторами iSCSI, але й містить власний програмний ініціатор, що спрощує збільшення ємності сховища на NAS-сервері. Для додавання нових пристроїв і логічних томів досить підключити систему зберігання даних iSCSI до NAS-сервера.

– Підтримка апаратних і програмних RAID-масивів. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує контролери SAS, SCSI, SATA і IDE всіх ведучих виробників апаратних RAID-контролерів. Убудована підтримка програмних RAID-масивів рівня 0, 1, 5, 6 забезпечує додаткову гнучкість і економічність при створенні рішень зберігання даних. Дзеркалюваня двох апаратних RAID-масивів дозволяє знизити ймовірність збою. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, забезпечує максимальну гнучкість із погляду резервування, продуктивності, економічності й обсягу системи зберігання даних.

– Критичний режим програмних RAID-масивів. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, має засоби,

які дозволяють створювати RAID-масиви навіть при відсутності або поломці одного з жорстких дисків.

– Відправлення по електронній пошті повідомлень про неполадки RAID. Якщо в RAID виникають неполадки, програмне забезпечення DSS висилає системному адміністраторові повідомлення по електронній пошті.

– Підтримка технології S.M.A.R.T. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, доступна підтримка технології автоматичного моніторингу, аналізу й відправлення звітів S.M.A.R.T. Вона дозволяє виявляти можливі збої жорстких дисків і відправляти звіти про їхнє стояння.

– Відправлення повідомлень S.M.A.R.T. По електронній пошті. Якщо з жорсткими дисками виникають проблеми, програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, висилає системному адміністраторові повідомлення по електронній пошті.

– Підтримка декількох HBA-адаптерів Fibre Channel. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримуються HBA-адаптери Fibre Channel компаній Emulex, Qlogic і LSI, які дозволяють підвищити пропускну здатність і продуктивність мережі, а також знизити рівень затримки. Fibre Channel – це стандартний тип підключення високошвидкісних пристроїв зберігання до комп'ютерів у мережах зберігання даних (SAN).

– Підтримка апаратних RAID-контролерів. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує більшість апаратних RAID-контролерів ATA / SATA / SCSI.

– Моментальні знімки. Моментальний знімок – це образ логічного тому на певний момент часу. Образ моментального знімка може використовуватися як для цілісного, так і для тимчасового резервного копіювання; при цьому для користувачів зберігається повний і безперебійний доступ до логічного того. Якщо

файл буде випадково віддалений або змінений, його можна відновити з раніше створеного знімка.

– Розклад створення моментальних знімків. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримується функція автоматичного створення моментальних знімків за розкладом, у заздалегідь заданий час (наприклад, щогодини).

– Робота з декількома логічними томами і їхніми групами. Функція об'єднання логічних томів у групи дозволяє адміністраторам сховищ створювати пули логічних СЗД, у які входить кілька фізичних дисків. Завдяки тому, що програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з більшою кількістю груп логічних томів, адміністраторам стає простіше дотримувати вимог користувачів і застосунків – і при цьому максимально ефективно задіяти доступний фізичний простір на дисках.

– Інтерактивне розширення логічних томів. При необхідності адміністратор може збільшувати ємність логічних томів без перезапуску застосунків, повторно тому, а також виконувати резервне копіювання й відновлення даних на тім.

– Підтримка інтерактивного розширення ємності. Функція розширення апаратних RAID-контролерів дозволяє збільшувати розмір існуючих пристроїв без переміщення даних.

Моніторинг

– Моніторинг апаратного забезпечення. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує моніторинг стану встаткування, зокрема відслідковує дані про температуру, напругу живлення й швидкості обертання вентилятора, які передаються датчиками з материнської плати.

– Підтримка протоколу SNMP v2 і v3. Протокол SNMP використовується в програмному забезпеченні системи зберігання даних з використанням

технології Fibre Channel 6, для відстеження таких параметрів, як швидкість передачі даних і коефіцієнт завантаження процесора й оперативної пам'яті.

– Повідомлення по електронній пошті. Якщо в пристрої зберігання даних виникає технічний збій, адміністраторові по електронній пошті висилається повідомлення про це.

– Ведення журналів. Файл журналу – це ефективний інструмент, що допомагає при аналізі й усуненні технічних проблем.

– Перевірка системи. Спеціальна функція дозволяє перевіряти продуктивність системи.

Підтримка апаратного забезпечення:

– Підтримка 8 і більше HBA -адаптерів Qlogic у режимі кінцевого пристрою. Нова версія рішення дозволяє підключати більше 8 HBA-адаптерів Qlogic у режимі кінцевого пристрою. Це дозволяє підключати більше дисків, збільшуючи кількість ресурсів, які доступні на одному комп'ютері.

– Підтримка технології SSD-кешування Adaptec MaxIQ. SSD-кеш Adaptec MaxIQ являє собою програмно-апаратне рішення, що дозволяє серверу зберігати часто використовувані дані на швидкому твердотільному накопичувачі. Це потужне рішення кешування здатне прискорювати роботу системи зберігання даних в 8 у порівнянні з рішеннями на базі звичайних жорстких дисків. Воно може використовуватися з будь-яким контролером зберігання даних Adaptec серії 5Z, 5 або 2. Тепер програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, забезпечує підтримку цієї технології. Потрібно лише придбати SSD-кеш Adaptec MaxIQ і підключити вхідний у цей комплект твердотільний накопичувач до свого контролера зберігання даних. Рішення DSS буде працювати з ним автоматично.

– Розширник в убудованому диспетчері RAID-масивів LSI. Диспетчер RAID-масивів LSI, доступний у програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримує функцію розширника. Завдяки цьому в інтерфейсі програмного забезпечення системи

зберігання даних з використанням технології Fibre Channel 6, доступна функція збільшення кількості ресурсів і керування ними.

– Підтримка до 32 процесорів. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, може підтримувати роботу з 32 процесорами – це підвищує пропускну здатність при обробці більших обсягів даних.

– Підтримка переривань MSI. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує переривання MSI – ця функція дозволяє оптимізувати роботу з останніми моделями PCI-карт.

– Підтримка джерел безперебійного живлення. Джерела безперебійного живлення запобігають наслідку раптового відключення електрики й дозволяють завершити роботу комп'ютера без втрати даних. Ці пристрої підключаються до сервера через COM-порт або USB-порт.

– Підтримка мережевих ІБЖ. По протоколі SNMP програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, може обмінюватися даними з іншими серверами (наприклад, з іншою системою DSS). Це забезпечує безпечне відключення у випадку збою живлення: сервер з ІБЖ через мережу відправляє іншим серверам у мережі (у підлеглому режимі) сигнал про збій живлення й відключає їх.

Функції для роботи з мережевими сховищами (NAS)

– Підтримка Windows Active Directory / основного контролера домена. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з Windows Active Directory (AD), основним контролером домена й протоколом LDAP, а також синхронізацію імен користувачів і груп між доменом AD і службою NIS. Завдяки цьому йому доступні відомості про користувачів, групи, системи й інші ресурси, які зберігаються в AD. Підтримка списків контролю доступу дозволяє автоматично одержувати з домена інформацію про повноваження користувачів.

– Підтримка служби NIS. Завдяки взаємодії зі службою каталогів NIS програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, дозволяє без праці витягати всю інформацію, що розташовується на будь-якому іншому сервері зберігання даних. Ця служба відповідає за поширення даних про конфігурацію систем (наприклад, про імена користувачів і хостів) між всіма комп'ютерами в мережі.

– Журналюєма файлова система. Функція журналювання робить файлову систему більше надійної й забезпечує швидке й ефективне відновлення після непередбаченого завершення роботи.

– Контроль квот для користувачів і груп. Для контролю роботи користувачів з високими вимогами до ємності сховища адміністратор може привласнювати їм виділені загальні ресурси або обмежувати доступне їм дисковий простір за допомогою квот. Такі квоти можуть виділятися як окремим користувачам, так і групам, у які вони входять.

– Антивірусний захист (сканування загальних ресурсів на сервері) У рішенні програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6, доступна убудована система антивірусного захисту, що виконує сканування загальних ресурсів у заздалегідь зазначений час. Система дозволяє обновляти базу визначень вірусів, що може розташовуватися на мережевому пристрої зберігання.

– Антивірус (онлайн-сканування). Програмне забезпечення також підтримує онлайн-сканування файлів, які передаються по протоколах SMB і FTP.

– Підтримка запам'ятовувальних пристроїв USB як динамічні диски. Якщо підключені USB-модулі, вони можуть використовуватися як динамічні диски – наприклад, для створення локальних резервних копій.

– Агенти резервного копіювання Veritas, EMC Dantz, CA BrightStor. Агенти резервного копіювання Veritas, Dantz і CA BrightStor ARCserve допомагають адміністраторам більш ефективно захищати інформацію – вони поліпшують передачу даних, підвищують безпеку мережі й забезпечують

перегляд каталогів у режимі реального часу, використання завдань моніторингу й підтримку крос-платформного резервного копіювання.

– Підтримка мережевих клієнтів і протоколу NFS. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримуються протоколи передачі файлів CIFS/SMB, NFS, FTP, FTPS, SFTP і Apple Talk – це означає, що той самий сервер може працювати із клієнтами Windows, Linux, Unix і Macintosh.

– Протокол LDAP. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, може працювати із зовнішніми системами по протоколі LDAP. Завдяки цьому адміністратор може створити кілька систем і працювати з ними з одного комп'ютера (із системи з базою даних LDAP або з будь-якого іншого сервера LDAP).

– Можливість роботи суперкористувача. Облікового запису суперкористувача надається необмежений доступ до всіх файлів і томів. Ця можливість буває необхідна в тих випадках, коли частина каталогів або файлів стає недоступна.

Функції для роботи з iSCSI

– Робота декількох користувачів CHAP з одним кінцевим пристроєм. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, користувачам CHAP можна привласнювати певні кінцеві пристрої iSCSI. Протокол CHAP – це спеціальна схема автентифікації за іменем користувача й паролем, що дозволяє перевіряти особистість користувача й контролювати його права доступу до кінцевих пристроїв.

– Підтримка багатошляхового вводу-виводу (MPIO). Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, дозволяє встановлювати кілька підключень до одного кінцевого пристрою – ця можливість підвищує продуктивність і надійність системи.

– Захист кінцевих пристроїв паролем. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, можна

встановлювати пароль на підключення до кінцевих пристроїв iSCSI, щоб захистити їх від несанкціонованого доступу.

– Логічні пристрої тільки для читання. Ця функція дозволяє перемикати логічні пристрої в режим «Тільки читання» (ця функція може використовуватися тільки з кінцевими пристроями iSCSI).

Функції резервного копіювання

– Механізм завдань і запуск за розкладом. Уніфікований механізм завдань із функцією запуску за розкладом спрощує реплікацію даних і томів, створення резервних копій і моментальних знімків. При цьому система дозволяє створювати кілька розкладів для запуску кожного завдання.

– Віртуальні стрічкові накопичувачі. Система дозволяє настроїти емуляцію стрічкового накопичувача на основі загального ресурсу або динамічного тому (наприклад, знімного жорсткого диска). Резервне копіювання на віртуальні стрічкові накопичувачі виконується точно так само, як і на фізичні.

– Технологія WORM. Технологія однократного запису WORM застосовується при архівуванні даних.

– Підтримка стрічкових бібліотек. Бібліотеки стрічкових накопичувачів дають можливість автоматизувати резервне копіювання даних.

– Убудована база даних. У системі зберігається інформація про всі файли резервних копій, включаючи журнал резервного копіювання.

– Керування строком зберігання стрічок. Рішення дозволяє встановлювати розкладу, по якому замість старих резервних копій на стрічкові накопичувачі будуть записуватися нові.

– Швидка первісна синхронізація томів. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримується швидка первісна синхронізація для реплікації томів.

– Точне налаштування DRBD. Функція точного налаштування розподілених реплікуємих блокових пристроїв (DRBD) дозволяє прискорити процес реплікації на тім же встаткуванні.

Інші функції

– Підтримка файлів підкачування. Файл підкачування – це спеціальна область жорсткого диска, що разом з ОЗП використовується для зберігання вмісту пам'яті. Якщо програмному забезпеченню DSS потрібна додаткова пам'ять (наприклад, для підготовки файлової системи після роботи в критичному режимі RAID), воно зберігає дані у файл підкачування.

– Підтримка контролера домену Windows 2008. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує роботу з ОС Windows 2008 як контролер домену. Якщо в середовищі використовуються служби Active Directory, доступні в Windows 2008, то програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, зможе підключатися до них і використовувати їхні можливості.

– Підтримка РК-панелей. У програмному забезпеченні системи зберігання даних з використанням технології Fibre Channel 6, підтримуються різні конфігурації РК-дисплеїв: один, два або кілька моніторів, а також пірамідальна конфігурація.

– Окремі ліцензії на кожний пристрій зберігання. Схема ліцензування має на увазі, що на кожний екземпляр здобувається окрема ліцензія. Поза залежністю від того, скільки користувачів працює із пристроєм зберігання – п'ять, п'ятсот, п'ятнадцять тисяч або навіть більше – вартість ліцензії буде незмінною.

– Локальні резервні копії динамічних пристроїв зберігання. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, дозволяє визначити жорсткий диск або USB-модуль як динамічний пристрій. У цьому випадку робота з ним буде вестися точно так само, як зі звичайним стрічковим накопичувачем – на нього можна зберегти резервну копію NAS-сервера, а потім витягти його без відключення системи.

– Перехресна синхронізація даних. Програмне забезпечення системи зберігання даних з використанням технології Fibre Channel 6, підтримує синхронізацію файлів і каталогів з одного NAS-сервера на іншій. При аварійному

відновленні або синхронізації резервних копій з диска на диск використовується блокове перенесення даних, що дозволяє зменшити навантаження на мережу. Синхронізація даних може виконуватися в обох напрямках: при переносі файлів система може одночасно бути як джерелом, так і об'єктом призначення, забезпечуючи перехресне копіювання декількох серверів.

– Відключення граничного розміру сховища при ліцензуванні 48 ТБ дискової ємності. Якщо ліцензована ємність сховища досягає 48 ТБ, то всі обмеження на максимальний розмір знімаються. Ви можете встановити будь-яку версію програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6, а потім придбати додаткові ліцензійні ключі. Як тільки ємність досягне 48 ТБ, вам буде доступний максимальний обсяг сховища, що підтримується цим рішенням.

– Майстер налаштування. При першому запуску програмного забезпечення системи зберігання даних з використанням технології Fibre Channel 6, система відкриває спеціальний майстер, що допоможе вам настроїти мову, мережа й системний час.

– API-інтерфейс. При роботі із сервером можна обійтися навіть без веб-браузера, оскільки всі операції виконуються через безпечне підключення по протоколі SSH. В API-інтерфейсі підтримуються наступні функції: – робота з користувачами й загальними ресурсами; – робота з кінцевими пристроями iSCSI; – стан устаткування; – відновлення ПЗ; – перезавантаження й відключення комп'ютера.

– Віддалена підтримка. Наша служба підтримки може звертатися до системи DSS у віддаленому режимі. Ця можливість дозволяє значно прискорити усунення проблем. З метою безпеки ця функція вручну ініціалізується адміністратором.

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників.

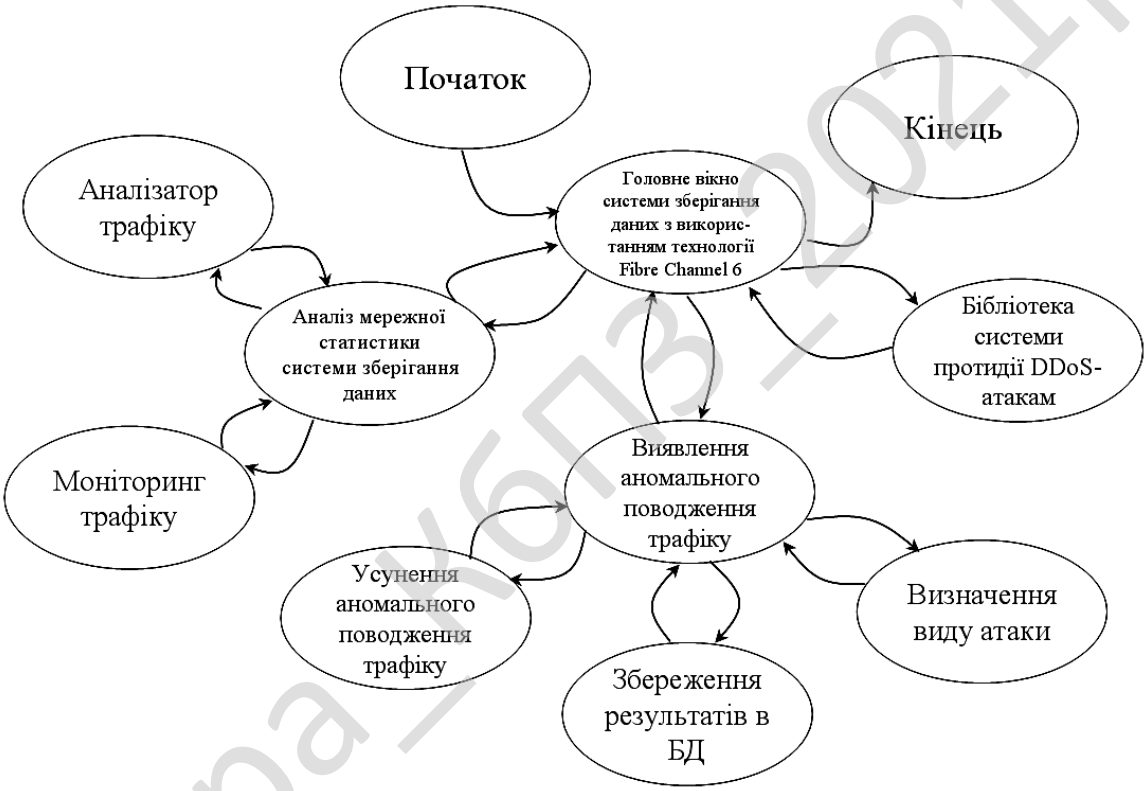

Рисунок 3.3 – Діаграма взаємодії процесів

Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

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.

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

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію магістерської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Було створено блок-схеми роботи системи. 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.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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

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

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

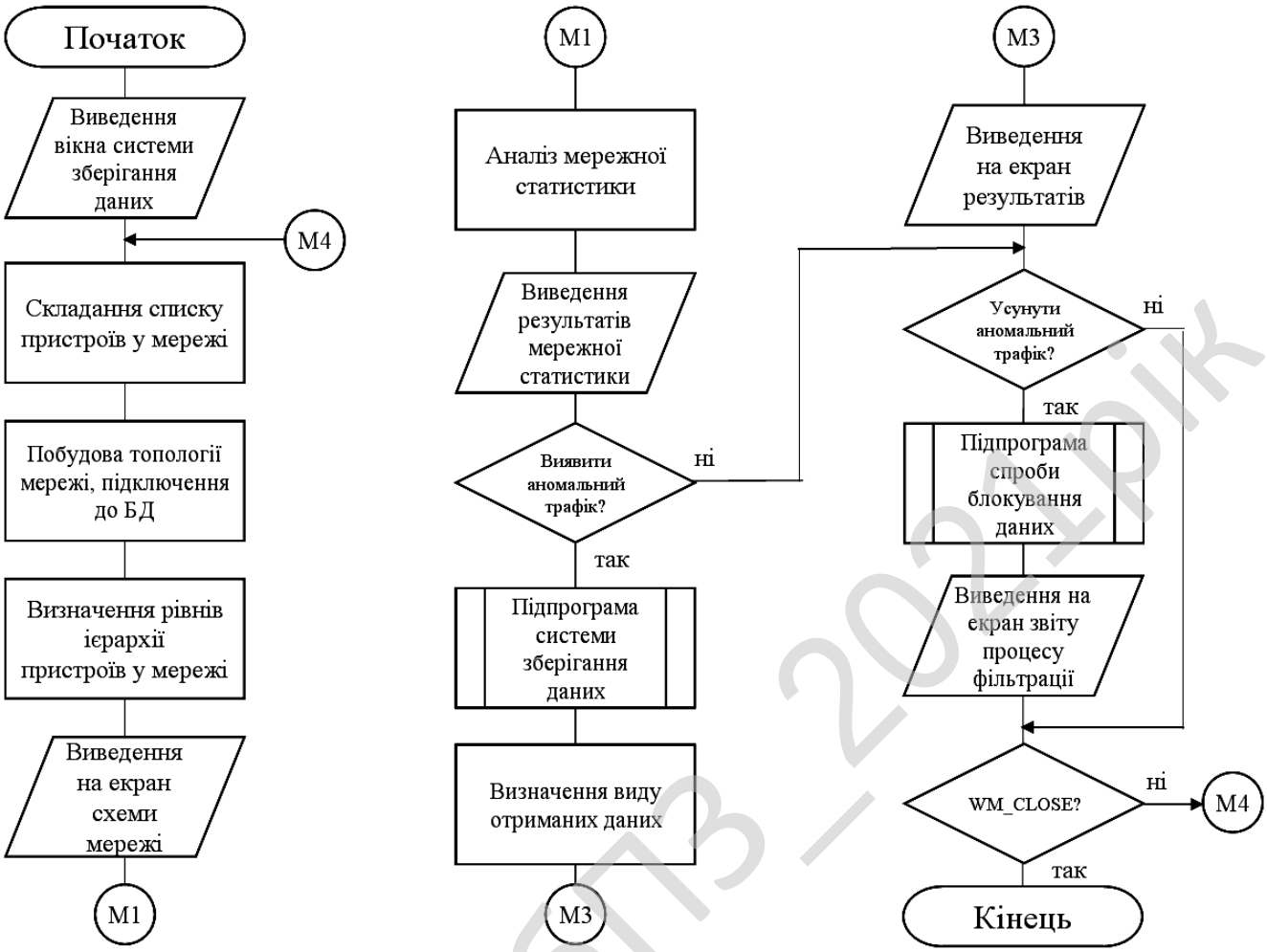

Рисунок 4.1 – Блок-схема основної програми

Було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів; Діаграма компонент; Діаграма об'єктів; Діаграма розгортання.

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

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням

тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

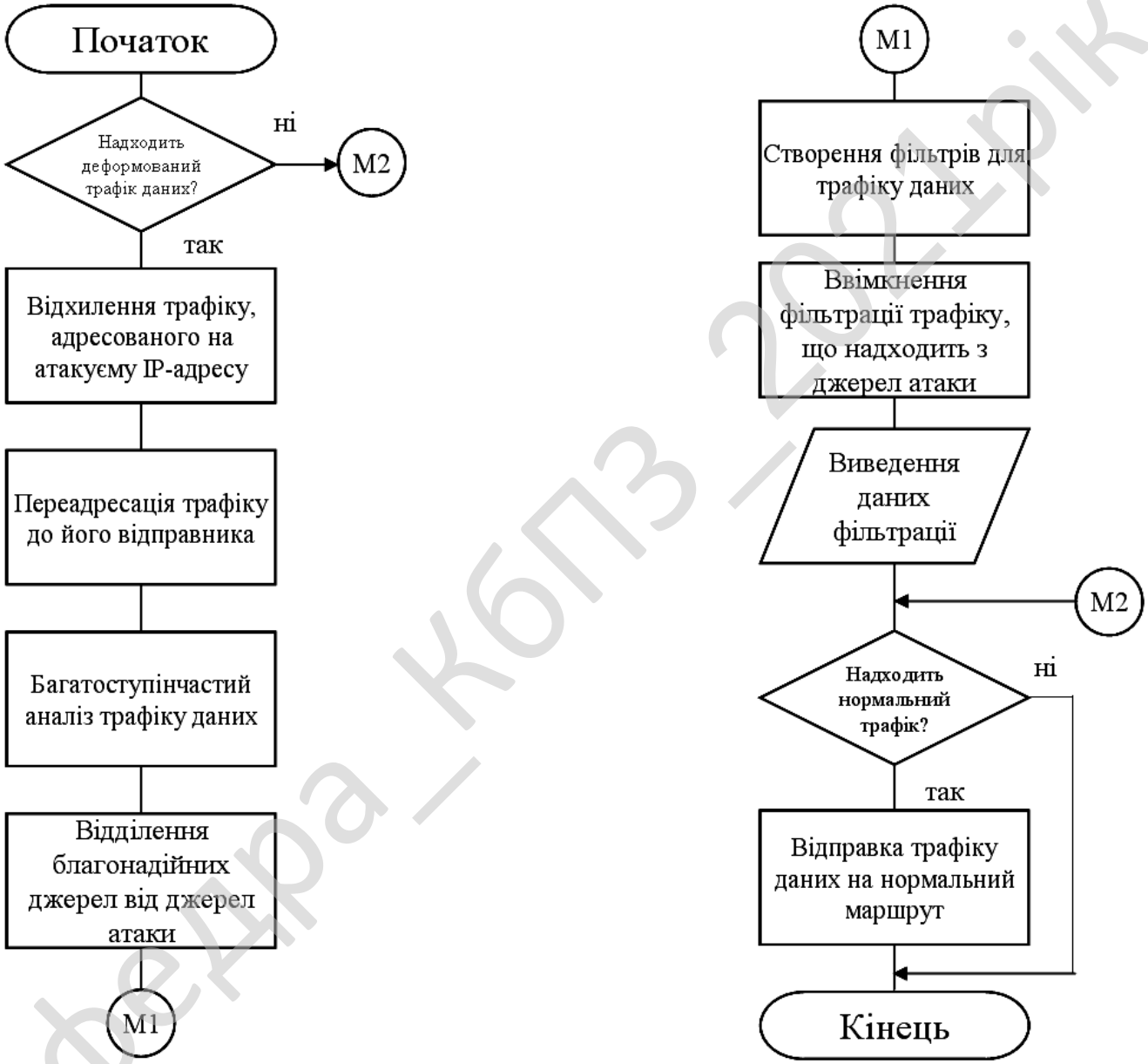

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

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

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

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

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

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

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

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

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.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для

візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

– Сповіщення про зміни за допомогою RSS-потоків та електронної пошти.

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

– Налаштування власних (custom) полів для задач, затрат часу, проектів та користувачів.

– Легка інтеграція із системами керування версіями (SVN, CVS, Git, Mercurial, Bazaar и Darcs).

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

Діаграма Ганта (*Gantt chart*, також стрічкова діаграма, графік Ганта) – це популярний тип діаграм, який використовується для ілюстрації плану, графіка робіт за будь-яким проектом. Є одним з методів планування та управління проектами.

Діаграма Ганта являє собою відрізки (графічні плашки), розміщені на горизонтальній шкалі часу. Кожен відрізок відповідає окремому завданню або підзадачі. Завдання і підзадачі, складові плану, розміщуються по вертикалі. Початок, кінець і довжина відрізка на шкалі часу відповідають початку, кінцю і тривалості завдання. На деяких діаграмах Ганта також показується залежність між завданнями.

Діаграма може використовуватися для представлення поточного стану виконання робіт: частина прямокутника, що відповідає завданню, заштриховується, відзначаючи відсоток виконання завдання; показується вертикальна лінія, що відповідає моменту «сьогодні».

Часто діаграма Ганта використовується спільно з таблицею зі списком робіт, рядки якої відповідають окремо взятій задачі, з ображеній на діаграмі, а стовпці містять додаткову інформацію про задачу.

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

Кожному, хто розробляв програмні продукти, добре знайоме співвідношення «20/80» – останні 20 % роботи тривають 80 % часу.

Як це не парадоксально, але нічого дивного в цій пропорції немає, адже саме на завершальній стадії починається тестування проекту, коли виявляються помилки, і що більший проект, то більше буде знайдено помилок.

Водночас досить часто виявляється, що більшість цих помилок були відомі та могли бути виправлені з меншими витратами на попередніх стадіях роботи, але не були вчасно описані, а потім загубилися серед інших важливих завдань.

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього

виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

Сукупність інформації про дефект. Головний компонент такої системи – база даних, що містить відомості про виявлені дефекти. Ці відомості можуть включати в себе:

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

– опис кроків для відтворення дефекту (неправильної поведінки програми);

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

Крім того, розвинені системи надають можливість прикріплювати файли, які допомагають описати проблему, наприклад, дамп пам'яті або скріншот.

Використання. Основна перевага систем відстеження помилок полягає в забезпеченні чітких централізованих оглядів, запитів на розробку (включаючи помилки і виправлення) та їх стан. У корпоративному середовищі, системи відстеження помилок можуть бути використані для генерації звітів по продуктивності програмістів виправлення помилок. Однак, це може іноді приводити до неточних результатів, тому що різні помилки можуть мати різні ступені пріоритету та серйозності, що пов'язано з складністю їх фіксації.

Життєвий цикл дефекту. Як правило, система відстеження помилок використовує той чи інший варіант «життєвого циклу» помилки, стадія якого визначається поточним станом помилки.

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

3. Дозволений – дефект переходить назад у сферу відповідальності тестувальника. Як правило, супроводжується резолюцією, наприклад:

– Виправлено (виправлення включені у версію таку-то).

– Дубль (повторює дефект, що вже знаходиться в роботі).

– Не виправлено (працює відповідно до специфікації, має занадто низький пріоритет, виправлення відкладено до наступної версії тощо).

– «В мене все працює» (запит додаткової інформації про умови, в яких дефект проявляється).

4. Далі тестувальник проводить перевірку виправлення, залежно від чого дефект або знову переходить у стан «Призначений» (якщо він описаний як виправлений, але не виправлений), або у стан «Закрито».

5. Відкрито повторно – дефект знайдено знову в іншій версії.

Система може надавати адміністраторові можливість налаштування користувачі, які можуть переглядати і редагувати помилки залежно від їх стану, переводити їх в інший стан або видаляти.

У корпоративному середовищі, система відстеження помилок може використовуватися для отримання звітів, що показують продуктивність програмістів при виправленні помилок. Однак, часто такий підхід не дає достатньо точних результатів через те, що різні помилки мають різну ступінь серйозності та складності. При цьому серйозність проблеми прямо не стосується складності її усунення.

Управління вимогами це процес запису, аналізу, трасування, пріоритезації і узгодження вимог та контролю змін і доведення до їх зацікавлених сторін. Це безперервний процес протягом всього життя проекту. Вимога – якість, якій мають відповідати результати проекту (продукту або послуги).

Мета управління вимогами полягає в тому, щоб переконатися, що організація відповідає потребам і очікуванням своїх клієнтів, внутрішніх або зовнішніх зацікавлених сторін. Управління вимогами починається з аналізу і

виявлення цілей і обмежень організації. Управління вимогами додатково включає в себе підтримку планування вимог, інтеграції вимог і організації роботи з ними (атрибути для вимог).

Управління вимогами передбачає спілкування між членами проектної групи і зацікавленими сторонами, і адаптацію до змін у вимогах протягом всього проекту. Щоб запобігти перетину поля одного класу вимог з іншим, постійні зв'язки між членами команди розробників є критичними. Наприклад, при розробці програмного забезпечення для внутрішнього використання у бізнесу можуть бути настільки сильні потреби, що він може проігнорувати вимоги користувачів, або вважати, що створені сценарії використання покриють також і користувальницькі вимоги.

Відслідковування вимоги фактично означає документування всього життєвого циклу вимоги. Часто необхідно дізнатися першоджерело кожної вимоги. Для цього всі зміни вимог повинні бути задокументовані, щоб досягти стану повного відстеження. Відстежувати треба бути навіть використання реалізованих вимог.

Вимоги мають різні джерела, такі як ділова людина, що замовляє продукт, менеджер зі збуту і фактичний користувач. У всіх цих людей є різні вимоги до продукту. Використовуючи відслідковування вимог, реалізована в системі функція може бути простежена назад до людини або групі, яка замовляла її під час збору вимог. Ця особливість може, наприклад, використовуватися в процесі розробки для пріоритезації вимог, визначаючи, наскільки цінною є дана вимога для певного користувача.

Відслідковування може також використовуватися після розгортання продукту. Наприклад, коли вивчення використання системи показує, що якась функція не використовується, можна визначити навіщо вона була потрібна спочатку.

Завдання управління вимогами

На кожному етапі процесу розробки існують ключові методи і задачі пов'язані з управлінням вимогами. Для ілюстрації, розглянемо наприклад стандартний процес розробки з п'ятьма фазами: дослідженням, аналізом здійсненності, дизайном, розробкою та тестуванням і випуском.

Дослідження. Під час фази дослідження збираються перші три класи вимог від користувачів, бізнесу і команди розробників. У кожній області задають однакові питання: які цілі, які обмеження, які використовуються процеси та інструменти і так далі. Тільки коли ці вимоги добре зрозумілі, можна приступати до розробки функціональних вимог.

Тут необхідне застереження: незалежно від того, як сильно група намагається це зробити, вимоги не можуть бути повністю визначені на початку проекту. Деякі вимоги змінюються, або тому що вони просто не були знайдені спочатку, або тому що внутрішні чи зовнішні сили торкаються проекту в середині циклу. Таким чином, учасники групи повинні спочатку погодитися, що головна умова успіху – гнучкість у мисленні та діях.

Результатом стадії дослідження є документ – специфікація вимог, схвалений усіма членами проекту. Пізніше, в процесі розробки, цей документ буде важливий для запобігання розповзання меж проекту або непотрібних змін. Оскільки система розвивається, кожна нова функція відкриває світ нових можливостей, таким чином специфікація вимог прив'язує команду до оригінального бачення системи і дозволяє контрольоване обговорення змін.

У той час як багато організацій все ще використовують звичайні документи для керування вимогами, інші управляють своїми базовими вимогами, використовуючи програмні інструменти.

Ці інструменти керують вимогами використовуючи базу даних, і зазвичай мають функції автоматизації відстеження (наприклад, дозволяючи створювати зв'язки між батьківськими і дочірніми вимогами, або між тестами і вимогами), управління версіями, і управління змінами. Зазвичай такі інструментальні засоби

містять функцію експорту, яка дозволяє створювати звичайний документ, експортуючи дані вимог.

Аналіз здійсненності

На стадії аналізу здійсненності визначається вартість вимог. Для користувальницьких вимог поточна вартість роботи порівнюється з майбутньою вартістю встановленої системи. Задаються питання такі як: «Скільки нам зараз варті помилки введення даних?» Або, «Яка вартість втрати даних через помилки оператора пов'язаної з використовуваним інтерфейсом?». Фактично, потреба в новому інструменті часто розпізнається, коли подібні питання потрапляють до уваги людей, що займаються в організації фінансами.

Ділова вартість включає відповіді на такі питання як: «У якого відділу є бюджет на це?» «Який рівень повернення коштів від нового продукту на ринку?» «Який рівень скорочення внутрішніх витрат на навчання і підтримку, якщо ми зробимо нову, більш просту в використанні систему?»

Технічна вартість пов'язана з вартістю розробки програмного забезпечення та апаратною вартістю. «Чи є у нас потрібні люди, щоб створити інструмент?» «Чи потребуємо ми нове устаткування для підтримки нової системи?»

Подібні питання дуже важливі. Група повинна з'ясувати, чи буде новий автоматизований інструмент мати достатню ефективність аби перенести частину тягаря користувачів на систему і зекономити час людей.

Ці питання також вказують на основну суть управління вимогами. Людина і інструмент формують систему, і це розуміння особливо важливе, якщо інструмент – комп'ютер або новий додаток на комп'ютері. Людський розум вкрай ефективний у паралельній обробці та інтерпретації тенденцій з недостатніми даними. Комп'ютерний процесор ефективний у послідовній обробці і точному математичному обчисленні. Основна мета управління вимогами для програмного проекту полягала б у тому, щоб гарантувати, що автоматизована робота призначена «правильному» процесору.

Наприклад, «не змушуйте людину пам'ятати, де вона знаходиться в системі. Примусьте інтерфейс завжди повідомляти про місцезнаходження людини в системі ». Або «не змушуйте людини вводити ті ж самі дані в два екрани. Примусьте систему зберігати дані і заповнювати їх де необхідно автоматично». Результатом стадії аналізу здійсненності є бюджет і графік проекту.

Дизайн. Припускаючи, що вартість точно визначена і переваги, які будуть отримані, є досить великими, проект може перейти до стадії проектування.

На стадії дизайну основна діяльність управління вимогами полягає в тому, щоб перевіряти чи відповідають результати дизайну документу вимог, щоб упевнитися, що робота залишається в межах проекту.

І знову, гнучкість є ключем до успіху. Ось класичний приклад змін проекту, які відмінно працювали. Проектувальники Форда на початку 1980-х очікували, що ціни на бензин піднімуться до 3,18 дол за галон до кінця десятиліття. На середині процесу дизайну автомобіля Ford Taurus, ціни встановилися приблизно на рівні 1,50 дол за галон. Колектив дизайнерів вирішив, що вони могли б створити більший, більш зручний, і більш потужний автомобіль, якщо б ціни на бензин залишилися низькими. Таким чином, вони перепроектувати автомобіль. Коли новий автомобіль вийшов, він встановив загальнонаціональні рекорди продажів.

У більшості випадків, однак, відступ від оригінальних вимог до такої міри не працює. Таким чином документ вимог стає ключовим інструментом, який допомагає команді приймати рішення про зміни дизайну.

Розробка та тестування. На стадії розробки і тестування, основна діяльність управління вимогами – це гарантувати, що робота і ціна залишаються в межах графіка і бюджету, і що створюваний інструмент дійсно відповідає вимогам. Основним інструментом, використовуваним на цій стадії, є створення прототипу і ітераційне тестування.

Для програмного додатка користувацький інтерфейс може бути створений на папері і перевірений з потенційними користувачами, в той час як створюється

основа програми. Результати цих тестів записуються в керівництві по дизайну користувацького інтерфейсу і передаються колективу дизайнерів. Це економить їх час і робить їх завдання набагато простіше.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм Khufu. Khufu – це 64-бітовий блоковий шифр. 64-бітовий відкритий тест спочатку розщеплюється на дві 32-бітові половини, *L* і *R*. Над обома половинами й певними частинами ключа виконується операція XOR. Потім, аналогічно DES, результати проходять деяку послідовність раундів. У кожному раунді молодший значущий байт *L* використовується як вхід S-блоку. У кожного S-блоку 8 вхідних біт і 32 вих ідних біта. Далі обраний в S -блоці 32-бітовий елемент піддається операції XOR з *R*. Потім *L* циклічно зрушується на число, кратним восьми биткам, *L* і *R* міняються місцями, і раунд завершується. Сам S-блок не статичний, він міняється кожні вісім раундів. Нарешті, по закінченні останнього раунду, над *L* і *R* виконується операція XOR з іншими частинами ключа, і половини поєднуються, утворюючи блок шифртексту.

Хоча частини ключа використовуються для операції XOR із блоком шифрування на початку й кінці виконання алгоритму, головне призначення ключа – генерація S-блоків. Ці S-блоки секретні, по суті, це частина ключа. Повний розмір ключа алгоритму Khufu дорівнює 512 біт (64 байт), алгоритм надає спосіб генерації S-блоків по ключу.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у магістерській дипломній роботі системи зберігання даних з використанням технології Fibre Channel 6. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи: Функції представлені у графічному вигляді (іконки); Функціональних кнопок ПЗ; Навігаційного меню; Верхнього меню.

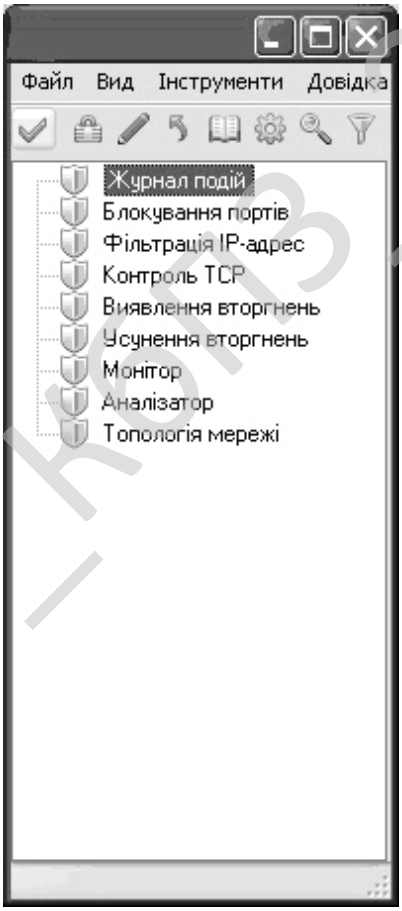

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем.

Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

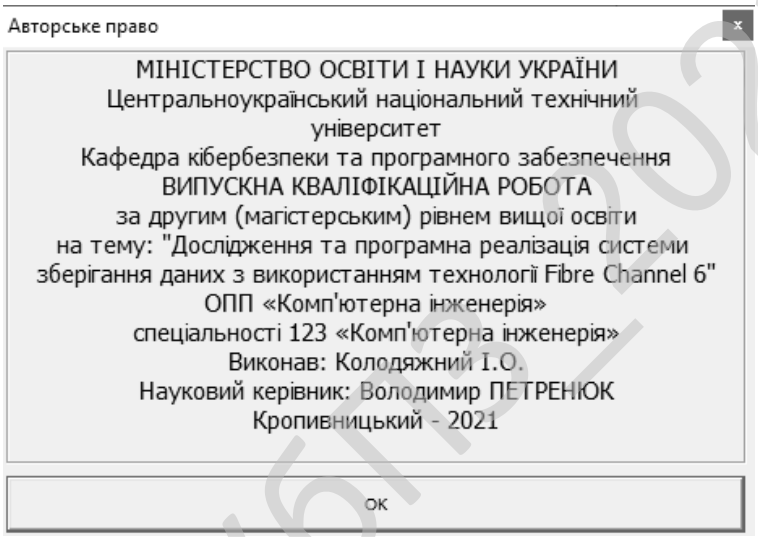

Рисунок 5.2 – Авторське право

Розглянемо процес впровадження програмного забезпечення, це процес налаштування програмного забезпечення під певні умови використання, а також навчання користувачів роботі з програмним продуктом. Впровадження програмного забезпечення це усі дії, що роблять розроблену програмну систему готовою до використання. Даний процес є частинною життєвого циклу програмного забезпечення.

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити.

Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

– Виділення критичних, з точки зору загального результату, процедур в діяльності організації. Коли набір таких процедур визначений, необхідно в першу чергу використовувати IT рішення для автоматизації операцій усередині саме цих процедур. Таким чином, розроблене IT рішення автоматично стає життєво важливим і затребуваним для організації, а також буде забезпечена публічність процесу впровадження;

– Розширення нормативної бази організації шляхом включення до неї регламентів, що описують порядок виконання процедур автоматизованих процесів. В іншому випадку є небезпека виникнення неузгодженості між автоматизованими процедурами та іншими процесами організації.

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в IT рішення за принципом найбільшої корисності для більшості учасників.

Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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 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 OT.

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 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи зберігання даних з використанням технології Fibre Channel 6.

Метою розробки є дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

Об'єктом дослідження є процес зберігання даних з використанням технології Fibre Channel 6.

Предметом дослідження є методи зберігання даних з використанням технології Fibre Channel 6.

Методи дослідження базуються на методах теорії телекомунікації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод зберігання даних з використанням технології Fibre Channel 6.

– Розроблено вітчизняний продукт зберігання даних з використанням технології Fibre Channel 6, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці). В магістерській роботі було проведено дослідження та виконана програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликі системні потреби;

б) незалежність від встановлених на комп'ютері баз даних;

в) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	19 (2 ост. цифри № за
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	2
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	19000 (2 ост. цифр № зал*10^3)
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A \, \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, $A = 2{,}45$; Size – загальний об'єм відлагодженого програмного коду, тис. рядків; B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПV_j, \qquad (7.3)$$

де: $ПV_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$

$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4). Для мови програмування Delphi він складає 2,66; S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо S =100 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \tag{7.5}$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор-маршрутизатор	155	2	310	5
Кабельні господарства ЛОМ на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 3}{1,2} = 567,5 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 567,5/(60·8) = 1,2 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2016, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	13000	39000
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	1,2	11500	41400
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 8,25	-	$Ф_{роб}$ = 297225

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{297225}{8,25 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград, вул. Глинки 16) ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 25 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Комп'ютерторг за 15.10.21 – джерело http://computorg.ua/ru/price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	INTEL CELERON G5905 (BX80701G5905 box	1750
Системна плата	ASROCK H470M-HDV, підтримка 10-г покоління Intel Core i9/Core i7/Core i5/Cor i3/Pentium/Celeron, 1 x PCI-E 3.0 x1, 1 x PCI E 3.0 x16, 1 x VGA, 2 x USB 3.2 Gen 1, 1 RJ45, 4 x USB 2.0, 1 x PS/2, 1 x DVI, 1 HDMI, 3 x Audio, Ami, UEFI	1200
Відеокарта	VC VTX Radeon HD6570 1GB GDDR3 128bit, 650 MHz/1334 MHz, PCI-E 2.1, DVI HDMI, VGA (VX6570 1GBK3-H)	750
Жорсткий диск	HDD: 320 Gb 7200 Serial ATA WD 16MB	1200
Оперативна пам’ять	DDR4 8GB 2666 MHZ GOODRAM (GR2666D464L19S/8G)	900
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slim Black Bulk (DVR-TD10RS)	416
Корпус	ATX Middle Tower FOXCONN Pro, 3GTLA 489, PSU 350W(FSP Brand: ATX-350PNR 12cm), black, (front bezel – black+light silver body material – 0.6mm), 80mm fan (rear 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	911

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: Al Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відраху-вання, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	28000	25	7000
Всього по групі	130690	-	27797,5
7. Нематеріальні активи	19000	10	1900
Разом	$К_р$ = 1672575		$А_р$ = 157540

Примітка: вартість автомобіля взята по даним з автосалону автотрейдінг, вкладки автобазар, джерело http: //www.auto-trading.com.ua/sale/lot20772.html, складає 3900 USD, що враховуючи курс 25 складає 97500 грн.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 600 \cdot 209/19 = 6600 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 6600 \cdot 10 \cdot 0{,}01 = 660 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 37\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_о), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 37(6600+660) = 2686 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.;

$З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.;

$З_{М3}$ – вартість фарби, картріджей, тонеру, грн.;

$N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на три місяці розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 105 грн., визначаємо вартість паперу за період розробки $N_м$ = 3 міс:

$$З_{М1} = Ц_п \cdot N_м. \quad (7.16)$$

$$З_{М1} = 105 \cdot 1 = 105 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості екземплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 5 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 12 грн./шт.

$$З_{М2} = 19 \cdot 5 + 12 = 107 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574 + 558 + 570 = 1702 \text{ грн.}$$

$$З_М = (105 + 107 + 1702)/19 = 101 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$O_n = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	6600
2. Додаткова зарплата виконавців	$З_д$	660
3. Відрахування на соціальні потреби	$С_{оц}$	2686
4. Загальногосподарські витрати	$Г_{осп}$	990
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	990
7. Амортизація основних фондів	$А_м$	2073
8. Повна собівартість програмного забезпечення	$С_п$	14100
9. Плановий прибуток	$П_р$	7050
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	21150
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	4230
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	25380

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 19 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \qquad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 157540 \cdot 3/(19 \cdot 12) = 2073 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 6600+660+2686+990+101+990+2073 = 14100 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 14100 = 7050 \text{ грн.}$$

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	25380
Всього капітальних витрат	–	25380

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	135274	56364
2. Витрати на електроенергію	$З_{ел}$	20580	17640
3. Витрати на амортизацію	$З_{ам}$	0	6345
Всього витрат за рік	$І$	155854	80349

Витрати на профілактичні роботи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 240 годин на рік до 100 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 240 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 7 = 135274 \text{ грн},$$

до:

$$З_{р\,нов} = 100 \cdot 60 \cdot 1{,}1 \cdot 1{,}22 \cdot 7 = 56364 \text{ грн}.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

$$З_{ел\ баз} = 7 \cdot 0{,}35 \cdot 4000 \cdot 2{,}1 = 20580 \text{ грн.}$$

$$З_{ел\ нов} = 7 \cdot 0{,}3 \cdot 4000 \cdot 2{,}1 = 17640 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукці	25	–	25380	–	6345
Всього відрахувань	-	–	25380	–	6345

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (21150 - 14100) \cdot 19 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}25 \cdot 33190 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 19000) \cdot 3/12 = 94565 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_p^*}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{264575}{(21150 - 14100) \cdot 19 \cdot 12 / 3} = 0{,}5 \text{ років}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	19
2. Повна собівартість розробленої програми	Грн.	14100
3. Ціна розробленої програми	Грн.	21150
4. Плановий прибуток від реалізації розробленої програми	Грн.	7050
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1672575
7. Загальний прибуток від реалізації програмної продукції	Грн.	133950
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	94565
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,5
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	25380
11.Величина економічного ефекту у користувача програмної продукції	Грн.	69160
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,3

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \quad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно;

$K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (155854\text{-}80349)\text{-}0{,}25 \cdot 25380 = 69160 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \quad (7.28)$$

$$T_{сп} = \frac{25380}{155854 - 80349} = 0{,}3 \ року.$$

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Протягом усієї історії людство приділяє прискіпливу увагу безпеці життя [1]. Охорона праці є складовою частиною безпеки життя.

Законом України “Про охорону праці” [2] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж ic тотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98, та

«Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста вулючають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Програміст працює з електронно-обчислювальною машиною (ЕОМ) та іншим обладнанням, яке є джерелом небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. Так як програміст постійно перебуває в приміщенні, тому для комфортних умов праці в цьому приміщенні необхідно створити належний мікроклімат.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– електромагнітні (у т.ч. високочастотні) випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– недостатня, або надмірна освітленість робочого місця;

– монотонність праці;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шуми;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	9
Довжина	9
Висота	3

У зазначеному приміщенні працє 12 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,75
Обсяг, V	$м^3$	не менше 20.0	20,25

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,1
Тепла	23-25	50-70	0,1	23-24	50-68	0,1

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер *HP Laser 107a*, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

8.5 Розрахункова частина

Проводемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 9 м, довжина 9 м, висота 3 м.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; E = 300 Лк;

S – площа освітлюваного приміщення (у нашому випадку S= 9 х 9 = 81 м^2);

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку K = 1,5);

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку Z = 1,1);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін ($\rho_{стін.}$) і стелі ($\rho_{стелі}$), значення коефіцієнтів дорівнюють $\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, S = 81 м^2;

h – розрахункова висота підвісу, h = 3 м (співпадає з висотою стелі, т.я. лампи освітлення закріплюються на стелі);

A – ширина приміщення, A = 9 м;

B – довжина приміщення, B = 9 м.

Підставимо всі значення у формулу та визначемо індекса приміщення:

$$i=1,5.$$

Знаючи індекс приміщення (i), за знаходимо n = 0,5 (з табличних даних коефіцієнтів використання світлового потоку світильників з відповідним типом ламп [3]). Підставимо всі значення у формулу, визначемо світловий потік: F=75651 Лм.

Для штучного освітлення приміщення використовуються *LED* світильники *PANEL-B2B-595*, світловий потік яких $F_л$ = 3500 Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо всі значення у формулу та визначемо індекса приміщення: N= 75651/ 3500=21,6 шт.

Для забезпечення нормованої мінімальної освітленості приймаємо необхідну кількість світильників 22 шт.

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров’я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи зберігання даних з використанням технології Fibre Channel 6.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів зберігання даних з використанням технології Fibre Channel 6.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем зберігання даних з використанням технології Fibre Channel 6.

– Досліджена система зберігання даних з використанням технології Fibre Channel 6.

– На основі отриманих результатів досліджень створена програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання зберігання даних з використанням технології Fibre Channel 6.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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 10.3.2 Rio Architect. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 XP/Vista/7/8/10.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм Khufu.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 69160 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,3 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Тарасенко Б.О. Дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6 // Збірник праць молодих науковців ЦНТУ. – Вип. 12. – Кропивницький: ЦНТУ, 2022.
2. Астахова, И.Ф. Компьютерные науки. Деревья, операционные системы, сети / И.Ф. Астахова, И.К. Астанин и др. – М.: Физматлит, 2013. – 88 с.
3. Астахова, И.Ф. Компьютерные науки. Деревья, операционные системы, сети / И.Ф. Астахова и др. – М.: Физматлит, 2013. – 88 с.
4. Гудыно, Л.П. Вычислительные системы, сети и телекоммуникации: Учебное пособие / А.П. Пятибратов, Л.П. Гудыно, А.А. Кириченко; Под ред. А.П. Пятибратов. – М.: КноРус, 2013. – 376 с.
5. Дейтел, Х.М. Операционные системы. Т. 2. Распределенные системы, сети, безопасность / Х.М. Дейтел, П.Д. Дейтел, Д.Р. Чофнес; Пер. с англ. С.М. Молявко.. – М.: БИНОМ, 2013. – 704 с.
6. Дейтел, Х.М. Операционные системы. Распределенные системы, сети, безопасность / Х.М. Дейтел, Д.Р. Чофнес. – М.: Бином, 2013. – 704 с.
7. Замятина, О.М. Вычислительные системы, сети и телекоммуникации. моделирование сетей.: Учебное пособие для магистратуры / О.М. Замятина. – Люберцы: Юрайт, 2016. – 159 с.
8. Зотов, А.Ф. Вычислительные системы, сети и телекоммуникации / А.Ф. Зотов. – М.: КноРус, 2012. – 288 с.
9. Идельчик, В.И. Электрические системы и сети: Учебник для вузов. / В.И. Идельчик. – М.: Альянс, 2016. – 592 с.
10. Кин, Э. Ничего личного: Как социальные сети, поисковые системы и спецслужбы используют наши персональные данные / Э. Кин. – М.: Альпина Паблишер, 2016. – 224 с.

11. Крук, Б.И. Телекоммуникальные системы и сети: Учебное пособиеВ 3 т. Т. 1. – Современные технологии / Б.И. Крук, В.Н. Попантонопуло. – М.: ГЛТ, 2013. – 620 с.

12. Крук, Б.И. Телекоммуникационные системы и сети.Уч пособие в 3-х томах.Том 1 / Б.И. Крук. – М.: ГЛТ, 2012. – 620 с.

13. Мелехин, В.Ф. Вычислительные системы и сети: Учебник для студентов учреждений высш. проф. образования / В.Ф. Мелехин, Е.Г. Павловский. – М.: ИЦ Академия, 2013. – 208 с.

14. Оссовская, М.П. Волоконно-оптические сети и системы связи: Учебное пособие КПТ / М.П. Оссовская. – СПб.: Лань КПТ, 2016. – 272 с.

15. Рождествина, А.А. Вычислительные системы, сети и телекоммуникации (для магістеров) / А.А. Рождествина. – М.: КноРус, 2013. – 376 с.

16. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: РиС, 2013. – 384 с.

17. Самуйлов, К.Е. Сети и системы передачи информации: телекоммуникационные сети: Учебник и практикум для академического магістериата / К.Е. Самуйлов, И.А. Шалимов, Д.С. Кулябов. – Люберцы: Юрайт, 2016. – 363 с.

18. Хрущев, Ю.В. Электроэнергетические системы и сети. электромеханические переходные процессы: Учебное пособие для прикладного магістериата / Ю.В. Хрущев, К.И. Заподовников, А.Ю. Юшков. – Люберцы: Юрайт, 2016. – 153 с.

19. Шевченко, В.П. Вычислительные системы, сети и телекоммуникации: Учебник / В.П. Шевченко. – М.: КноРус, 2012. – 288 с.

20. Будько М.Ю. Повышение безопасности работы компьютерних сетей на основе анализа потоков данних // Известия высших учебних заведений. Приборостроение. Том 52. Номер выпуска 5. – СПб.: СПбГУ ИТМО, 2009. – С. 31-34.

21. Будько М.Ю., Будько М.Б. Отслеживание изменений в структуре сети и решение задач повышения безопасности на основе анализа потоков данних // Научно-технический вестник Санкт-петербургского государственного университета информационних технологий, механики и оптики. Номер выпуска 59. – СПб.: СПбГУ ИТМО, 2009. – С. 78-82.

22. Будько М.Ю., Будько М.Б. Определение источника широковещательного шторма на основе данних протокола SNMP // Сборник трудов конференции молодих ученых. Выпуск 6. Информационні технологии. – СПб.: СПбГУ ИТМО, 2009. – С. 153-157.

23. Будько М.Ю. Сравнение эффективности критериев для определения связей между устройствами в сети // Труды XV Всероссийской научно-методической конференции "Телематика'2008". – 2008. – С. 179-181.

24. Будько М. Ю. Программный комплекс для анализа сетевой статистики // Труды XII-й международной научно-практической конференции «Теория и технология программирования и защиты информации». – СПб.: СПбГУ ИТМО, 2008. – С. 71-75.

25. Будько М.Ю. Метод динамического построения топологии сети для ре- шения задач обнаружения угроз безопасности // Сборник тезисов V Всероссийской межвузовской конференции молодих ученых. – СПб: СПбГУ ИТМО, 2008. – С. 98-99.

26. Будько М.Ю., Будько М.Б., Гирик А.В. Заявка на патент на изобретение № 2007111888/09 «Способ адаптивного управления передачей потокових медиаданных» // Федеральная служба по интеллектуальной собственности, патентам и товарным знакам; Бюллетень №28 – Москва, 2008.

27. Будько М. Ю., Будько М. Б., Гирик А. В. Применение авторегрессионного интегрированного скользящего среднего в алгоритмах управления перегрузками протоколов передачи потокових данних // Научно-технический вестник СПбГУ ИТМО. – 2007. – № 39. – С. 319-323.

28. Босько В.В. Анализ и сравнительное исследование перспективных

направлений развития цифровых телекоммуникационных систем и сетей / Смирнов А.А., Босько В.В., Мелешко Е.В. // Системи обробки інформації.– Х.: ХУ ПС, 2008. – Вип.7(74). – С.120-123.

29. Босько В.В. Разработка методики оценки середнего времени обслуживания информационных пакетов в телекоммуникационной сети / Смирнов А.А., Босько В.В., Мелешко Е.В. // Системи управління, навігації та зв'язку.– Київ: ДП «Центральний науково-дослідний інститут навігації і управління», 2009. – Вип. 2(10). – С.162-165.

30. Босько В.В. Разработка математической модели процесса динамического управления маршрутизацией данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Системи управління, навігації та зв 'язку. – К.: ДП «ЦНДІ навігації і управління», 2009. – Вип. 3(11). – С.208-210.

31. Босько В.В. Разработка метода определения оптимального множества путей передачи информации в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Збірник наукових праць. – Х.: ХУПС, 2009. – Вип. 3(21). – С.102-108.

32. В.В. Босько. Разработка метода прогнозирования поведения информационного потока в телекоммуникационной сети // Збірник наукових праць. Харківського університету Повітряних Сил: – Х.:ХУ ПС, – 2010.-Вип. 3 (25) .- С.126-130.

33. Босько В.В. Исследование особенностей построения современных телекоммуникационных систем и сетей / Смирнов А.А., Босько В.В. // Проблемы информатики и моделирования. Материалы восьмой международной научно-технической конференции 26-28 ноября. – Х.: НТУ "ХПИ", 2008. – С.54.

34. Босько В.В. Исследование влияния времени мониторинга телекоммуникационной сети на точность прогнозирования поведения трафика / Смирнов А.А., Босько В.В. // Перша міжнародна науково-практична конференція "Проблеми й перспективи розвитку IT-індустрії" м. Харків , 18 -19 листопада 2009р. – С.198-200.

35. Босько В.В. Анализ математических моделей телекоммуникационной

сети / Смирнов А.А., Босько В.В. // Проблемы информатики и моделирования. Материалы девятой международной научно-технической конференции 26-28 ноября. – Х.: НТУ “ХПИ”, 2009. – С.52-53.

36. Босько В.В. Метод повышения оперативности передачи данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Збірник тез доповідей науково-практичної конференції “Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку” 24-25 березня. – Х.: АВВ МВС України, 2010. – С.54.

37. Босько В.В. Динамическое управление процессом маршрутизации данных в телекоммуникационной сети / Смирнов А.А., Босько В.В. // Тези доповідей Міжнародної науково-практичної конференції м. Вінниця, Україна 19-21 травня 2010 року. – Вінниця: ВНТУ, 2010. – С.231-232.

38. Можаєв О.О. Часова прозорість мережі, як характеристика, що визначає виконання необхідної якості обслуговування / О.О. Можаєв, О.Д. Анохіна, С.Ю. Гайдаров, С.Г. Семенов // Системи обробки інформації. – Х.: ХВУ, 2004. – Вип. 11(39). – С.133-139.

39. Науменко М.І. Технології комп’ютерних мереж АСУ військами: підручник. / М.І. Науменко, Ю.В. Стасєв, І.В. Рубан – Х.: ХУ ПС, 2004. – 458 с.

40. Олифер В.Г. Компьютерные сети. Принципы, технологии, протоколы: учебник для вузов / В.Г. Олифер, Н.А. Олифер. –2-е изд. – СПб.: Питер, 2007. – 958 с.

41. Семенов Ю.А. Сети Інтернет. Архитектура и протоколы / Ю.А. Семенов. – М.: Блик плюс, 1998. – 424 с.

42. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум; пер. с англ. А. Леонтьев. – СПб.: Питер, 2002. – 848 с.

43. Чернявский Г.М. Новые технологии в спутниковых системах / Г.М. Чернявский // Информационные технологии и вычислительные системы. – М.: Институт микропроцессорных вычислительных систем РАН, 2005. – Вып.1 – С. 3 – 11.

44. Галкин В.А. Телекоммуникации и сети / В.А. Галкин, Ю.А.Григорьев.– М.: МГТУ имени Н.Э. Баумана, 2003. – 608 с.

45. ДСТУ 2481 – 94 Системи оброблення інформації інтелектуальні інформаційні технології. Терміни та визначення. – Х.: ДЕРЖСТАНДАРТ УКРАЇНИ, 1994. – 33 с.

46. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення.– К.:УкрНДІССІ, 1995. – 23 с.

47. ITU-T Y.1540. Internet protocol data communication service – IP packet transfer and availability performance parameters, 2007.

48. Пантелеев А.В. Т.А. Методы оптимизации в примерах и задачах / А.В. Пантелеев, Т.А. Летова. – М.: Высшая школа, 2002. – 544 с.

49. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

50. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12#Text

51. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

52. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0019.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Тарасенко Б.О.			Дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Петренюк В.І.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи зберігання даних з використанням технології Fibre Channel 6.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи зберігання даних з використанням технології Fibre Channel 6.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи зберігання даних з використанням технології Fibre Channel 6;

– 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 архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

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 XP/Vista/7/8/10.

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 10.3.2 Rio Architect.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинна бути розглянута розробка заходів з умов поліпшення охорони праці.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 140 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2021 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Петренюк В.І.

Дослідження та програмна реалізація

системи зберігання даних з використанням технології Fibre Channel 6

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 49

Літера: РП

Кропивницький – 2021 року

Основна програма

Файл Data_Center_Fibre_Channel_6.dpr основної програми

```
program Data_Center_Fibre_Channel_6;

uses
  Forms,
  Main in ' Main.pas'  {MainForm},
  About in ' About.pas'  {Form1},
  TCP_IP in ' TCP_IP.pas'  {Form2},
  Stat in ' Stat.pas'  {Form3};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TMainForm, MainForm);
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.CreateForm(TForm3, Form3);
  Application.Run;
end.
```

Файл Main.pas основної програми

```
unit Main;

interface

// опис бібліотек

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, Stat,
  ShellAPI, ShlObj, ImgList, TCP_IP, About;

//опис типів

type
  TMainForm = class(TForm)
    gbxShares: TGroupBox;
    lbxShares: TListBox;
    gbxSessions: TGroupBox;
    lvSessions: TListView;
    bvlSessions: TBevel;
    gbxFiles: TGroupBox;
    btnGetShares: TButton;
    btnCloseShares: TButton;
    btnAddShares: TButton;
    btnCloseSession: TButton;
    btnGetSessions: TButton;
    bvlTopSessions: TBevel;
    plButtonFiles: TPanel;
    btnGetFiles: TButton;
    btnCloseFile: TButton;
    bvlLeftFiles: TBevel;
    plFiles: TPanel;
    lvFiles: TListView;
    bvlTopFiles: TBevel;
    gbxTraffic: TGroupBox;
    lvTraffic: TListView;
    bvlTraffic: TBevel;
    tmrTraffic: TTimer;
    Button1: TButton;
    rgScope: TRadioGroup;
    GroupBox1: TGroupBox;
    cbUsageAll: TCheckBox;
    cbUsageConnectable: TCheckBox;
    cbUsageContainer: TCheckBox;
    GroupBox2: TGroupBox;
    cbTypeAny: TCheckBox;
    cbTypeDisk: TCheckBox;
    cbTypePrint: TCheckBox;
    NetTree: TTreeView;
    ImageList1: TImageList;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    function  IsNT(var Value: Boolean): Boolean;
    procedure btnGetSharesClick(Sender: TObject);
    procedure btnCloseSharesClick(Sender: TObject);
    function  SelectDirectory: String;
    procedure btnAddSharesClick(Sender: TObject);
    function  CardinalToTimeStr(Value:Cardinal):String;
    procedure btnGetSessionsClick(Sender: TObject);
    procedure btnCloseSessionClick(Sender: TObject);
    procedure btnGetFilesClick(Sender: TObject);
    procedure btnCloseFileClick(Sender: TObject);
    procedure tmrTrafficTimer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
```

```
    procedure NetTreeCustomDrawItem(Sender: TCustomTreeView;
      Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
    procedure NetTreeDblClick(Sender: TObject);
    procedure NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
    procedure Button4Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);

//опис типів та записів


  private
    { Private declarations }
  public
    { Public declarations }
    SessionCloseKey: array [0..512] of SmallInt;
    procedure Open_Do_Close_Enum(const ParentNode: TTreeNode;
        ResScope, ResType, ResUsage: DWORD; const NetContainerToOpen:
PNetResource);
  //  function OpenEnum(const NetContainerToOpen: PNetResource;
 //        ResScope, ResType, ResUsage: DWORD): THandle;
 //   function EnumResources(const ParentNode: TTreeNode;
  //       ResScope, ResType, ResUsage: DWORD; hNetEnum: THandle): UINT;
  end;

type
 TShareInfo2 = packed record
  shi2_netname : PWChar;
  shi2_type: DWORD;
  shi2_remark :PWChar;
  shi2_permissions: DWORD;
  shi2_max_uses : DWORD;
  shi2_current_uses : DWORD;
  shi2_path : PWChar;
  shi2_passwd : PWChar;
 end;
 PShareInfo2 = ^ TShareInfo2;
 TShareInfo2Array = array [0..512] of TShareInfo2;
 PShareInfo2Array = ^ TShareInfo2Array;

type
 TShareInfo50 = packed record
  shi50_netname : array [0..12] of Char;
  shi50_type : Byte;
  shi50_flags : Word;
  shi50_remark : PChar;
  shi50_path : PChar;
  shi50_rw_password : array [0..8] of Char;
  shi50_ro_password : array [0..8] of Char;
 end;

type
 TSessionInfo502 = packed record
   Sesi502_cname: PWideChar;
   Sesi502_username: PWideChar;
   Sesi502_num_opens: DWORD;
   Sesi502_time: DWORD;
   Sesi502_idle_time: DWORD;
   Sesi502_user_flags: DWORD;
   Sesi502_cltype_name: PWideChar;
   Sesi502_transport: PWideChar;
 End;
 PSessionInfo502 = ^TSessionInfo502;
 TSessionInfo502Array = array[0..512] of TSessionInfo502;
 PSessionInfo502Array = ^TSessionInfo502Array;

type
 TSessionInfo50 = packed record
   Sesi50_cname       : PChar;
```

```
    Sesi50_username    : PChar;
    sesi50_key         : Cardinal;
    sesi50_num_conns   : Word;
    sesi50_num_opens   : Word;
    sesi50_time        : Cardinal;
    sesi50_idle_time   : Cardinal;
    sesi50_protocol    : Byte;
    pad1               : Byte;
  end;

type
  TFileInfo3 = packed record
    fi3_id          : DWORD;
    fi3_permissions : DWORD;
    fi3_num_locks   : DWORD;
    fi3_pathname    : PWChar;
    fi3_username    : PWChar;
  end;
  PFileInfo3 = ^TFileInfo3;
  TFileInfo3Array = array[0..512] of TFileInfo3;
  PFileInfo3Array = ^TFileInfo3Array;

type
  TFileInfo50 = packed record
    fi50_id          : Cardinal;
    fi50_permissions : WORD;
    fi50_num_locks   : WORD;
    fi50_pathname    : PChar;
    fi50_username    : PChar;
    fi50_sharename   : PChar;
  end;

type
   TMibIfRow = packed record
     wszName            : array[0..255] of WideChar;
     dwIndex            : DWORD;
     dwType             : DWORD;
     dwMtu              : DWORD;
     dwSpeed            : DWORD;
     dwPhysAddrLen      : DWORD;
     bPhysAddr          : array[0..7] of Byte;
     dwAdminStatus      : DWORD;
     dwOperStatus       : DWORD;
     dwLastChange       : DWORD;
     dwInOctets         : DWORD;
     dwInUcastPkts      : DWORD;
     dwInNUCastPkts     : DWORD;
     dwInDiscards       : DWORD;
     dwInErrors         : DWORD;
     dwInUnknownProtos  : DWORD;
     dwOutOctets        : DWORD;
     dwOutUCastPkts     : DWORD;
     dwOutNUCastPkts    : DWORD;
     dwOutDiscards      : DWORD;
     dwOutErrors        : DWORD;
     dwOutQLen          : DWORD;
     dwDescrLen         : DWORD;
     bDescr             : array[0..255] of Char;
   end;
   TMibIfArray = array [0..512] of TMibIfRow;
   PMibIfRow = ^TMibIfRow;
   PMibIfArray = ^TMibIfArray;

type
  TMibIfTable = packed record
    dwNumEntries     : DWORD;
    Table            : TMibIfArray;
  end;
  PMibIfTable = ^TMibIfTable;
```

```
var
NetShareEnumNT:function (     servername:PWChar;
                          level:DWORD;
                          bufptr:Pointer;
                          prefmaxlen:DWORD;
                          entriesread,
                          totalentries,
                          resume_handle:LPDWORD): DWORD; stdcall;

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

var
NetShareDelNT:function (servername: PWideChar;
                        netname: PWideChar;
                        reserved: DWORD): LongInt; stdcall;

var
NetShareDel:function (  pszServer,
                        pszNetName:PChar;
                        usReserved:Word): DWORD; stdcall;

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

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

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

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

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

var
NetSessionDel:function( pszServer:PChar;
                        pszClientName: PChar;
                        sReserved: SmallInt):DWORD; stdcall;

var
```

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

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

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

var
NetFileClose2:function( pszServer:PChar;
                        ulFileId:LongWord):DWORD; stdcall;

var
GetIfTable:function(    pIfTable    : PMibIfTable;
                        pdwSize     : PULONG;
                        bOrder      : Boolean ): DWORD; stdcall;

var
  MainForm: TMainForm;

implementation

{$R *.dfm}

{ TMainForm }

//////////////////////////
//
//  Спочатку нам потрібно визначиться, під якою системою ми працюємо,
// щоб довідатися яку частину коду (для NT чи ні) використовувати в цей момент.
// Для цього напишемо невелику функцію, що і буде визначати тип системи.
//

function TMainForm.IsNT(var Value: Boolean): Boolean;
var Ver: TOSVersionInfo;
    BRes: Boolean;
begin
  Ver.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  BRes := GetVersionEx(Ver);
  if not BRes then   //Перевірка
  begin
    Result := False; //Інформація не отримана
    Exit;            //ідемо
  end else
    Result := True;  //Інформація отримана

  case Ver.dwPlatformId of //визначаємося
    VER_PLATFORM_WIN32_NT       : Value := True;  //Windows NT тьа вище -
підходить
    VER_PLATFORM_WIN32_WINDOWS  : Value := False; //Windows 9 x-Me- підходить
    VER_PLATFORM_WIN32s         : Result := False //Windows 3.x- не підходить
  end;
end;
```

```
//////////////////////////
//
//  Одержання всіх відкритих загальних ресурсів досліджуємої мережі
//

procedure TMainForm.btnGetSharesClick(Sender: TObject);
var
  i:Integer;
  FLibHandle : THandle;
  ShareNT : PShareInfo2Array;     //<= Змінні
  entriesread,totalentries:DWORD; //<= для Windows NT
  Share : array [0..512] of TShareInfo50; //<= Змінні
  pcEntriesRead,pcTotalAvail:Word;        //<= для Windows 9 x-Me
  OS: Boolean;
begin
  lbxShares.Items.Clear;
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnumNT := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnumNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    ShareNT := nil; //Очищаємо покажчик на масив структур
    //Виклик функції
    if NetShareEnumNT(nil,2,@ShareNT,DWORD(-1),
                      @entriesread,@totalentries,nil) <> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if entriesread > 0 then //Обробка результатів
    for i:= 0 to entriesread- 1 do
      lbxShares.Items.Add(String(ShareNT^[i].shi2_netname));
  end else begin //Код для 9 x-me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' ); //Завантажуємо бібліотеку
    if FLibHandle = 0 then Exit;
    //Зв' язуємо функцію
    @NetShareEnum := GetProcAddress(FLibHandle,' NetShareEnum' );
    if not Assigned(NetShareEnum) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Виклик функції
    if NetShareEnum(nil,50,@Share,SizeOf(Share),
                  @pcEntriesRead,@pcTotalAvail)<> 0 then
    begin  //Якщо виклик невдалий вивантажуємо бібліотеку
      FreeLibrary(FLibHandle);
      Exit;
    end;
    if pcEntriesRead > 0 then //Обробка результатів
    for i:= 0 to pcEntriesRead- 1 do
      lbxShares.Items.Add(String(Share[i].shi50_netname));
  end;
  FreeLibrary(FLibHandle); //Не забуваємо вивантажити бібліотеку
end;

//////////////////////////
//
//  Закриття загального ресурсу
//

procedure TMainForm.btnCloseSharesClick(Sender: TObject);
```

```
var
  OS:Boolean;
  FLibHandle : THandle;
  Name9x:array [0..12] of Char;
  NameNT:PWChar;
  i:Integer;
  ShareName: String;
begin
  if not IsNT(OS) then Close; //Визначаємо тип системи

  if lbxShares.Items.Count = 0 then Exit;
  for i:= 0 to lbxShares.Items.Count-1 do
    if lbxShares.Selected[i] then Break;  //Шукаємо обраний елемент
  if not lbxShares.Selected[i] then Exit; //Якщо не знайдений ідемо
  ShareName := lbxShares.Items.Strings[i];

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDelNT := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDelNT) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    i:= SizeOf(WideChar)*256;
    GetMem(NameNT,i);  //Виділяємо пам' ять під змінну
    StringToWideChar(ShareName,NameNT,i); //Перетворимо в PWideChar
    NetShareDelNT(nil,NameNT,0);   //Видаляємо ресурс
    FreeMem(NameNT);  //Звільняємо пам' ять
  end else begin  //Код для 9 х-ме
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareDel := GetProcAddress(FLibHandle,' NetShareDel' );
    if not Assigned(NetShareDel) then //Перевірка
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Name9x, SizeOf(Name9x), #0); //Очищаємо масив
    move(ShareName[1],Name9x[0],Length(ShareName)); //Заповнюємо масив
    NetShareDel(nil,@Name9x,0); //Видаляємо ресурс
  end;
  FreeLibrary(FLibHandle);
end;

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

function TMainForm.SelectDirectory: String;
var
  lpItemID : PItemIDList;
  BrowseInfo : TBrowseInfo;
  DisplayName : array[0..MAX_PATH] of Char;
  TempPath : array[0..MAX_PATH] of Char;
begin
  FillChar(BrowseInfo, sizeof(TBrowseInfo), #0);
  BrowseInfo.hwndOwner := Handle;
  BrowseInfo.pszDisplayName := @DisplayName;
  BrowseInfo.lpszTitle := ' Specify a directory' ;
  BrowseInfo.ulFlags := BIF_RETURNONLYFSDIRS;
  lpItemID := SHBrowseForFolder(BrowseInfo);
  if Assigned(lpItemId) then begin
    SHGetPathFromIDList(lpItemID, TempPath);
    GlobalFreePtr(lpItemID);
  end else Result := ' ' ;
  Result := String(TempPath);
```

```
end;

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

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

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

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAddNT := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAddNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    TmpLength := SizeOF(WideChar)*256; //Визначаємо необхідний розмір

    GetMem(TmpNameNT, TmpLength); //Конвертуємо в PWChar
    StringToWideChar(TmpName, TmpNameNT, TmpLength);
    ShareNT.shi2_netname := TmpNameNT; //Ім' я

    ShareNT.shi2_type := STYPE_DISKTREE; //Тип ресурсу
    ShareNT.shi2_remark := ' ' ; //Коментар
    ShareNT.shi2_permissions := ACCESS_ALL; //Доступ
    ShareNT.shi2_max_uses := DWORD(-1); // Кіл-ть максим. підключ.
    ShareNT.shi2_current_uses := 0; // Кіл-ть поточних підкл.

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

    ShareNT.shi2_passwd := ' ' ; //Пароль

    NetShareAddNT(nil,2,@ShareNT, nil); //Додаємо ресурс
    FreeMem (TmpNameNT); //звільняємо пам' ять
    FreeMem (TmpDirNT);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetShareAdd := GetProcAddress(FLibHandle,' NetShareAdd' );
    if not Assigned(NetShareAdd) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    FillChar(Share9x.shi50_netname, SizeOf(Share9x.shi50_netname), #0);
    move(TmpName[1],Share9x.shi50_netname[0],Length(TmpName)); //Ім' я
```

```
    Share9x.shi50_type := STYPE_DISKTREE; //Тип ресурсу
    Share9x.shi50_flags := SHI50F_FULL; //Доступ
    FillChar(Share9x.shi50_remark,
      SizeOf(Share9x.shi50_remark), #0); //Коментар
    FillChar(Share9x.shi50_path,
      SizeOf(Share9x.shi50_path), #0);
    Share9x.shi50_path := PAnsiChar(TmpDir); //Шлях до ресурсу
    FillChar(Share9x.shi50_rw_password,
      SizeOf(Share9x.shi50_rw_password), #0); //Пароль повного доступу
    FillChar(Share9x.shi50_ro_password,
      SizeOf(Share9x.shi50_ro_password), #0); //Пароль для читання
    NetShareAdd(nil,50,@Share9x,SizeOf(Share9x));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Помітьте що активний і неактивний час сесій буде даватися нам
//  у вигляді кіл-ті секунд (тип Cardinal). Напишемо невелику
//  функцію, задача якої буде перетворювати Кіл-ть секунд у більше
//  звичну форму відображення.
//

function TMainForm.CardinalToTimeStr(Value: Cardinal): String;
var d,h,m,s: Real;
begin
    d:=0;
    h:=0;
    m:=0;
    s:=Value;
    if s > 59 then begin
        m:=int(s / 60);
        s:= s-s-(m*60);
    end;
    if m > 59 then begin
        h:=int(m/60);
        m:= m-m-(h*60);
    end;
    if h > 23 then begin
        d:=int(h/24);
        h:= h-h-(d*24);
    end;
    Result:=' ' ;
    if (d>0) then Result:=Result+floattostr(d)+'  d. ' ;
    if (h<9) then Result:=Result+' 0' +floattostr(h)+' :'  else
Result:=Result+floattostr(h)+' :' ;
    if (m<9) then Result:=Result+' 0' +floattostr(m)+' :'  else
Result:=Result+floattostr(m)+' :' ;
    if (s<9) then Result:=Result+' 0' +floattostr(s)     else
Result:=Result+floattostr(s);
end;

////////////////////////////
//
//  Одержання списку сесій досліджуємої мережі Ethernet-фабрики Brocade VCS
систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)
//

procedure TMainForm.btnGetSessionsClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  SessionInfo50: array [0..512] of TSessionInfo50;
  SessionInfo502 : PSessionInfo502Array;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
```

```
  lvSessions.Items.Clear;

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

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionEnumNT := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnumNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    SessionInfo502 := nil;
    if NetSessionEnumNT(nil,nil,nil,502,@SessionInfo502,DWORD(-
1),@entriesreadNT, @totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo502^[i].sesi502_cname); //Ім' я комп' ютера
        SubItems.Add(SessionInfo502^[i].sesi502_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo502^[i].sesi502_num_opens));
//Відкритих ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].Sesi502_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo502^[i].sesi502_idle_time));
//Час не активний
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetSessionEnum := GetProcAddress(FLibHandle, ' NetSessionEnum' );
    if not Assigned(NetSessionEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetSessionEnum
(nil,50,@SessionInfo50,SizeOf(SessionInfo50),@EntriesRead,@TotalAvial) = 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvSessions.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(SessionInfo50[i].Sesi50_cname); //Ім' я комп' ютера
досліджуємої мережі Ethernet-фабрики Brocade VCS систем протидії DDoS-атакам
(Data_Center_Fibre_Channel_6)
        SubItems.Add(SessionInfo50[i].Sesi50_username);   //Ім' я користувача
        SubItems.Add(IntToStr(SessionInfo50[i].sesi50_num_opens)); //Відкритих
ресурсів
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].Sesi50_Time)); //Час
активний
        SubItems.Add(CardinalToTimeStr(SessionInfo50[i].sesi50_idle_time));
//Час не активний
        SessionCloseKey[i]:= SessionInfo50[i].sesi50_key; //Унікальний
ідентифікатор для закриття
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Завершення обраної сесії
//

procedure TMainForm.btnCloseSessionClick(Sender: TObject);
```

```
var
  OS: Boolean;
  FLibHandle : THandle;
  CNameNT: PWideChar;
  CName9x: PAnsiChar;
  Key:SmallInt;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvSessions.Selected) then Exit;
  i:= lvSessions.Selected.Index; //Визначаємо номер обраної сесії

  if OS then begin
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDelNT := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDelNT) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CNameNT := PWChar(WideString(' \\' +lvSessions.Items.Item[i].Caption));
    NetSessionDelNT(nil,CNameNT,nil);
  end else begin
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetSessionDel := GetProcAddress(FLibHandle, ' NetSessionDel' );
    if not Assigned(NetSessionDel) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
    //Перетворимо дані в необхідний вид
    CName9x := PAnsiChar(lvSessions.Items.Item[i].Caption);
    key := SessionCloseKey[i]; //Беремо ключ із масиву
    NetSessionDel(nil,CName9x,Key);
  end;
  FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Одержання списку відкритих файлів досліджуємої мережі Ethernet-фабрики
Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)
//

procedure TMainForm.btnGetFilesClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  FileInfoNT: PFileInfo3Array;
  FileInfo9x: array [0..512] of TFileInfo50;
  TotalEntries,EntriesReadNT: DWORD;
  EntriesRead,TotalAvial: Word;
  i:integer;
begin
  lvfiles.Items.Clear;

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

  if OS then begin  //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileEnumNT := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnumNT) then
    begin
      FreeLibrary(FLibHandle);
```

```
      Exit;
    end;
    FileInfoNT := nil;
    if NetFileEnumNT(nil,nil,nil,3,@FileInfoNT,DWORD(-1),@entriesreadNT,
@totalentries, nil)=0 then
    for i:=0 to EntriesReadNT-1 do
    begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfoNT^[i].fi3_id)); //Ідентифікатор
        SubItems.Add(FileInfoNT^[i].fi3_pathname);   //Шлях до файлу
        SubItems.Add(FileInfoNT^[i].fi3_username);   //Ім' я користувача
      end;
    end;
  end else begin //Код для Windows 9 x-Me
   FLibHandle := LoadLibrary(' SVRAPI.DLL' );
   if FLibHandle = 0 then Exit;
   @NetFileEnum := GetProcAddress(FLibHandle, ' NetFileEnum' );
    if not Assigned(NetFileEnum) then
    begin
      FreeLibrary(FLibHandle);
      Exit;
    end;
   if NetFileEnum (nil,
nil,50,@FileInfo9x,SizeOf(FileInfo9x),@EntriesRead,@TotalAvial)= 0 then
   for i:=0 to EntriesRead-1 do
   begin
      with lvFiles.Items.Add do  //Заповнення даними зі структури
      begin
        Caption := string(IntToStr(FileInfo9x[i].fi50_id)); //Ідентифікатор
        SubItems.Add(FileInfo9x[i].fi50_pathname); //Шлях до файлу
        SubItems.Add(FileInfo9x[i].fi50_username); //Ім' я користувача
      end;
   end;
 end;
 FreeLibrary(FLibHandle);
end;

//////////////////////////
//
//  Закриття файлу
//

procedure TMainForm.btnCloseFileClick(Sender: TObject);
var
  OS: Boolean;
  FLibHandle : THandle;
  i: Integer;
begin
  if not IsNT(OS) then Close; //З' ясовуємо тип системи

  if not Assigned(lvFiles.Selected) then Exit;
  i:= lvFiles.Selected.Index; //Визначаємо номер обраного файлу

  if OS then begin //Код для NT
    FLibHandle := LoadLibrary(' NETAPI32.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose := GetProcAddress(FLibHandle, ' NetFileClose' );
    if not Assigned(NetFileClose) then
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose(nil,StrToInt(lvFiles.Items.Item[i].Caption));//Закриваємо файл
  end else begin //Код для Windows 9 x-Me
    FLibHandle := LoadLibrary(' SVRAPI.DLL' );
    if FLibHandle = 0 then Exit;
    @NetFileClose2 := GetProcAddress(FLibHandle, ' NetFileClose2' );
    if not Assigned(NetFileClose2) then
```

```
    begin
      FreeLibrary(FLibHandle);
      Close;
    end;
    NetFileClose2(nil,StrToInt(lvFiles.Items.Item[i].Caption));
  end;
  FreeLibrary(FLibHandle);
end;

////////////////////////////
//
//  Визначаємо вхідний / вихідний трафік досліджуємої мережі Ethernet-фабрики
Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)
//

procedure TMainForm.tmrTrafficTimer(Sender: TObject);
  // Допоміжна функція, що перетворить MAC адресу до "нормального" виду
  //Визначаємо спеціальний тип, щоб можна було передати у функцію масив
  type TMAC = array [0..7] of Byte;
  //Як перше значення масив, друге значення, розмір даних у масиві
  function GetMAC(Value: TMAC; Length: DWORD): String;
  var
    i: Integer;
  begin
    if Length = 0 then Result := ‘  00-00-00’  else
    begin
      Result := ‘ ‘ ;
      for i:= 0 to Length-2 do
        Result := Result + IntToHex(Value[i],2)+’ -’ ;
      Result := Result + IntToHex(Value[ Length-1],2);
    end;
  end;

//Сама процедура
var
  FLibHandle : THandle;
  Table: TMibIfTable;
  i : integer;
  Size   : integer;
begin
  tmrTraffic.Enabled := false; //Припиняємо таймер
  lvTraffic.Items.BeginUpdate;
  lvTraffic.Items.Clear;  //Очищаємо список
  FLibHandle := LoadLibrary(‘ IPHLPAPI.DLL’ ); //Завантажуємо бібліотеку
  if FLibHandle = 0 then Exit;
  @GetIfTable := GetProcAddress(FLibHandle, ‘ GetIfTable’ );
  if not Assigned(GetIfTable) then
  begin
    FreeLibrary(FLibHandle);
    Close;
  end;

  Size := SizeOf(Table);
  if GetIfTable(@Table, @Size, false ) = 0 then //Виконуємо функцію
    for i:= 0 to Table.dwNumEntries-1 do begin
      with lvTraffic.Items.Add do begin //Виводимо результати
        Caption := String(Table.Table[i].bDescr); //Найменування інтерфейсу
        SubItems.Add(GetMAC(TMAC(Table.Table[i].bPhysAddr),
          Table.Table[i].dwPhysAddrLen)); //MAC адреса
        SubItems.Add(IntToStr(Table.Table[i].dwInOctets)); //Усього прийнято
байт з досліджуємої мережі Ethernet-фабрики Brocade VCS систем протидії DDoS-
атакам (Data_Center_Fibre_Channel_6)
        SubItems.Add(IntToStr(Table.Table[i].dwOutOctets)); //Усього відправлено
байт у досліджуєму мережу Ethernet-фабрики Brocade VCS систем протидії DDoS-
атакам (Data_Center_Fibre_Channel_6)
      end;
    end;
  lvTraffic.Items.EndUpdate;
  FreeLibrary(FLibHandle);
```

```
  tmrTraffic.Enabled := true; //Не забуваємо активувати таймер
end;


 function OpenEnum(const NetContainerToOpen: PNetResource; ResScope, ResType,
ResUsage: DWORD): THandle;
var
 hNetEnum: THandle;
begin
 Result:=0;
 if (NO_ERROR<>WNetOpenEnum(ResScope, ResType, ResUsage,
                                  NetContainerToOpen, hNetEnum))
 then ShowMessage(‘ Помилка!’ )
 else Result:=hNetEnum;
end;

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

const
 RESOURCE_BUF_ENTRIES = 2000;

var
 ResourceBuffer: array[1..RESOURCE_BUF_ENTRIES] of TNetResource;
 i, ResourceBuf, EntriesToGet: dword;
 NewNode: TTreeNode;
begin
 Result:=0;
 while true do
  begin
   ResourceBuf:=sizeof(ResourceBuffer);
   EntriesToGet:=RESOURCE_BUF_ENTRIES;
   if (NO_ERROR<>WNetEnumResource(hNetEnum, EntriesToGet,
                              @ResourceBuffer, ResourceBuf))
   then
    begin
     case GetLastError() of
      NO_ERROR:  // проход буферу без перемикання
      Break;
      ERROR_NO_MORE_ITEMS:
          // Повертає 0 у тому випадку, коли останов
          // RESOURCE_BUF_ENTRIES дані на попередньому виклику, щоб
          // WNetEnumResource, та були точно
          // RESOURCE_BUF_ENTRIES дані в запису на момент
          // попереднього виклику
      Exit;
      else ShowMessage(Помилка!’ );
      Result:=1;
      Exit;
     end;
    end;
   for i:=1 to EntriesToGet do
    begin
     NewNode:=ShowResource(ParentNode, ResourceBuffer[i]);
     if (ResourceBuffer[i].dwUsage and RESOURCEUSAGE_CONTAINER)<>0
     then MainForm.Open_Do_Close_Enum(NewNode, ResScope, ResType, ResUsage,
@ResourceBuffer[i]);
     Application.ProcessMessages;
    end;
  end;
end;
```

```
procedure TMainForm.Open_Do_Close_Enum(const ParentNode: TTreeNode; ResScope,
ResType, ResUsage: DWORD; const NetContainerToOpen: PNetResource);
var
 hNetEnum: THandle;
begin
 hNetEnum:=OpenEnum(NetContainerToOpen, ResScope, ResType, ResUsage);
 if (hNetEnum=0)
 then Exit;
 EnumResources(ParentNode, ResScope, ResType, ResUsage, hNetEnum);
 if (NO_ERROR<>WNetCloseEnum(hNetEnum))
 then ShowMessage(' WNetCloseEnum Помилка' );
end;

procedure TMainForm.Button1Click(Sender: TObject);
var
 ResScope, ResType, ResUsage: dword;
begin
 Button1.Caption:=' Пошук мережних ресурсів. Чекайте...' ;
 Button1.Enabled:=false;
 //
 NetTree.Items.Clear;
 case rgScope.ItemIndex of
  1: ResScope:=RESOURCE_GLOBALNET;
  2: ResScope:=RESOURCE_REMEMBERED;
  else ResScope:=RESOURCE_CONNECTED;
 end;
 ResType:=0;
 if cbTypeAny.Checked
 then ResType:=ResType or RESOURCETYPE_ANY;
 if cbTypeDisk.Checked
 then ResType:=ResType or RESOURCETYPE_DISK;
 if cbTypePrint.Checked
 then ResType:=ResType or RESOURCETYPE_PRINT;
 ResUsage:=0;
 if cbUsageConnectable.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONNECTABLE;
 if cbUsageContainer.Checked
 then ResUsage:=ResUsage or RESOURCEUSAGE_CONTAINER;
 Open_Do_Close_Enum(NetTree.Items.Add(nil, ' Network Resources' ),
                                ResScope, ResType, ResUsage, nil);
 //
 Button1.Caption:=' Обновити список ресурсів' ;
 Button1.Enabled:=true;

end;

procedure TMainForm.NetTreeCustomDrawItem(Sender: TCustomTreeView;
  Node: TTreeNode; State: TCustomDrawState; var DefaultDraw: Boolean);
begin
if cdsSelected in State
 then Sender.Canvas.Font.Style:=Sender.Canvas.Font.Style+[fsUnderline];
end;

procedure TMainForm.NetTreeDblClick(Sender: TObject);
begin
ShellExecute(0,' open' ,PChar(NetTree.Selected.Text),' ' ,' ' ,SW_SHOW);
end;

procedure TMainForm.NetTreeGetImageIndex(Sender: TObject; Node: TTreeNode);
begin
 if Node.HasChildren
 then Node.ImageIndex:=1
 else Node.ImageIndex:=0;
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
Form1.Show;
end;
```

```
procedure TMainForm.Button2Click(Sender: TObject);
begin
Form2.Show;
end;

procedure TMainForm.Button3Click(Sender: TObject);
begin
Form3.Show;
end;

end.
```

Файл IPHLPAPI.pas - обробка API функцій

```
unit IPHLPAPI;

interface
uses
  Windows, winsock;

const
  VERSION       = ' 1.5' ;

//------- Заголовок з Microsoft IPTYPES.H-H-----------------

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

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

  NETBIOSTypes  : array[0..8] of string[20] =
    ( ' Невизначений' , ' Передача' , ' Рівень до рівня' , ' ' , '
Змішаний' , ' ' , ' ' , ' ' , ' Гібрід'
      );

 // Типи адаптеру
{ v1.4-> 1.5
  IF_OTHER_ADAPTERTYPE = 0;
  IF_ETHERNET_ADAPTERTYPE = 1;
  IF_TOKEN_RING_ADAPTERTYPE = 2;
  IF_FDDI_ADAPTERTYPE = 3;
  IF_PPP_ADAPTERTYPE = 4;
  IF_LOOPBACK_ADAPTERTYPE = 5;
  IF_SLIP_ADAPTERTYPE = 6;

Знайдено у ipifcons.h :
#define MIB_IF_TYPE_OTHER               1
#define MIB_IF_TYPE_ETHERNET            6
#define MIB_IF_TYPE_TOKENRING           9
#define MIB_IF_TYPE_FDDI                15
#define MIB_IF_TYPE_PPP                 23
#define MIB_IF_TYPE_LOOPBACK            24
#define MIB_IF_TYPE_SLIP                28
}
  IF_OTHER_ADAPTERTYPE = 1;
  IF_ETHERNET_ADAPTERTYPE = 6;
  IF_TOKEN_RING_ADAPTERTYPE = 9;
  IF_FDDI_ADAPTERTYPE = 15;
  IF_PPP_ADAPTERTYPE = 23;
  IF_LOOPBACK_ADAPTERTYPE = 24;
  IF_SLIP_ADAPTERTYPE = 28;

//  AdaptTypes    : array[0..6] of string[10] =
//    ( ' інший' , ' ethernet' , ' tokenring' , ' FDDI' , ' PPP' , '
loopback' , ' SLIP'  );
  AdaptTypes    : array[1..28] of string[10] =
```

```
          ( ' інший' , ' ' , ' ' , ' ' , ' ' , ' ethernet' , ' ' , ' ' , '
tokenring' ,
           ' ' , ' ' , ' ' , ' ' , ' ' , ' FDDI' , ' ' , ' ' , ' ' , ' ' , '
' ' , ' ' , ' PPP' ,
           ' loopback' , ' ' , ' ' , ' ' , ' SLIP'  );
      // Кінець змін в типі адаптерів

      //-------для інших MS заготовочних файлів------------------

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

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

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

      //---IP адресні структури-------------------------

        PTIP_ADDRESS_STRING = ^TIP_ADDRESS_STRING;
        TIP_ADDRESS_STRING = array[0..15] of char; //  IP рядок
        //
        PTIP_ADDR_STRING = ^TIP_ADDR_STRING;
        TIP_ADDR_STRING = packed record // для використання у зв' язних списках
          Next: PTIP_ADDR_STRING;
          IpAddress: TIP_ADDRESS_STRING;
          IpMask: TIP_ADDRESS_STRING;
          Context: DWORD;
        end;

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

        PTFixedInfo = ^TFixedInfo;
        TFixedInfo = packed record
          HostName: array[1..MAX_HOSTNAME_LEN + 4] of char;    // дані
          DomainName: array[1..MAX_DOMAIN_NAME_LEN + 4] of char;   // дані
          CurrentDNSServer: PTIP_ADDR_STRING;
          DNSServerList: TIP_ADDR_STRING;
          NodeType: UINT;
          ScopeID: array[1..MAX_SCOPE_ID_LEN + 4] of char;   // дані
          EnableRouting: UINT;
          EnableProxy: UINT;
          EnableDNS: UINT;
        end;

      //-----структура мережного інтерфейсу досліджуємої мережі Ethernet-фабрики
Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)-----------
------------

      ////////////////////////
      //
//
      // Наступне є діючими станами для WAN да LAN інтерфейсів. //
      // Порядок станів створений для визначення. Для      //
      // стану >= CONNECTED можливо передавати дані зразу. Стан >= DISCONNECTED
//
      // може передавати деякі дані. Стан < DISCONNECTED може   //
      // не передавати дані.
//
      // карта з поміткою UNREACHABLE якщо DIM викликає InterfaceUnreachable для
//
      // причин. Крім невдачі з'єднання.                                  //
      //
//
      // NON_OPERATIONAL- Перевірка для LAN інтерфейсу. Позначає карту що не
працює       //
      //                      або не з'єднується з картою.        //
```

```
      // UNREACHABLE- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується за потрібний час.
//
      // DISCONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      не з'єднується.
      //
      // CONNECTING- Перевірка WAN інтерфейсів . Означає спробу з'єднання     //
      //                      з сайтом, якого немає.              //
      // CONNECTED- Перевірка WAN інтерфейсів . Позначає, що віддалений сайт
//
      //                      з'єднується.
//
      // OPERATIONAL- Перевірка LAN Interfaces. Позначає карту підключену //
      //                      в праці.                                  //
      //
//
      // Усі дії користувачів записуються до  MIB-II значення    //
      // можуть бути використовані                                        //
      //
//
      ////////////////////////

      const
      // дані додані до ipifcons.h
        IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        IF_OPER_STATUS_UNREACHABLE = 1 ;
        IF_OPER_STATUS_DISCONNECTED = 2 ;
        IF_OPER_STATUS_CONNECTING = 3 ;
        IF_OPER_STATUS_CONNECTED = 4 ;
        IF_OPER_STATUS_OPERATIONAL = 5 ;

        MIB_IF_TYPE_OTHER = 1 ;
        MIB_IF_TYPE_ETHERNET = 6 ;
        MIB_IF_TYPE_TOKENRING = 9 ;
        MIB_IF_TYPE_FDDI = 15 ;
        MIB_IF_TYPE_PPP = 23 ;
        MIB_IF_TYPE_LOOPBACK = 24 ;
        MIB_IF_TYPE_SLIP = 28 ;

        MIB_IF_ADMIN_STATUS_UP = 1 ;
        MIB_IF_ADMIN_STATUS_DOWN = 2 ;
        MIB_IF_ADMIN_STATUS_TESTING = 3 ;

        MIB_IF_OPER_STATUS_NON_OPERATIONAL = 0 ;
        MIB_IF_OPER_STATUS_UNREACHABLE = 1 ;
        MIB_IF_OPER_STATUS_DISCONNECTED = 2 ;
        MIB_IF_OPER_STATUS_CONNECTING = 3 ;
        MIB_IF_OPER_STATUS_CONNECTED = 4 ;
        MIB_IF_OPER_STATUS_OPERATIONAL = 5 ;

      type
        PTMibIfRow = ^TMibIfRow;
        TMibIfRow = packed record
          wszName: array[1..MAX_INTERFACE_NAME_LEN] of WCHAR;
          dwIndex: DWORD;
          dwType: DWORD;       // дивись MIB_IF_TYPE
          dwMTU: DWORD;
          dwSpeed: DWORD;
          dwPhysAddrLen: DWORD;
          bPhysAddr: array[1..MAXLEN_PHYSADDR] of byte;
          dwAdminStatus: DWORD;    // дивись MIB_IF_ADMIN_STATUS
          dwOperStatus: DWORD;     // дивись MIB_IF_OPER_STATUS
          dwLastChange: DWORD;
          dwInOctets: DWORD;
          dwInUcastPkts: DWORD;
          dwInNUCastPkts: DWORD;
          dwInDiscards: DWORD;
```

```
          dwInErrors: DWORD;
          dwInUnknownProtos: DWORD;
          dwOutOctets: DWORD;
          dwOutUCastPkts: DWORD;
          dwOutNUCastPkts: DWORD;
          dwOutDiscards: DWORD;
          dwOutErrors: DWORD;
          dwOutQLen: DWORD;
          dwDescrLen: DWORD;
          bDescr: array[1..MAXLEN_IFDESCR] of char; //byte;
        end;

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

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

        PTIP_ADAPTER_INFO = ^TIP_ADAPTER_INFO;
        TIP_ADAPTER_INFO = packed record
          Next: PTIP_ADAPTER_INFO;
          ComboIndex: DWORD;
          AdapterName: array[1..MAX_ADAPTER_NAME_LENGTH + 4] of char;       //
дані
          Description: array[1..MAX_ADAPTER_DESCRIPTION_LENGTH + 4] of char;
// дані
          AddressLength: UINT;
          Address: array[1..MAX_ADAPTER_ADDRESS_LENGTH] of byte;      // дані
          Index: DWORD;
          aType: UINT;
          DHCPEnabled: UINT;
          CurrentIPAddress: PTIP_ADDR_STRING;
          IPAddressList: TIP_ADDR_STRING;
          GatewayList: TIP_ADDR_STRING;
          DHCPServer: TIP_ADDR_STRING;
          HaveWINS: BOOL;
          PrimaryWINSServer: TIP_ADDR_STRING;
          SecondaryWINSServer: TIP_ADDR_STRING;
          LeaseObtained: LongInt ; // UNIX час, секунди з 1970
          LeaseExpires: LongInt;   // UNIX час, секунди з 1970
          SpareStuff: array [1..200] of char ;   // дані- простір для списку IP
адрес досліджуємої мережі Ethernet-фабрики Brocade VCS систем протидії DDoS-
атакам (Data_Center_Fibre_Channel_6)
        end;

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

        PTMibTCPRow = ^TMibTCPRow;
        TMibTCPRow = packed record
          dwState: DWORD;
          dwLocalAddr: DWORD;
          dwLocalPort: DWORD;
          dwRemoteAddr: DWORD;
          dwRemotePort: DWORD;
        end;
        //
        PTMibTCPTable = ^TMibTCPTable;
        TMibTCPTable = packed record
          dwNumEntries: DWORD;
          Table: array[0..0] of TMibTCPRow;
        end;
        //
        PTMibTCPStats = ^TMibTCPStats;
        TMibTCPStats = packed record
          dwRTOAlgorithm: DWORD;
          dwRTOMin: DWORD;
```

```
    dwRTOMax: DWORD;
    dwMaxConn: DWORD;
    dwActiveOpens: DWORD;
    dwPassiveOpens: DWORD;
    dwAttemptFails: DWORD;
    dwEstabResets: DWORD;
    dwCurrEstab: DWORD;
    dwInSegs: DWORD;
    dwOutSegs: DWORD;
    dwRetransSegs: DWORD;
    dwInErrs: DWORD;
    dwOutRsts: DWORD;
    dwNumConns: DWORD;
  end;

//-----UDP СТРУКТУРА -------------------------

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

//------IP СТРУКТУРА -------------------------

 //
  PTMibIPNetRow = ^TMibIPNetRow;
  TMibIPNetRow = packed record
    dwIndex: DWord;
    dwPhysAddrLen: DWord;
    bPhysAddr: TMACAddress;
    dwAddr: DWord;
    dwType: DWord;
  end;
  //
  PTMibIPNetTable = ^TMibIPNetTable;
  TMibIPNetTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPNetRow;
  end;
  //
  PTMibIPStats = ^TMibIPStats;
  TMibIPStats = packed record
    dwForwarding: DWORD;
    dwDefaultTTL: DWORD;
    dwInReceives: DWORD;
    dwInHdrErrors: DWORD;
    dwInAddrErrors: DWORD;
    dwForwDatagrams: DWORD;
    dwInUnknownProtos: DWORD;
    dwInDiscards: DWORD;
    dwInDelivers: DWORD;
    dwOutRequests: DWORD;
    dwRoutingDiscards: DWORD;
```

```
    dwOutDiscards: DWORD;
    dwOutNoRoutes: DWORD;
    dwReasmTimeOut: DWORD;
    dwReasmReqds: DWORD;
    dwReasmOKs: DWORD;
    dwReasmFails: DWORD;
    dwFragOKs: DWORD;
    dwFragFails: DWORD;
    dwFragCreates: DWORD;
    dwNumIf: DWORD;
    dwNumAddr: DWORD;
    dwNumRoutes: DWORD;
  end;
  //
  PTMibIPAddrRow = ^TMibIPAddrRow;
  TMibIPAddrRow = packed record
    dwAddr: DWORD;
    dwIndex: DWORD;
    dwMask: DWORD;
    dwBCastAddr: DWORD;
    dwReasmSize: DWORD;
    Unused1,
    Unused2: WORD;
  end;
  //
  PTMibIPAddrTable = ^TMibIPAddrTable;
  TMibIPAddrTable = packed record
    dwNumEntries: DWORD;
    Table: array[0..ANY_SIZE- 1] of TMibIPAddrRow;
  end;

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

//---- ICMP-СТРУКТУРА -----------------------

  PTMibICMPStats = ^TMibICMPStats;
  TMibICMPStats = packed record
    dwMsgs: DWORD;
    dwErrors: DWORD;
    dwDestUnreachs: DWORD;
    dwTimeEcxcds: DWORD;
    dwParmProbs: DWORD;
    dwSrcQuenchs: DWORD;
    dwRedirects: DWORD;
    dwEchos: DWORD;
    dwEchoReps: DWORD;
```

```
    dwTimeStamps: DWORD;
    dwTimeStampReps: DWORD;
    dwAddrMasks: DWORD;
    dwAddrReps: DWORD;
  end;

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

//---------імпорт до IPHLPAPI.DLL------------------

var

GetAdaptersInfo: function ( pAdapterInfo: PTIP_ADAPTER_INFO;
  pOutBufLen: PULONG ): DWORD; stdcall;

GetNetworkParams: function ( FixedInfo: PTFixedInfo; pOutPutLen: PULONG ):
        DWORD; stdcall;

GetTcpTable: function ( pTCPTable: PTMibTCPTable; pDWSize: PDWORD;
  bOrder: BOOL ): DWORD; stdcall;

GetTcpStatistics: function ( pStats: PTMibTCPStats ): DWORD; stdcall;

GetUdpTable: function ( pUdpTable: PTMibUDPTable; pDWSize: PDWORD;
 bOrder: BOOL ): DWORD; stdcall;

GetUdpStatistics: function ( pStats: PTMibUdpStats ): DWORD; stdcall;

GetIpStatistics: function ( pStats: PTMibIPStats ): DWORD; stdcall;

GetIpNetTable: function ( pIpNetTable: PTMibIPNetTable;
  pdwSize: PULONG;  bOrder: BOOL ): DWORD; stdcall;

GetIpAddrTable: function ( pIpAddrTable: PTMibIPAddrTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdcall;

GetIpForwardTable: function ( pIPForwardTable: PTMibIPForwardTable;
  pdwSize: PULONG; bOrder: BOOL ): DWORD; stdCall;

GetIcmpStatistics: function ( pStats: PTMibICMPInfo ): DWORD; stdCall;

GetRTTAndHopCount: function ( DestIPAddress: DWORD; HopCount: PULONG;
  MaxHops: ULONG; RTT: PULONG ): BOOL; stdCall;

GetIfTable: function ( pIfTable: PTMibIfTable; pdwSize: PULONG;
  bOrder: boolean ): DWORD; stdCall;

GetIfEntry: function ( pIfRow: PTMibIfRow ): DWORD; stdCall;

// попередження – недокументована функція, можливі помилки при
використанні
GetFriendlyIfIndex: function (var IfIndex: DWORD): DWORD; stdcall;


const
    IpHlpDLL = ' IPHLPAPI.DLL' ;
var
    IpHlpModule: THandle;

    function LoadIpHlp: Boolean;

implementation

function LoadIpHlp: Boolean;
begin
```

```
            Result := True;
            if IpHlpModule <> 0 then Exit;

        // відкрити DLL
            IpHlpModule := LoadLibrary (IpHlpDLL);
            if IpHlpModule = 0 then
            begin
                Result := false;
                exit ;
            end ;
            GetAdaptersInfo := GetProcAddress (IpHlpModule, ' GetAdaptersInfo' ) ;
            GetNetworkParams := GetProcAddress (IpHlpModule, ' GetNetworkParams' )
;
            GetTcpTable := GetProcAddress (IpHlpModule, ' GetTcpTable' ) ;
            GetTcpStatistics := GetProcAddress (IpHlpModule, ' GetTcpStatistics' )
;
            GetUdpTable := GetProcAddress (IpHlpModule, ' GetUdpTable' ) ;
            GetUdpStatistics := GetProcAddress (IpHlpModule, ' GetUdpStatistics' )
;
            GetIpStatistics := GetProcAddress (IpHlpModule, ' GetIpStatistics' ) ;
            GetIpNetTable := GetProcAddress (IpHlpModule, ' GetIpNetTable' ) ;
            GetIpAddrTable := GetProcAddress (IpHlpModule, ' GetIpAddrTable' ) ;
            GetIpForwardTable := GetProcAddress (IpHlpModule, ' GetIpForwardTable'
) ;
            GetIcmpStatistics := GetProcAddress (IpHlpModule, ' GetIcmpStatistics'
) ;
            GetRTTAndHopCount := GetProcAddress (IpHlpModule, ' GetRTTAndHopCount'
) ;
            GetIfTable := GetProcAddress (IpHlpModule, ' GetIfTable' ) ;
            GetIfEntry := GetProcAddress (IpHlpModule, ' GetIfEntry' ) ;
            GetFriendlyIfIndex := GetProcAddress (IpHlpModule, '
GetFriendlyIfIndex' ) ;
        end;

        initialization
            IpHlpModule := 0 ;
        finalization
            if IpHlpModule <> 0 then
            begin
                FreeLibrary (IpHlpModule) ;
                IpHlpModule := 0 ;
            end ;

        end.
```

Файл IPHelper.pas- функції роботи з IP-протоколом

```
unit IPHelper;


interface

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

const
  NULL_IP       = '   0.0.  0.0' ;

//---перетворення добре відомих номерів портів до імен сервісів--------

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


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

```
    );


//------перетворення ICMP кодів помилок до рядків-------------

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


//-----Перетворення різних перерахованих величин у рядки--------

  ARPEntryType  : array[1..4] of string = ( ' інший' , ' неправильний' ,
    ' динамічний' , ' статичний'
    );
  TCPConnState  :
    array[1..12] of string =
    ( ' closed' , ' listening' , ' syn_sent' ,
    ' syn_rcvd' , ' established' , ' fin_wait1' ,
    ' fin_wait2' , ' close_wait' , ' closing' ,
    ' last_ack' , ' time_wait' , ' delete_tcb'
    );

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

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

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

type

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

  TIfRows = array of TMibIfRow ; // динамічний масив колонок

// для IpHlpAdaptersInfo
  TAdaptorInfo = record
    AdapterName: string ;
```

```
    Description: string ;
    MacAddress: string ;
    Index: DWORD;
    aType: UINT;
    DHCPEnabled: UINT;
    CurrIPAddress: string ;
    CurrIPMask: string ;
    IPAddressTot: integer ;
    IPAddressList: array of string ;
    IPMaskList: array of string ;
    GatewayTot: integer ;
    GatewayList: array of string ;
    DHCPTot: integer ;
    DHCPServer: array of string ;
    HaveWINS: BOOL;
    PrimWINSTot: integer ;
    PrimWINSServer: array of string ;
    SecWINSTot: integer ;
    SecWINSServer: array of string ;
    LeaseObtained: LongInt ; // UNIX час, секунди з 1970
    LeaseExpires: LongInt;   // UNIX час, секунди з 1970
  end ;

  TAdaptorRows = array of TAdaptorInfo ;


//--------експортуємі дані---------------------

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

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

implementation

var
  RecentIPs     : TStringList;

//-------Основні функції----------------------

{ отримання наступного "токена" з рядка }
```

```
function NextToken( var s: string; Separator: char ): string;
var
  Sep_Pos       : byte;
begin
  Result := ' ' ;
  if length( s ) > 0 then begin
    Sep_Pos := pos( Separator, s );
    if Sep_Pos > 0 then begin
      Result := copy( s, 1, Pred( Sep_Pos ) );
      Delete( s, 1, Sep_Pos );
    end
    else begin
      Result := s;
      s := ' ' ;
    end;
  end;
end;

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

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

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

end;
```

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

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

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

//-----------------------------------
{ голона частина, фіксація мережний параметрів досліджуємої мережі Ethernet-
фабрики Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)}

procedure Get_NetworkParams( List: TStrings );
var
    NetworkParams: TNetworkParams ;
    I, ErrorCode: integer ;
begin
    if not Assigned( List ) then EXIT;
    List.Clear;
    ErrorCode := IpHlpNetworkParams (NetworkParams) ;
    if ErrorCode <> 0 then
    begin
        List.Add (SysErrorMessage (ErrorCode));
        exit;
    end ;
    with NetworkParams do
    begin
        List.Add( ' Ім'я хосту          : '  + HostName );
        List.Add( ' Домен            : '  + DomainName );
        List.Add( ' NETBIOS тип : '  + NETBIOSTypes[NodeType] );
        List.Add( ' DHCP область        : '  + ScopeID );
        List.Add( ' ROUTING визначено   : '  + IntToStr( EnableRouting ) );
        List.Add( ' PROXY   визначено   : '  + IntToStr( EnableProxy ) );
        List.Add( ' DNS     визначено   : '  + IntToStr( EnableDNS ) );
        if DnsServerTot <> 0 then
        begin
            for I := 0 to Pred (DnsServerTot) do
                List.Add( ' DNS адреса серверу   : '  + DnsServerNames [I] ) ;
        end ;
    end ;
end ;

//-------------------------------------//
function IpHlpNetworkParams (var NetworkParams: TNetworkParams): integer ;
var
  FixedInfo     : PTFixedInfo;         // дані
  InfoSize      : Longint;
  PDnsServer    : PTIP_ADDR_STRING ;   // дані
```

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

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

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


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

// включення байтів у/з для кожного адаптера

function IpHlpIfTable(var IfTot: integer; var IfRows: TIfRows): integer ;
var
  I,
  TableSize   : integer;
  pBuf, pNext : PChar;
begin
  result := ERROR_NOT_SUPPORTED ;
  if NOT LoadIpHlp then exit ;
  SetLength (IfRows, 0) ;
  IfTot := 0 ; // дані
  TableSize := 0;
   // перший виклик: необхідно отримати розмір пам' яті
  result := GetIfTable (Nil, @TableSize, false) ;  // дані
  if result <> ERROR_INSUFFICIENT_BUFFER then exit ;
  GetMem( pBuf, TableSize );
```

```
  try
      FillChar (pBuf^, TableSize, #0);  // очищаємо буфер з  W98 не беремо
крапку таблиці
      result := GetIfTable (PTMibIfTable (pBuf), @TableSize, false) ;
      if result <> NO_ERROR then exit ;
      IfTot := PTMibIfTable (pBuf)^.dwNumEntries ;
      if IfTot = 0 then exit ;
      SetLength (IfRows, IfTot) ;
      pNext := pBuf + SizeOf(IfTot) ;
      for i := 0 to Pred (IfTot) do
      begin
         IfRows [i] := PTMibIfRow (pNext )^ ;
         inc (pNext, SizeOf (TMibIfRow)) ;
      end;
  finally
      FreeMem (pBuf) ;
  end ;
end;

procedure Get_IfTable( List: TStrings );
var
  IfRows        : TIfRows ;
  Error, I      : integer;
  NumEntries    : integer;
  sDescr, sIfName: string ;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
  SetLength (IfRows, 0) ;
  Error := IpHlpIfTable (NumEntries, IfRows) ;
  if (Error <> 0) then
      List.Add( SysErrorMessage( GetLastError ) )
  else if NumEntries = 0 then
      List.Add( ' данних немає ' )
  else
  begin
      for I := 0 to Pred (NumEntries) do
      begin
          with IfRows [I] do
          begin
             if wszName [1] = #0 then
                 sIfName := ' '
             else
                 sIfName := WideCharToString (@wszName) ;  // конвертуємо Юнікод
до рядка
             sIfName := trim (sIfName) ;
             sDescr := bDescr ;
             sDescr := trim (sDescr);
             List.Add (Format (
              ' %0.8x |%3d | %16s |%8d |%12d |%2d |%2d |%10d |%10d | %-s| %-s'
,
               [dwIndex, dwType, MacAddr2Str( TMacAddress( bPhysAddr ),
               dwPhysAddrLen ), dwMTU, dwSpeed, dwAdminStatus,
               dwOPerStatus, Int64 (dwInOctets), Int64 (dwOutOctets),  //
конвертуємо до 32-біт
               sIfName, sDescr] )  // дані, додані в/з
               );
          end;
      end ;
  end ;
  SetLength (IfRows, 0) ;  // вільна пам' ять
end ;

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

```
  result := GetIfEntry (@IfRow) ;
end ;

//-----------------------------------
{ інформація про інстальовані адаптери }

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

        // беремо список IP адрес та конвертуємо в IPAddressList
            I := 0 ;
            PIpAddr := @AdapterInfo^.IPAddressList ;
            while (PIpAddr <> Nil) do
            begin
                AdpRows [AdpTot].IPAddressList [I] := PIpAddr.IpAddress ;
                AdpRows [AdpTot].IPMaskList [I] := PIpAddr.IpMask ;
                PIpAddr := PIpAddr.Next ;
                inc (I) ;
```

```
            if Length (AdpRows [AdpTot].IPAddressList) <= I then
            begin
                 SetLength (AdpRows [AdpTot].IPAddressList, I -2) ;
                 SetLength (AdpRows [AdpTot].IPMaskList, I -2) ;
            end ;
        end ;
        AdpRows [AdpTot].IPAddressTot := I ;

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

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

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

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

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

        inc (AdpTot) ;
        if Length (AdpRows) <= AdpTot then
                        SetLength (AdpRows, AdpTot -2) ;  // більше пам' яті
        AdapterInfo := AdapterInfo^.Next;
    end ;
    SetLength (AdpRows, AdpTot) ;
```

```
      end ;
  finally
      FreeMem( pBuf );
  end ;
end ;

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

//-----------------------------------
{ моніторимо час доступу до IP досліджуємої мережі Ethernet-фабрики Brocade VCS
систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)}
function Get_RTTAndHopCount( IPAddr: DWORD; MaxHops: Longint; var RTT: Longint;
  var HopCount: Longint ): integer;
begin
  if not GetRTTAndHopCount( IPAddr, @HopCount, MaxHops, @RTT ) then
  begin
    Result := GetLastError;
    RTT :=-1; // Расположення BAD_HOST_NAME,etc...
    HopCount :=-1;
  end
  else
    Result := NO_ERROR;
end;

//-----------------------------------
{ ARP-таблиця включає відношення між віддаленим IP та віддаленим MAC-адресом.
}
```

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

  // необхідно відновити показник!
  finally
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( IPNetRow ) );
    FreeMem( pBuf );
  end ;
end;


//-----------------------------------
procedure Get_TCPTable( List: TStrings );
var
  TCPRow        : TMIBTCPRow;
  i,
    NumEntries  : integer;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  DestIP        : string;
  pBuf          : PChar;
begin
  if not Assigned( List ) then EXIT;
  List.Clear;
```

```
  RecentIPs.Clear;
  // перший виклик: беремо довжину таблиці
  TableSize := 0;
  NumEntries := 0 ;
  ErrorCode := GetTCPTable(Nil, @TableSize, false );  // дані
  if Errorcode <> ERROR_INSUFFICIENT_BUFFER then
    EXIT;

  // беремо розмір пам' яті, викликаємо знову
  GetMem( pBuf, TableSize );
  // беремо таблицю
  ErrorCode := GetTCPTable( PTMIBTCPTable( pBuf ), @TableSize, false );
  if ErrorCode = NO_ERROR then
  begin

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

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

```
      List.Add( ' Максимальне число підключень : '  + IntToStr( dwRTOAlgorithm )
);
      List.Add( ' Активні підключення             : '  + IntToStr( dwActiveOpens
) );
      List.Add( ' пасивні підключення            : '  + IntToStr( dwPassiveOpens
) );
      List.Add( ' Невдала спроба відкриття     : '  + IntToStr( dwAttemptFails )
);
      List.Add( ' Скидання встановленого підключення  : '  + IntToStr(
dwEstabResets ) );
      List.Add( ' Поточне встановлене підключення.: '  + IntToStr( dwCurrEstab )
);
      List.Add( ' Отримані сегменти        : '  + IntToStr( dwInSegs ) );
      List.Add( ' Передані сегменти            : '  + IntToStr( dwOutSegs ) );
      List.Add( ' Перепідключені сегменти   : '  + IntToStr( dwReTransSegs ) );
      List.Add( ' помилка входу          : '  + IntToStr( dwInErrs ) );
      List.Add( ' Перезавантаження вихідних          : '  + IntToStr( dwOutRsts
) );
      List.Add( ' Совокупні зв' язки    : '  + IntToStr( dwNumConns ) );
    end
  else
    List.Add( SyserrorMessage( ErrorCode ) );
end;

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

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

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

  // виділяємо пам' ять, викликаємо знову
  GetMem( pBuf, TableSize );

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

```
              ] ) );
          inc( pBuf, SizeOf( TMIBUDPRow ) );
        end;
      end
      else
        List.Add( ' немає даних.'  );
    end
    else
      List.Add( SyserrorMessage( ErrorCode ) );
    dec( pBuf, SizeOf( DWORD ) + NumEntries -SizeOf( TMibUDPRow ) );
    FreeMem( pBuf );
end;

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

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

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

//-----------------------------------
{ отримуємо дані з таблиці маршрутизації досліджуємої мережі Ethernet-фабрики
Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6); }
```

```
procedure Get_IPForwardTable( List: TStrings );
var
  IPForwRow     : TMibIPForwardRow;
  TableSize     : DWORD;
  ErrorCode     : DWORD;
  i             : integer;
  pBuf          : PChar;
  NumEntries    : DWORD;
begin

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

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

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

//-----------------------------------
procedure Get_IPStatistics( List: TStrings );
var
  IPStats       : TMibIPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
```

```
  if NOT LoadIpHlp then exit ;
  ErrorCode := GetIPStatistics( @IPStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with IPStats do
    begin
      if dwForwarding = 1 then
        List.add( ' Розблокована пересилка      : '  + ' так'  )
      else
        List.add(' Розблокована пересилка      : '  + ' ні'  );
      List.add( ' Любий TTL             : '  + inttostr( dwDefaultTTL ) );
      List.add( ' Датаграмма прийнята     : '  + inttostr( dwInReceives ) );
      List.add( ' Помилка заголовку      (In)  : '  + inttostr( dwInHdrErrors )
);
      List.add( ' Помилка адреси    (In)  : '  + inttostr( dwInAddrErrors ) );
      List.add( ' Датаграма переслана     : '  + inttostr( dwForwDatagrams ) );
// дані
      List.add( ' Невизначений протокол (In)  : '  + inttostr( dwInUnknownProtos
) );
      List.add( ' Датаграмма відмовлена     : '  + inttostr( dwInDiscards ) );
      List.add( ' Датаграма встановлена     : '  + inttostr( dwInDelivers ) );
      List.add( ' Зовнішній запрос            : '  + inttostr( dwOutRequests )
);
      List.add( ' Маршрутизація не виконана      : '  + inttostr(
dwRoutingDiscards ) );
      List.add( ' Немає маршрутів        (Out)  : '  + inttostr( dwOutNoRoutes )
);
      List.add( ' Перебраний час     : '  + inttostr( dwReasmTimeOut ) );
      List.add( ' Запит перебору     : '  + inttostr( dwReasmReqds ) );
      List.add( ' Повний перебор : '  + inttostr( dwReasmOKs ) );
      List.add( ' Помилка перебору     : '  + inttostr( dwReasmFails ) );
      List.add( ' Повна фрагментація: '  + inttostr( dwFragOKs ) );
      List.add( ' Помилка фрагментації   : '  + inttostr( dwFragFails ) );
      List.add( ' Датаграмма фрагментована    : '  + inttostr( dwFRagCreates )
);
      List.add( ' Кількість інтерфейсів    : '  + inttostr( dwNumIf ) );
      List.add( ' Кількість IP-адрес  : '  + inttostr( dwNumAddr ) );
      List.add( ' Маршрут в таблиці маршрутизатора  : '  + inttostr( dwNumRoutes
) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

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

//-----------------------------------
procedure Get_UdpStatistics( List: TStrings );
var
  UdpStats      : TMibUDPStats;
  ErrorCode     : integer;
begin
  if not Assigned( List ) then EXIT;
  ErrorCode := GetUDPStatistics( @UdpStats );
  if ErrorCode = NO_ERROR then
  begin
    List.Clear;
    with UDPStats do
    begin
      List.add( ' Датаграми (In)     : '  + inttostr( dwInDatagrams ) );
      List.add( '  Датаграми (Out)    : '  + inttostr( dwOutDatagrams ) );
      List.add( ' Немає портів          : '  + inttostr( dwNoPorts ) );
```

```
      List.add( ' Помилка    (In)    : '  + inttostr( dwInErrors ) );
      List.add( ' UDP список портів  : '  + inttostr( dwNumAddrs ) );
    end;
  end
  else
    List.Add( SysErrorMessage( ErrorCode ) );
end;

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

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

```
  else
    IcmpIn.Add( SysErrorMessage( ErrorCode ) );
  Dispose( ICMPStats );
end;

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

initialization

  RecentIPs := TStringList.Create;

finalization

  RecentIPs.Free;

end.
```

Файл TCP_IP.pas- монітор TCP/IP з'єднань досліджуємої мережі Ethernet-фабрики Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)

```
unit TCP_IP;

interface

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

type
  TForm2 = class(TForm)
    StaticText2: TStaticText;
    StaticText3: TStaticText;
    TCPMemo: TMemo;
    UDPMemo: TMemo;
    Timer1: TTimer;
    cbTimer: TCheckBox;
    btRTTI: TSpeedButton;
    SpeedButton1: TSpeedButton;
    edtRTTI: TEdit;
    procedure Timer1Timer(Sender: TObject);
    procedure SpeedButton1Click(Sender: TObject);
    procedure btRTTIClick(Sender: TObject);
    procedure cbRecentIPsClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
      procedure DOIpStuff;
  public
    { Public declarations }
  end;

var
  Form2: TForm2;

implementation

{$R *.dfm}

procedure TForm2.Timer1Timer(Sender: TObject);
begin
  if cbTimer.State = cbCHECKED then
 begin
   Timer1.Enabled := false;
   DoIPStuff;
   Timer1.Enabled := true;
 end;
end;

procedure TForm2.DOIpStuff;
begin

  Get_TCPTable( TCPMemo.Lines );
  Get_UDPTable( UDPMemo.Lines );

end;

procedure TForm2.SpeedButton1Click(Sender: TObject);
begin
  Speedbutton1.Enabled := false;
  DoIPStuff;
  Speedbutton1.Enabled := true;
end;

procedure TForm2.btRTTIClick(Sender: TObject);
```

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

end;

procedure TForm2.cbRecentIPsClick(Sender: TObject);
begin
//edtRTTI.Text := cbRecentIPs.Items[cbRecentIPs.ItemIndex];
end;

procedure TForm2.FormCreate(Sender: TObject);
begin

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

end.
```

Файл Stat.pas- статистика досліджуємої мережі Ethernet-фабрики Brocade VCS систем протидії DDoS-атакам (Data_Center_Fibre_Channel_6)

```
unit Stat;

interface

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

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

var
  Form3: TForm3;

implementation

{$R *.dfm}

procedure TForm3.DOIpStuff;
begin

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

end;

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

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

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

end;

procedure TForm3.FormCreate(Sender: TObject);
begin

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

end.
```

Файл About.pas - довідка

```
unit About;

interface

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

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

var
  Form1: TForm1;

implementation

{$R *.dfm}

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

end.
```