Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

# ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
**за другим (магістерським) рівнем вищої освіти**
на тему
**“Дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6”**



Виконав здобувач вищої освіти
ІІ курсу, групи КІ-23М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Барабаш А.І.
«____» _____ 2024 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 2024 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 2024 року

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

*Барабашу Артему Івановичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 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 № 19-13 від 07.08.2024 року

3. Строк подання студентом роботи до захисту *2.12.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6*

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

| | |
|---|---|
| *1. Призначення та область використання.* | *6. Наукова новизна.* |
| *2.Перегляд аналогічних існуючих систем.* | *7. Маркетингове та економічне обґрунтування ІТ-проєкту.* |
| *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. 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 | Доренська А.О. | 05.10.2024 | 14.11.2024 |
| Охорона праці | Марченко К.М., к.т.н., доцент | 06.10.2024 | 16.11.2024 |
| | | | |

7. Дата видачі завдання « 6 » вересня 2024 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 | 10.10.2024 р. | |
| 2. | Постановка задачі, оформлення ТЗ | 15.10.2024 р. | |
| 3. | Розробка моделі компонента | 20.10.2024 р. | |
| 4. | Розробка структур даних | 25.10.2024 р. | |
| 5. | Розробка алгоритмів зв’язку та відображення | 30.10.2024 р. | |
| 6. | Програмування алгоритмів | 10.11.2024 р. | |
| 7. | Розрахунок економічної ефективності | 13.11.2024 р. | |
| 8. | Розрахунки з охорони праці та техніки безпеки | 15.11.2024 р. | |
| 9. | Оформлення ПЗ | 17.11.2024 р. | |
| 10. | Попередній захист роботи | 2.12.2024 р. | |
| | | | |



Дата видачі завдання
« 6 » вересня 2024 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2024 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

# АНОТАЦІЯ

**Барабаш А.І. Дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.**

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мереж SAN на основі стандарту Fibre Channel Gen 6.

Метою розробки є дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6.

Об'єктом дослідження є процес мереж SAN на основі стандарту Fibre Channel Gen 6.

Предметом дослідження є методи мереж SAN на основі стандарту Fibre Channel Gen 6.

Методи дослідження базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 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 з ОС Windows 10/11.

Програму розроблено в середовищі Python.

**Ключові слова**: комп'ютерна інженерія, SAN, Fibre Channel Gen 6

# ABSTRACT

**Barabash A.I. Research and software implementation of the SAN network system based on the standard Fiber Channel Gen 6. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.**

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the SAN network system based on the Fiber Channel Gen 6 standard.

The purpose of the development is the research and software implementation of the SAN network system based on the Fiber Channel Gen 6 standard.

The object of research is the process of SAN networks based on the Fiber Channel Gen 6 standard.

The subject of the study is the methods of SAN networks based on the Fiber Channel Gen 6 standard.

The research methods are based on the methods of the theory of building computer networks, the methods of mathematical statistics, and the methods of software development.

The result of the work is the software implementation of the SAN network system based on the Fiber Channel Gen 6 standard.

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

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

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

The program was developed in the Python environment.

**Keywords:** computer engineering, SAN, Fiber Channel Gen 6

**ЗМІСТ**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ ..... 3
ВСТУП..... 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ ..... 7
1.1 Призначення системи..... 7
1.2 Область застосування..... 8
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ ..... 9
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти..... 9
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування..... 11
2.3 Розгорнута постановка завдання ..... 14
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ ..... 15
3.1 Опис функціонування системи ..... 15
3.2 Розробка структурної схеми..... 38
3.3 Розробка функціональної схеми ..... 45
3.4 Розробка діаграми процесів..... 50
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ..... 52
4.1 Розробка блок-схем та опис алгоритмів функціонування системи..... 52
4.2 Захист розробленого програмного забезпечення..... 65
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ..... 68
6 НАУКОВА НОВИЗНА ..... 72

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ ..... 73
7.1 Визначення цільової аудиторії кінцевого готового продукту ..... 73
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок ... 74
7.3 Вибір методу оцінки вартості ПЗ ..... 75
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості ..... 75
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ ..... 78
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ ..... 80
7.7 Визначення ключових факторів успіху конкретного проєкту ..... 81
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ ..... 82
8.1 Вступ ..... 82
8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 83
8.3 Розробка заходів з умов поліпшення охорони праці ..... 85
8.4 Техніка безпеки та протипожежна профілактика ..... 86
8.5 Розрахункова частина ..... 88
9 ОСНОВНІ ВИСНОВКИ ..... 91
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... 93

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

| | | |
|---|---|---|
| КВ | – | коефіцієнт варіації |
| КЗ | – | канал зв'язку |
| ПС | – | програмна середа |
| СеМО | – | експонентна мережа масового обслуговування |
| СМО | – | система масового обслуговування |
| СПД | – | система передачі даних |

# ВСТУП

**Актуальність теми**. Ріст попиту на встаткування мереж зберігання даних (SAN) стимулюється потребою в розгортанні високошвидкісних мереж зберігання, здатних обслуговувати системи зберігання даних (СЗД) на флеш-пам'яті й віртуальні середовища. Для цього розроблені Fibre Channel нового покоління.

Продажі мережного встаткування для мереж зберігання даних (SAN), включаючи комутатори Fibre Channel і адаптери HBA (Host Bus Adapter), продовжують рости.

Обсяг продажів конвергентних мережних адаптерів (Converged Network Adapter, CNA) збільшився за останній квартал на 13%. Виросли й поставки адаптерів iSCSI. На ринку комутаторів і адаптерів Fibre Channel лідирує Brocade (торік вона передала свою технологію серверних HBA компанії Qlogic), а в сегменті адаптерів CNA для ЦОД – Intel. На частку продукції Brocade доводиться близько 75% світового ринку FC по кількості поставлених портів і приблизно 50% по обсязі продажів у грошовому вирахуванні.

У найближчі три роки, за прогнозом IDC, ємність систем зберігання FC у світі буде становити приблизно 38% від загальної ємності СЗД різного типу, тоді як частка систем NAS – близько 30%, при цьому першим належать провідні місця в тестах на продуктивність СЗД. Як очікується, попит на продукти FC буде рости: по показнику ємності середні темпи росту поставок систем зберігання даних Fibre Channel складуть 36%. Що стосується поточного року, то, відповідно до розрахунків Infonetics, продажу мережного встаткування FC виростуть майже на 10%.

**Мета й завдання дослідження.** Метою роботи є дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6.

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

– Огляд існуючих систем мереж SAN на основі стандарту Fibre Channel Gen 6.

– Дослідження системи мереж SAN на основі стандарту Fibre Channel Gen 6.

– Програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6.

*Об'єктом дослідження* є процес мереж SAN на основі стандарту Fibre Channel Gen 6.

*Предметом дослідження* є методи мереж SAN на основі стандарту Fibre Channel Gen 6.

*Методи дослідження* базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод мереж SAN на основі стандарту Fibre Channel Gen 6.

– Розроблено вітчизняний продукт мереж SAN на основі стандарту Fibre Channel Gen 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 мереж SAN на основі стандарту Fibre Channel Gen 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 LVII Науково-технічній конференції здобувачів вищої освіти LV науково-технічної конференції «Наука в ЦНТУ: основні досягнення та перспективи розвитку» (2024 р.), основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №15.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 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 даними опитування Enterprise Strategy Group (ESG), 85% компаній планують зберегти або збільшити інвестиції в Fibre Channel SAN. Як переваги цього протоколу вони відзначають високу продуктивність і надійність. Згідно IDC, більше 90% компаній зі списку Fortune1000 використовують у своїх ЦОД Fibre Channel як стандарт де-факто. Виділена мережа зберігання SAN залишається оптимальним варіантом для високошвидкісного доступу до даних, оскільки її традиційний протокол Fibre Channel споконвічно розрахований на швидку передачу більших блоків і низькі затримки. Важливим фактором росту ринку SAN стане перехід на встаткування нового покоління – Fibre Channel Gen 6 (32 Гбіт/с).

У підсистемах уведення-виводу СЗД будуть ще ширше використовуються високопродуктивні мережні адаптери Ethernet, з переходом на FC нового покоління. Цього вимагає й активне використання в СЗД швидкої флеш-пам'яті й накопичувачів SSD – канали взаємодії між пристроями не повинні ставати вузьким місцем. Наприклад, швидкість передачі даних у мережі 10Gb становить близько 1,25 Гбайт/с, або приблизно 4,5 Тбайт/годину. У той же час навіть простий накопичувач SSD дозволяє читати дані зі швидкістю 500 Мбайт/с, що становить майже половину номінальної пропускної здатності мережі 10Gb. Проблема не тільки у високій швидкості флеш-пам'яті, але й у ємності, що збільшується, дискових накопичувачів (HDD), що досягла 6 Тбайт на диск. Резервне копіювання дискових масивів стає складним завданням.

## 1.2 Область застосування

Серед інших драйверів росту потреби в портах SAN – віртуалізація серверів, збільшення обсягів збережених даних (у середньому на 40% щорічно), віртуалізація ресурсів зберігання й хмарні сервіси. Зараз в основному підтримується п'яте покоління FC зі швидкостями передачі даних 10 і 16 Гбіт/с. Стратегія розроблювачів в області FC спрямована на зниження операційних витрат і збільшення часу безперебійної роботи, підвищення продуктивності й надійності, на підтримку засобами FC впровадження SSD і SDDC. Однак на ринку з'явилися комутатори FC і директори шостого покоління, і на основі таких рішень уже розробляється архітектура ЦОД.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 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

**Рішення від Сторус на базі концентратора сховищ Stonefly**

Сторус пропонує новітній концентратор компанії Stonefly Networks для організації SAN у мережах IP. Концентратор не тільки дозволяє здійснювати підключення зовнішніх дискових масивів і організацію їх у мережні ресурси, але й віртуалізацію існуючих усередині серверів дисків у єдиний зовнішній дисковий простір. Концентратор використовує протокол iSCSI (Internet Small Computer Systems Interface). iSCSI дозволяє передавати SCSI команди усередині пакета TCP/IP. Оскільки більшість організацій мають мережа Ethernet, те IP SAN ставати природним засобом для підключення до цієї мережі зовнішніх сховищ.

Концентратор Stonefly являє собою пристрій висотою 1U, монтуємий в стійку.

**Адаптери QLogic стандарту FibreChannel і iSCSI**

Інфраструктура FibreChannel складається з прикінцевих пристроїв (серверів, дискових стійок, стрічкових бібліотек з FC-Інтерфейсами) і комутаторів для об'єднання в мережу. Для зв'язку пристроїв між собою застосовуються оптичні багато- і одномодові кабелі. Інтерфейси FibreChannel звичайно убудовані в дискові стійки й стримерні бібліотеки; для зв'язку серверів з SAN-Інфраструктурою використовуються спеціальні адаптери.

Адаптери стандарту FibreChannel виробництва QLogic установлюються в сервера в рознімання PCI-X або PCI Express, містять 1, 2 або 4 FC-порти зі швидкістю передачі даних 4 або 8 гігабіт у секунду.

Адаптери QLogic стандарту iSCSI виконують роль апаратних ініціаторів для зв'язку з iSCSI пристроями через стандартну мережу Gigabit Ethernet.

**Комутатори QLogic стандарту FibreChannel**

Комутатори стандарту FibreChannel поєднують у мережу SAN-архітектури сервери й пристрої зберігання даних (дискові стійки, стрічкові бібліотеки).

Для зв'язку пристроїв FibreChannel між собою застосовуються оптичні багато- і одномодові кабелі.

Комутатори FibreChannel виробництва QLogic випускаються у варіантах як з фіксованою конфігурацією, так і з можливістю розширення для набору необхідної кількості портів FibreChannel. Комутатори також можуть поєднуватися в стек для створення комутуючої матриці на велику кількість портів з високою пропускною здатністю й малим часом затримки при передачі даних.

**QLogic SANBox 1400 – комутатори початкового рівня**

– 10 фіксованих портів FibreChannel на швидкості 4 Gbit.

– Затримка комутації – не більше 0.4 ms (пікова), пропускна сумарна здатність – більше 80 Gb/s (full duplex), що не блокує архітектура.

– Інтерфейси – стандартні SFP модулі (Small Form Pluggable) з гарячою заміною на багатомодовий кабель 50/62.5 micron (до 500 метрів) або одномодовий кабель 9 micron (до 10км).

– Висота 1U + спеціальні вушка для установки в 19-дюймову шафу.

**QLogic SANBox 5600Q – стекуємі комутатори стандарту FC 4GBit**

– 8, 12 або 16 FC-портів з автовизначенням швидкості на 4/2/1Gbps (розширення ліцензіями по 4 порти).

– Об'єднання в стек до 8 пристроїв (до 96 FC-портів) – через швидкісні 20 гігабітні з'єднання ISL (Inter-Switch Links).

– Моделі з одним убудованим блоком живлення або з 2мя блоками з гарячою заміною.

– Сполучення по типі " full-fabric"," public-loop", а також " комутатор-комутатор" на всіх портах, включаючи 10 Gbit-ні.

– Інтерфейси – стандартні SFP модулі (Small Form Pluggable) з гарячою заміною на багатомодовий кабель 50/62.5 micron (до 500 метрів) або одномодовий кабель 9 micron (до 10км).

– Форм-фактор 1U для установки в шафу 19".

**QLogic SANBox 5802V – стекуємі продуктивні комутатори стандарту FC 8GBit**

– До 20 портів на шасі на швидкості 8 Gbps, назад сумісних з попередніми стандартами 4/2/1Gbps.

– Об'єднання в стек – до 120 портів на один стек комутаторів.

– Швидкість передачі – 1700 MBps через порти 8-Gbps, 5100 MBps через порти 20-Gbps.

– Агрегована пропускна здатність 544-Gbps на одне шасі, наскрізна архітектура, що не блокує.

– Інтерфейси – стандартні SFP модулі (Small Form Pluggable) з гарячою заміною на багатомодовий кабель 50/62.5 micron (до 500 метрів) або одномодовий кабель 9 micron (до 10км).

– Форм-фактор 1U для установки в шафу 19".



## 2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-

орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети). Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython. Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android. У міру старіння платформи її підтримка в основній гілці мови припиняється. Наприклад, із серії 2.6 припинена підтримка Windows 95, Windows 98 і Windows ME. Однак на цих платформах можна використовувати попередні версії Python – на даний момент співтовариство активно підтримує версії Python починаючи від 2.3 (для них виходять виправлення). При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть

бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх. Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий. Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

## 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 мереж SAN на основі стандарту Fibre Channel Gen 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

Після ефектного підйому в 90-х роках продуктивності настільних комп'ютерів, робочих станцій і пристроїв зовнішньої пам'яті (а також переходу до розподілених клієнт-серверним архітектурам) дуже багато уваги стало приділятися надійним і високопродуктивним засобам обміну даними між комп'ютерами й пристроями зовнішньої пам'яті. В Fibre Channel (дуже надійна технологія з'єднань із пропускною здатністю, обчислювальної гігабітами в секунду) для забезпечення конкурентного обміну інформацією між робочими станціями, універсальними електронно-обчислювальними машинами, серверами, системами зберігання даних і інших периферійних пристроїв застосовуються стандартні мережні протоколи й протоколи зв'язку із пристроями масової пам'яті. Архітектура Fibre Channel підтримує безліч топологій, агрегатна пропускна здатність яких може досягати порядку Тбіт/с. У число пропонованих сьогодні на ринку готових до використання (off-the-shelf) виробів з підтримкою Fibre Channel входять комутатори, концентратори, підсистеми зберігання інформації, пристрою пам'яті й адаптери.

На більшість робочих станцій покладають наступні завдання:

– обмін даними із пристроями масової пам'яті;

– обмін даними з іншими робочими станціями й серверами по локальній мережі.

Для виконання цих завдань робочі станції обладнані спеціальними окремими портами.

В 80-х роках кілька компаній виступило із пропозицією розробити стандарт обміну інформацією, за допомогою якого й системи пам'яті, і вузли локальної мережі могли б передавати дані по тим самим волоконним (fibre) лініях

(термін "волоконний" fibre використовується в цьому випадку як узагальнений термін для опису як оптоволоконого, так і мідного кабелю). В основу стандарту були покладені наступні вимоги:

– передача даних повинна здійснюватися зі швидкостями від 133 Мбіт/с до 4 Гбіт/с (залежно від обраної вартості);

– канали зв'язку повинні підтримувати передачу даних на відстані до 10 км;

– лінії зв'язку повинні мати компактні рознімання;

– смуга пропущення каналів зв'язку повинна залишатися високої незалежно від відстані передачі даних

– канали зв'язку повинні забезпечувати взаємне з'єднання більшої кількості пристроїв масової пам'яті, чим дозволяють існуючі багатоабонентскі канали для підключення пристроїв пам'яті;

– мережі зв'язку повинні будуватися на базі стандартних компонентів;

– мережі зв'язку повинні допускати підбор співвідношення "ціна/продуктивність" для застосування їх у самому широкому діапазоні систем від невеликих настільних або убудованих до суперкомп'ютерів;

– канали зв'язку повинні забезпечувати передачу команд всіх уже існуючих і широко використовуваних протоколів, включаючи IP (Internet Protocol), SCSI, IPI, HiPPI-FP і аудіо/відео.

Специфікація, що вийшла в результаті, Fibre Channel (ANSI X3.230-1994) це опис технології міжз’єднань, що може використовуватися і як канал зв'язку із системами зберігання даних, і як мережна технологія:

– її мережні характеристики задовольняють вимогам по підключенню, відстаням передачі й мультиплексуванню протоколів;

– вона підтримує всі звичайні функції каналів зв'язку із системами пам'яті, забезпечуючи простоту, надійність, повторювану продуктивність і гарантовану доставку.

Таким чином, Fibre Channel може служити і як мережа зберігання даних (storage area network SAN), і як високошвидкісна локальна мережа (local area network LAN).

**Мережі зберігання даних (SAN)**

Мережі зберігання даних SAN (Storage Area Network) використовуються для підключення одного або декількох серверів до однієї або більше системам масової пам'яті. Кожна система масової пам'яті може являти собою:

– дисковий масив RAID;

– систему резервного копіювання на магнітну стрічку;

– бібліотеку магнітних стрічок;

– бібліотеку компакт-дисків (CD-ROM);

– довільний набір дисків JBOD (Just a Bunch Of Disks).

У серверах і робочих станціях Fibre Channel може використовуватися для поділу доступу до тому самому пристрою масової пам'яті. Старі SCSI-системи підключаються до мереж SAN за допомогою мостів Fibre Channel-SCSI.

**Мережі**

Технологія Fibre Channel розроблявся як засіб підтримки інформаційних додатків корпоративного рівня, тому розроблювачі приділяли особливу увагу підвищенню її продуктивності в порівнянні із традиційними локальними мережами типу Ethernet. Серед інших достоїнств Fibre Channel для корпоративних мереж можна назвати наступні:

– протокольний стік, що забезпечує доставку з підтвердженням (confirmed delivery);

– можливість обходу протокольного стека з метою підвищення продуктивності;

– повна підтримка звичайних функцій автоматичного визначення мережних адрес, включаючи підтримку ARP, RARP і інших протоколів з автовизначенням;

– підтримка двоточечних каналів, кільцевих каналів з поділом смуги пропущення й каналів, що комутируються, з масштабованою пропускною здатністю;

– виділення реальних каналів (або віртуальних каналів із дробовою смугою пропущення), що забезпечують гарантовану пропускну здатність для виконання таких відповідальних операцій, як резервне копіювання й ін.;

– можливість виділення як фізичних, так і віртуальних каналів;

– дуже швидкий час установлення з'єднань (вимірюване в мікросекундах) за допомогою апаратних протоколів Fibre Channel;

– послуги із установленням (при малих затримках) і без установлення з'єднання;

– автоматичне конфігурування мережного вузла незалежно від топології Fibre Channel;

– повна підтримка синхронізуємих за часом додатків (типу передачі відео) за допомогою віртуальних каналів із дробовою пропускною здатністю;

– можливість вибору між масовою (high-bandwidth) і швидкісний (low-latency з малими затримками) передачами з використанням кадрів змінного розміру (0-2 Кбайт). Дуже малі затримки забезпечуються при передачі кадрів розміром менш 100 байт, а масові передачі пересиланням більших обсягів даних у кадрах максимального розміру.

**Фізичний і сигнальний рівні**

Fibre Channel являє собою архітектуру з п'ятьома незалежними рівнями, що визначають:

– фізичні лінії зв'язку й швидкості передачі;

– схему кодування;

– протокол кадрування й керування потоками;

– загальний сервіс;

– інтерфейси із протоколами верхнього рівня.

**Фізичний стандарт Fibre Channel**

Повністю перший варіант фізичного стандарту Fibre Channel (FC-PH) був опублікований у січні 1993 року, другий варіант з'явився в жовтні 1993, а статус затверджений стандарт одержав у листопаді 1994 року. У ньому визначаються наступні три нижніх рівні:

– FC-0 (фізичні інтерфейси й середовище передачі);

– FC-1 (протокол передачі):

– FC-2 (протокол кадрування й обміну сигналами)

**Фізичні інтерфейси й середовище передачі: FC-0**

Рівень FC-0 це самий нижній рівень, що визначає фізичне середовище передачі. Архітектура Fibre Channel може працювати з різними фізичними лініями й швидкостями передачі. Реалізований в Fibre Channel підхід забезпечує максимальну гнучкість, дозволяючи застосовувати вже існуючі кабельні системи (як і самі останні технології в цій області).

Наприклад, Fibre Channel допускає здійснювати уведення в будинок одномодовим оптоволоконим кабелем, усередині будинку виконувати розведення багатомодовим оптоволоконим кабелем, а індивідуальні робітники станції підключати мідними проводами.

На коротких відстанях у системах Fibre Channel застосовуються твінаксиальні мідні з'єднання. При лінійній швидкості передачі в 1,0625 Гбод ефективна пропускна здатність становить 100 Мбайт/с. Звичайно мідні кабелі оснащуються стандартними розніманнями DB-9 або недавно розробленими з'єднувачами HSSDC.

Також випробувані й схвалені оптоволоконі лінії (з ефективною швидкістю передачі 2,125 і 4,25 Гбод). Оптоволоконі кабелі, як правило, оснащуються здвоєними розніманнями типу SC.

Устаткування для роботи в середовищі Fibre Channel може мати інтерфейс для підключення оптоволоконого або мідного кабелю або забезпечуватися

універсальним (media independent) інтерфейсом. У стандарті визначаються три універсальних інтерфейси:

– GLM (Gigabaud Link Modules зв'язкові модулі для швидкостей передачі в трохи Гбод), до складу яких входять пристрій перетворення сигналів з паралельної форми в послідовну й назад (SERDES). Модулі GLM являють собою 20-розрядні інтерфейси з керуючими й схемами, що кодують, Fibre Channel і використовуються головним чином для конфігурування на підприємстві-виготовлювачі, хоча допускають також внесення змін на місці або модернізацію самими користувачами.

– GBIC (Gigabit Interface Converters конвертери для швидкостей передачі в декілька Гбіт) які забезпечували послідовний інтерфейс із пристроєм SERDES. Модулі GBIC можуть вставлятися й витягати із працюючих вузлів Fibre Channel. Ця функція досить корисна при обслуговуванні багатопортових пристроїв типу комутаторів і концентраторів, коли потрібна модифікація окремих портів без порушення роботи інших портів.

– MAI (Media Interface Adapter інтерфейсний адаптер лінії), використовуваний для підключення мідних кабелів з розніманнями DB-9 до багатомодовим оптоволоконим ліній. Напруга живлення оптоволоконих приймачепередатчиків подається в цьому випадку через окремі виводи рознімання DB-9.

**Протокол передачі: FC-1**

Відновлення синхросигналу, байтова синхронізація й кодування/декодування реалізовані в FC-1 за допомогою збалансованої по постійному струмі схеми кодування 8У/10У, що володіє чудовими характеристиками передачі. За цією схемою вісім біт даних (один байт) передаються як 10-розрядна група. Цей збалансований код (винахід IBM) дозволяє розробляти дешеві компоненти й забезпечує гарну швидкість передачі з більше простим відновленням синхросигналів. Поділ на байти й слова забезпечує спеціальний символ (називаний символом коми). У число інших достоїнств цього

коду входять можливість виявлення помилок і простота логічної реалізації як кодує, так і декодувального пристрою.

**Протокол кадрування й обміну сигналами: FC-2**

Надійність обміну даними цей наслідок застосування в Fibre Channel протоколу кадрування й сигналізації FC-2. Протокол FC-2 визначає механізм транспортування даних, що не залежить від протоколу верхнього рівня. Протокол FC-2 це самоконфігурующийся протокол, що підтримує з'єднання точка-точка, петлю з арбітражним доступом і комутируються среди, що.

Обмін даними по каналі Fibre Channel виконується між зв'язаними між собою портами. У кожному вузлі є спеціалізована мікросхема (ASIC) з убудованим пристроєм керування каналом Fibre Channel (Fibre Channel Link Control Facility), що здійснює як логічне, так і фізичне керування каналом і забезпечує логічний інтерфейс із іншою частиною системи.

Кожний порт має власний передавач і приймач і може виступати в ролі:

– ініціатора (originator);

– відповідача (responder);

– і ініціатора, і відповідача одночасно.

Кожному порту призначається унікальне ім'я, називане Ідентифікатором порту N_порт або NL_порт (N_Port або NL_Port). N_порт може бути портом такого вузла, як:

– сервер;

– робоча станція;

– периферійний пристрій.

Якщо N_порт включається в кільцеву топологію петлю (loop), то він стає NL_портом.

Порти обмінюються дуплексними потоками даних.

Крім структури кадрів, протокол FC-2 підтримує багатий набір функцій, у який входять:

– 32-розрядний CRC-код (циклічний контрольний код), що забезпечує виявлення помилок передачі й гарантує цілісність даних;

– спеціальні конструкції для підвищення ефективності мультиплексування операцій;

– механізм керування передачею потоків даних, що гарантує доставку інформації (для сервісу без організації з'єднання (connectionless) механізм керування міжбуферний і/або міжвузловий, для сервісу на основі з'єднань (connection-based) міжвузловий);

– набір базових функцій, використовуваних у різних протоколах верхнього рівня;

– убудовані протоколи для підтримки канальних операцій, керування конфігурацією Fibre Channel, відновлення помилок і одержання інформації про стан ліній зв'язку й портів;

– необов'язкові заголовки, які можуть використовуватися для маршрутизації даних у мережі;

– керуюча інформація в заголовку, що полегшує маршрутизацію на апаратному рівні;

– процеси сегментації й складання даних.

**Групування даних**

Для полегшення передачі даних (з верхніх рівнів) по мережі Fibre Channel у рівні FC-2 була визначена ієрархія груп даних, до складу якої входять:

– упорядковані набори (Ordered sets);

– кадри (Frames);

– пакети (Sequences);

– обміни (Exchanges).

**Упорядковані набори**

Упорядкований набір це група із чотирьох 10-розрядних інформаційних і керуючих символів. Останні призначені для виконання деяких канальних функцій нижнього рівня, таких як позначення границь кадрів і обмін сигналами

між обома кінцями каналу. Сигнальний обмін забезпечує ініціалізацію каналу після включення живлення й виконання основних дій по відновленню каналу після збою.

**Кадри**

Кадр це найменший по розмірах неподільний пакет даних, що пересилається по мережі. Структура кадрів показана на мал. 2. Всі адреси вказуються в заголовку кадру. Кадри не видні протоколам верхнього рівня й містять наступні поля:

– 4-байтовий обмежник "початок кадру";

– 24-байтовий заголовок кадру (визначається протоколом FC-2);

– необов'язкове 64-байтне поле (визначається протоколом FC-2);

– поле змінної довжини, що містить дані більше високих протокольних рівнів;

– 4-байтний CRC-код (для виявлення помилок передачі);

– 4-байтний обмежник "кінець кадру".

Підтвердження передачі кадру (або групи кадрів) входить у функції механізму керування потоком передачі. Для повідомлення про недоставляння кадрів служать повідомлення "Зайнята" (Busy) і "Відмова" (Reject).

**Пакети**

В архітектурі Fibre Channel не накладається ніяких обмежень на обсяг переданих між додатками даних. У звичайних локальних мережах прикладне ПЗ повинне знать про максимально припустимий для пересилання розмірі кадру (або пакета). В Fibre Channel розміри кадрів від прикладного ПЗ сховані.

Прикладний програміст для Fibre Channel працює з одиницями передачі даних, називаної пакетом (Sequence). Пакет сегментується й передається по лінії Fibre Channel у вигляді впорядкованого набору з одного або декількох кадрів.

Поділ пакета на кадри, розмір яких відповідає величині, заданої в процесі встановлення з'єднання між взаємодіючими портами або між портами й мережею Fibre Channel, входить у функції протоколу FC-2. Будь-який пакет, створений

програмним забезпеченням верхнього рівня, може бути розбитий на кілька кадрів. У заголовку кадру є поле Sequence Identifier, у яке записується унікальний ідентифікатор пакета. Усередині пакета кадри розрізняються полем Sequence Count, що зберігає порядковий номер кадру в пакеті.

Пакет також є мінімальною відновлюваною одиницею передачі. При виявленні помилок передачі протокол FC-2 визначає, при передачі якого пакета відбулася дана помилка й виконує повторну передачу цього пакета (і всіх наступних).

**Обміни**

Обмін це сукупність неконкуруючих пакетів (один або більше), що ставляться до однієї операції, що, приміром, може складатися з декількох передач, включаючи:

– команду зчитування певних даних;

– передачу даних;

– передачу статусу завершення операції.

Кожна передача (команда, дані, статус завершення) реалізується у вигляді окремого пакета. Однак всі разом вони формують обмін.

У кожний момент часу в рамках одного обміну може передаватися тільки один пакет, хоча в цілому по каналі зв'язку можуть передаватися кілька конкуруючих пакетів з різних обмінів. Це одна з форм мультиплексування, підтримувана в технології Fibre Channel. Кожний обмін унікальним образом ідентифікується кожним бере участь у її передачі N-портом.

Поле Originator Exchange ID (Ідентифікатор ініціатора посилки) заповнюється N-портом, що відправляє найперший пакет, а поле Responder Exchange ID (Ідентифікатор одержувача посилки) N-портом, що одержує найперший пакет (зміст поля залежить від типу реалізації). Ці ідентифікатори записуються в заголовки кадрів і використовуються N-портами для керування даним обміном.

**Протоколи верхніх рівнів**

У фізичному стандарті Fibre Channel визначений так званий транспортний сервіс (transport service), у завдання якого входить надійне й швидке пересилання даних і який має можливості масштабування, розрахованими практично на будь-які прикладні вимоги. Поверх рівнів фізичного стандарту визначені також наступні два протокольних рівні:

– Common Transport for Generic Services Єдина транспортна служба для Загального сервісу (FC-3);

– Upper Layer Protocol Mapping Відображення протоколів верхнього рівня (FC-4).

Ці рівні розширюють функціональні характеристики каналів Fibre Channel і забезпечують можливість взаємодії різних підсистем.

**Загальний сервіс**

Рівень FC-3 надає сервери, доступні всім N-портам. У цей час визначені наступні чотири сервери:

– сервер імен (Name Server);

– сервер альтернативних імен (Alias Server);

– служба керування (Management Services);

– сервер ключів (Key Server).

**Сервер імен**

Сервер імен підтримує каталог N_портів у межах однієї мережі Fiber Channel і базу даних з такими елементами, як ідентифікатори (ID), зовнішні імена й протоколи FC-4, підтримувані для кожного N_порту. Будь-який N_порт може в будь-який час звертатися до сервера імен для одержання списку інших підключених до мережі N_портів.

**Сервер альтернативних імен**

Сервер альтернативних імен використовується для присвоєння альтернативних імен багатоадресним групам (multicast groups) і групам захвата (hunt groups). В усі N_порти, яким було поставлено у відповідність альтернативне

ім'я багатоадресної групи, посилає груповий кадр (muiticast frame). Група захвата це сукупність всіх N_ портів, інтерфейси Fibre Channel яким об'єднані в одному вузлі. Цій безлічі N_ портів сервер альтернативних імен призначає спеціальний ідентифікатор, у результаті чого всі кадри з даним ідентифікатором будуть направлятися в будь-який вільний (незайнятий) N_ порт із цього набору. Подібний підхід дозволяє підвищити ефективність пересилання завдяки зниженню ймовірності того, що якийсь N_порт буде зайнятий.

**Служби керування**

Служби керування це засобу доступу до мережі Fibre Channel, її внутрішній топології й конфігураційним даним з боку керуючих додатків. Керуючі додатки можуть, наприклад, указувати, яким N_портам дозволено взаємодіяти один з одним. Інші служби дозволяють керуючим додаткам виявляти характер взаємозв'язків у мережі Fibre Channel. Для правильного конфігурування мережі Fibre Channel передбачена служба надання прав доступу.

**Сервер ключів**

Сервер ключів забезпечує стандартний метод розподілу ключів доступу й ключів шифрування. Звертання до сервера ключів, може знадобитися, наприклад, що управляє додатку для одержання необхідного ключа доступу, перед тим, як звернутися до служб керування.

**Рівень відображення протоколів верхнього рівня (FC-4)**

В Fibre Channel дані можуть одночасно передаватися по тому самому фізичному інтерфейсі із застосуванням різних протоколів. Як протоколи, відображуваних на рівні FC-4, були специфіковані (або запропоновані) наступні протоколи:

– SCSI (Small Computer System Interface);

– IP (Internet Protocol);

– HiPPI (High Performance Parallel Interface Framing Protocol);

– SBCCS (Single Byte Command Code Set Mapping) для реалізації ESCON і блокових мультиплексних інтерфейсів;

– Audio Video Fast File Transfer;

– протоколи реального часу для убудованих авіаційних електронних систем;

– VI (Virtual Interface).

У стандарті Fibre Channel допускається також використання фірмових протоколів.

Протоколи верхнього рівня відображаються за допомогою FC-4 на нижчі фізичний і сигнальний рівень FC-PH. Фізичний протокол і протокол обміну сигналами реалізовані на апаратному рівні, що забезпечує їхню високу продуктивність. Як правило, канальні протоколи будуються за принципом "команда/дані/статус", описаному вище. Кожна із цих інформаційних категорій має свої власні характеристики й вимагає окремої обробки. При цьому всі три категорії обробляються тим самим апаратно реалізованим протоколом. Апаратна реалізація Fibre Channel усуває необхідність у додатковій комп'ютерній обробці уведення/виводу, що істотно підвищує продуктивність серверів, робочих станцій і систем масової пам'яті.

**Топології мереж Fibre Channel**

Технологія Fibre Channel підтримує наступні три фізичні топології об'єднання вузлів:

– точка-точка ;

– петля;

– комутуючі топології.

**Петля**

Петлі, з'єднані з корпоративною мережею за допомогою комутатора, називаються петлями загального користування (public loops). Автономна петля називається приватної (private loop). Звичайно для об'єднання вузлів у петлю використовуються концентратори (hubs).

**Комутуючі топології**

Основне завдання комутаторів приймати кадри від вузла-джерела й пересилати їхньому вузлу-приймачу. У кожного вузла є унікальна адреса Fibre Channel, використовуваний при маршрутизації кадрів. У комутаторах від вузлів ховається наступна інформація:

– тип мережної топології;

– маршрут передачі кадрів по комутирується сети, Що, Fibre Channel;

– внутрішня структура мережі Fibre Channel, що може складатися з декількох з'єднаних один з одним комутаторів.

На частку кожного з вузлів залишається тільки підтримка з'єднання типу точка-точка між собою й відповідним комутатором. Передача вузлом інформації аналогічна набору телефонного номера в телефонній мережі: вузол просто вставляє ідентифікатор (ID) вузла-приймача в заголовок кадру, що передує переданим даним, і посилає цей кадр комутатору. Якщо зазначений ID неправильний, то комутатор відкидає даний кадр. Якщо з якої-небудь причини комутатор не може обробити кадр, він відповідає сигналом "зайняте" (Busy ), змушуючи вузол повторити спробу передачі.

На комутатор покладає відповідальність за маршрутизацію кадрів залежно від класу обслуговування, зазначеного вузлом-ініціатором. Після оцінки відстані між вузлами-кореспондентами він вибирає підходящій обраній якості обслуговування середовище й мережні компоненти.

**Петлі й концентратори**

Петля Fibre Channel може поєднувати до 127 портів. Гнучкість петлі забезпечується за рахунок застосування портових шунтів (port bypass). Ці шунтувальні схеми встановлюються або на об'єднавчій панелі дисководів, або в зовнішньому пристрої, називаному концентратором (hub).

Портові шунти вміють автоматично визначати наявність активного вузла й включати його в петлю. Якщо вузол відключений або несправний, вони так само автоматично виключають його з петлі. Всі ці операції виконуються без якого-небудь втручання оператора.

**Класи обслуговування**

Комутатори й вузли Fibre Channel можуть підтримувати п'ять різних класів обслуговування:

– Клас 1 Acknowledged Connection Service (виділені з'єднання з підтвердженням);

– Клас 2 Acknowledged Connectionless Service (передачі без організації з'єднання з підтвердженням);

– Клас 3 Unacknowledged Connectionless Service (передачі без організації з'єднання й без підтвердження);

– Клас 4 Fractional Bandwidth Connection-oriented Service (з'єднання із дробовою смугою пропущення;

– Клас 6 Unidirectional Connection Service (односпрямовані з'єднання);

**Клас 1: Виділені з'єднання з підтвердженням**

Клас 1 це справжні виділені з'єднання (за принципом комутації каналів), що надають повну смугу пропущення. Такі з'єднання в Fibre Channel установлюються (і розриваються) за кілька мікросекунд.

З'єднання класу 1 це наскрізний шлях між взаємодіючими вузлами через комутатор. Оскільки єдиними накладними витратами є час установлення й розриву з'єднання, те цей сервіс особливо ефективний при передачі великого обсягу інформації.

З'єднання класу 1 забезпечують гарантовану доставку даних за допомогою підтвердження прийому. У деяких завданнях (типу резервного копіювання й відновлення бази даних) гарантована доставка застосовується для надійного й швидкісного пересилання інформації без накладних витрат, властивих протокольному мережному стеку.

Однієї їхній особливостей класу 1 є можливість Camp on, що дозволяє комутатору відслідковувати зайнятий порт і ставити його в чергу на встановлення наступного з'єднання. Як тільки даний порт звільняється, комутатор відразу ж установлює з ним з'єднання. Тим самим скорочуються

строки встановлення з'єднання, оскільки в цьому випадку не потрібно передавати в ініціюючу передачу N_порт сигнал Зайнятий, що змушує його повторювати спробу встановлення з'єднання.

Ще одна особливість класу 1 пакетований зв'язок (Stacked Connect), що дозволяє N_ порту-ініціаторові запитувати встановлення декількох з'єднань через комутатор. Ця можливість також скорочує накладні витрати на встановлення з'єднання й підвищує ефективність роботи комутатора.

Крім того, у класі 1 передбачена функція буферизації (buffered service), що дозволяє двом різношвидкісним портам обмінюватися даними по лінії Fibre Channel.

Серед варіантів з'єднань класу 1 є можливість Dedicated Simplex Service виділена односпрямована передача. Як правило, з'єднання класу 1 двонаправлені, однак у деяких випадках передача даних ведеться тільки в одному напрямку. У цьому випадку приймач і передавач вузла розв'язані між собою, внаслідок чого вузол може передавати дані другому вузлу й одночасно приймати їх від третього.

**Клас 2: Передачі без установлення з'єднання з підтвердженням**

Сервіс класу 2 являє собою незалежну маршрутизацію кожного кадру даних, що гарантує доставку шляхом підтвердження прийому. При цьому канал між двома взаємодіючими вузлами не виділяється. Комутатор мультиплексує потоки даних між N_портами й NL_портами без виділення каналу через комутатор.

Механізм керування потоками класу 2 усуває проблеми перевантаження, властиві багатьом мережам без комутації каналів, шляхом посилки N_ порту-ініціаторові сигналу зайнятості у випадку, якщо порт-приймач не в змозі приймати повідомлення. При одержанні цього сигналу N_ порт повторює спробу відправлення повідомлення.

Завдяки надзвичайно малим затримкам передачі кадрів, з'єднання класу 2 є ідеальним засобом пересилання різної бізнесу-інформації.

**Клас 3: Передачі без установлення з'єднання й без підтвердження**

Клас 3 аналогічний класу 2 за винятком того, що відправникові підтвердження про прийом не посилають.

Режим передачі без підтвердження прийому використовується в системах масової пам'яті, підключених до петель Fibre Channel. У петлі встановлюється логічне з'єднання точка-точка , по якому дані надійно пересилаються між накопичувачами.

Передачі класу 3 можуть також використовуватися протоколами, не пов'язаними із системами масової пам'яті, наприклад, IP або VI. Ці протоколи створюють власні логічні з'єднання точка-точка й підтверджують прийом власними засобами, опираючись лише на засоби надійної доставки даних Fibre Channel.

**Клас 4: З'єднання із дробовою смугою пропущення**

Клас 4 це орієнтований на з'єднання сервіс із дробовою смугою пропущення. Між кореспондентами встановлюється віртуальне з'єднання зі смугою пропущення, достатньої для надання послуг з передбачуваною якістю. Один вузол може встановлювати до 256 одночасних з'єднань класу 4.

З'єднання класу 4 це двонаправлене з'єднання, по одному віртуальному каналі в кожному напрямку, для кожного з яких може підтримуватися різний набір параметрів якості обслуговування. У число цих параметрів входять гарантована смуга пропущення й максимальна затримка передачі від одного кінця до іншого. Застережене при встановленні кожного віртуального каналу якість обслуговування підтримується спеціальною функцією комутатора (facilitator).

Передача кадрів по з'єднаннях класу 4 синхронізується комутатором, для того щоб не перевищити виділену для віртуального каналу смугу пропущення. Комутатор мультиплексує кадри різних віртуальних каналів між тими самими або різними парами вузлів. Сервіс класу 4 забезпечує впорядковану доставку кадрів.

Керування потоками класу 4 є наскрізним, з гарантованою доставкою. Сервіс класу 4 найкраще підходить для критичних за часом доставки додатків (типу передачі відеосигналів).

**Клас 6: Односпрямовані з'єднання із груповою передачею й пріоритетами**

Сервіс класу 6 аналогічний сервісу класу 1 за винятком того, що є односпрямованим. Додатково він забезпечує надійні групові (multicast) доставки й передачі по пріоритетах. Сервіс класу 6 ідеальний для широкомовних відеододатків і систем реального часу, пересилаючих дані великого обсягу.

**Топології Fibre Channel**

У технології Fibre Channel передбачене використання кожної з наступних топологій:

– топологія точка-точка з виділеною смугою пропущення;

– петля з поділом смуги пропущення;

– топологія, що комутирується, з масштабованою смугою пропущення.

**Топологія точка-точка з виділеною смугою пропущення**

Топологію точка-точка можна реалізувати двома способами. У першому випадку це просто два N_порти, з'єднані один з одним і які обмінюються інформацією. В іншому способі для організації з'єднань між парами вузлів з деякої їхньої безлічі використовується зовнішній комутатор ліній зв'язку.

У кожному разі обоє N_порту користуються всією смугою пропущення каналу. Обмін даними дуплексний. Таким чином, 1-гігабітна приймально-передавальна лінія надає виділену смугу пропущення в 2 Гбіт/с.

**Петля з поділом смуги пропущення**

Петля Fibre Channel це недороге рішення, що дозволяє декільком вузлам одночасно використовувати 1-гігабітну смугу пропущення. Вузли розділяють доступ до петлі, однак будь-якій парі вузлів після встановлення логічного двоточечного з'єднання доступна вся смуга пропущення в 1 Гбіт. У петлю може входити до 127 вузлів, але тільки один з них може бути портом комутатора.

Звичайно петлі використовуються в мережах і призначені для заміни SCSI у дисководах наступного покоління.

Вузол петлі Fibre Channel запитує керування петлею шляхом видачі сигналу, називаного примітивом (primitive). Кожний вузол петлі (ще що не встановив з'єднання з яким-небудь іншим вузлом петлі) передає цей примітив (як і всі кадри даних) сусідньому вузлу. Якщо примітив вертається з адресою вузла-відправника, то цей вузол (який може бути й портом комутатора) одержує керування петлею.

Якщо право керування петлею одночасно запитують два (або більше) вузли (або вузол і порт комутатора), то керування передається вузлу з меншою адресою. Після того, як вузол або порт комутатора одержить керування, він може відкрити дуплексний двоточечний канал зв'язку з будь-яким іншим вузлом.

У кожний момент часу по петлі може обмінюватися даними тільки одна пара вузлів. Обраний може бути будь-який клас обслуговування. Однак у більшості випадків додатки користуються класом 3. При поверненні керування виникає чергова арбітражна ситуація. Всім портам гарантований рівноправний доступ до петлі.

Петля Fibre Channel є автоконфігуруємий і може працювати як з комутатором, так і без нього. Кожний вузол петлі (включаючи порт комутатора) може автоматично визначати конфігурацію петлі й коректно взаємодіяти з іншими вузлами петлі Fibre Channel без втручання оператора.

Автономна петля, не підключена до жодного комутатору, називається приватною петлею (private loop). Петля, підключена до комутатора Fibre Channel (через порт, називаний FL_Port), називається загальною петлею (public loop).

**Концентратори**

Петлі можуть складатися з вузлів, безпосередньо зв'язаних один з одним у кільце. Однак при такому підході відмова будь-якого вузла (і навіть простої його невключення) приведе до відмови всієї петлі. Ця проблема вирішується за допомогою концентратора (hub), у якому для визначення працездатності вузлів

застосовуються спеціальні схеми обходу (або шунтування) портів (port-bypass circuits). Це пристрій або розмикає канал (тим самим вставляючи відповідний вузол у петлю), або замикає (крім вузла з петлі й гарантуючи працездатність петлі навіть у випадку повної несправності вузла). Крім того, концентратори дозволяють вставляти й видаляти вузли з контуру в процесі роботи.

У пристроях масової пам'яті на базі Fibre Channel уже є петля, реалізована на його панель^-панелі-кросі-панелі. Кожний вузол крос-панелі теж оснащений схемою обходу, аналогічної застосовуваної в концентраторах. У такий спосіб забезпечується "гаряча" заміна дисководів, підключених до петлі Fibre Channel.

**Топологія, що комутирується, з масштабованою смугою пропущення**

Комутатори Fibre Channel дуже легко встановлювати й використовувати, оскільки протокол Fibre Channel забезпечує самоконфігуруємість і самоконтроль. При включенні вузла або комутатора Fibre Channel він спочатку визначає, що перебуває на іншому кінці сполучного кабелю, і потім вирішує, як із цим взаємодіяти.

Якщо з'єднані два вузли, то вони автоматично починають працювати в режимі точка-точка . Якщо вузол підключений до комутатора, то він спочатку реєструється в комутаторі й потім обмінюється конфігураційною інформацією. Якщо кабель з'єднує два комутатори, то вони в процесі обміну інформацією визначають тип конфігурації мережі й мережних адрес. Всі ці дії виконуються автоматично, без якого-небудь втручання з боку оператора.

У комутаторах є два наступні типи портів:

– F_порт/FL_порт (F_Port/FL_Port), за допомогою яких комутатор з'єднується з вузлами, і

– E_порт (E_Port), використовувані для зв'язку з іншими комутаторами.

Якщо порт комутатора може виступати й у ролі F_ порту, і в ролі E_ порту, то він називається G_портом (G_Port) (універсальним портом). Багато комутаторів оснащені портами саме цього типу, що автоматично визначають у процесі ініціалізації, чи бути їм портами типу F_Port або E_Port.

Комутатори Fibre Channel, які можуть одночасно підтримувати комутуючі як, так і не комутируються канали, що, фактично являють собою два окремих пристрої в одному корпусі й дозволяють щонайкраще використовувати обидві технології комутації. Одночасна підтримка комутації каналів і комутації кадрів необов'язкова. Деякі типи комутаторів призначені тільки для комутації каналів, інші ж тільки для комутації кадрів.

У режимі комутації кадрів смуга пропущення виділяється динамічно залежно від типу ліній. Маршрутизація кадрів між портами комутатора здійснюється на базі алгоритму адаптивної комутації. Для керування потоками між комутатором і підключеними портами N_Port, NL_Port або E_Port на рівні ліній зв'язку комутатор повинен уміти буферізувати кадри даних. Комутація кадрів в основному використовується в тих додатках, де необхідні короткі передачі з малими затримками.

У майбутньому E_ порти можна буде підключати й до глобальних мереж за допомогою пристроїв Inter Working Units (спільна розробка японських і американських компаній).

**Ініціалізація**

У процесі ініціалізації комутатори автоматично призначають адреси й здійснюють пошук каналів зв'язку один з одним. Вузли можна додавати й витягати із системи в процесі її експлуатації в будь-який момент. При додаванні нових вузлів останні самостійно реєструються в комутаторі й обмінюються з ним конфігураційними параметрами. У процесі реєстрації N_ порту привласнюється адреса відповідного порту комутатора. Адреси NL_портів складаються зі старших 16 розрядів адреси FL_порту комутатора, при цьому молодші 8 розрядів дозволяють адресувати до 126 вхідних у петлю портів.

**Сервери Fibre Channel**

Набір функцій, називаних Загальним Сервісом (Generic Services), які розширюють можливості основних протоколів Fibre Channel, надаються

серверами Fibre Channel. Ці сервери, як правило, застосовуються в що комутируються топологіях.

З'єднані один з одним комутатори утворять зв'язну архітектуру (fabric) ґрати . Серверні функції можуть концентруватися в одному комутаторі ґрат або розподілятися по всіх комутаторах. Протоколом, що надає всі види послуг, є єдиний транспортний протокол Fibre Channel (FC-CT), прозорий для всіх типів і топологій ґрати .

**Сервер реєстрації**

Сервер реєстрації звичайно являє собою якусь логічну структуру усередині комутатора, відповідальну за прийом і обробку запитів реєстрації. У завдання сервера реєстрації входить також підтвердження (або призначення) адреси N_порту для кожного вузла, що послав запит на реєстрацію. Після цього він може надати серверу імен всі реєстраційні параметри кожного з N_портів.

**Контролер комутатора або ґрати**

Контролер комутатора або решітк виконує такі службові функції, як ініціалізація, конфігурування, маршрутизація, і додатково може включати сервіс, що сприяє більше ефективній роботі ґрат .

**Сервер імен**

Сервер імен забезпечує пошук вузлів ґрати й визначення їхніх параметрів і ідентифікаторів N_Port. Ім'я вузла це 8-байтова величина, що може бути як IEEE-адресою (унікальним на глобальному рівні), так і локальним ім'ям. У кожному разі воно є унікальним в адресному просторі конкретного комутатора.

Ідентифікатори N_Port призначаються в процесі реєстрації (наприклад, але не обов'язково, за допомогою сервера реєстрації). Після цього в будь-якому вузлі або N_ порту можна організувати табличну структуру зв'язків між собою й іншими N_портами. Передбачене також перетворення імен для IP-адрес, протоколів верхнього рівня й класів обслуговування.

**Сервер альтернативних імен**

Якщо один сервер має кілька зв'язків із ґратами Fibre Channel, то всі вони можуть позначатися як якась група захвата (hunt group), що сервер альтернативних імен може привласнити групове альтернативне ім'я, або псевдонім (alias). Псевдоніми можуть також привласнюватися й багатоадресним (multicast) групам, що представляють собою групу N_ портів, кожний з яких приймає кадри, передані будь-яким іншої N-портом цієї групи. Кадр, відправлений N_ портом-джерелом, розмножується й передається всім іншим членам даної багатоадресної групи за допомогою сервісу класу 3. Сервер альтернативних імен виконує реєстрацію й скасування всіх псевдонімів. Один N_порт може одночасно зареєструватися в декількох багатоадресних групах. Він може також зареєструвати список (list) N_ портів як члени багатоадресної групи. За маршрутизацію кадрів з обліком багатоадресних груп і псевдонімів відповідальність несуть ґрати Fibre Channel. Fibre Channel цей надійний, високопродуктивний, зручний і недорогий комунікаційний засіб, призначений для систем інтенсивної обробки й обміну даними нового покоління. Вона дозволяє підняти продуктивність мереж масової пам'яті й серверів на якісно новий рівень. Високошвидкісні лінії зв'язку Fibre Channel коштують дешевше, ніж багато сьогоднішніх систем. На базі Fibre Channel можна будувати гетерогенні кластери накопичувачів, серверів і робочих станцій. Ця технологія має характеристики як комунікаційного каналу, так і мережі, що дає більше надійному, швидке, просте, ефективне й менш дороге рішення для створення інформаційних систем. Великі сховища й центри обробки інформації дозволяють краще управляти даними й здійснювати їхню доставку, що в результаті підвищує якість прийняття рішень. Масштабований комп'ютер і кластери масової пам'яті з можливостями швидкого доступу дозволять значно підвищити ефективність використання корпоративних ресурсів. Технологія Fibre Channel цей засіб не відставати від бурхливого розвитку систем масової пам'яті й використовувати їх з максимальною ефективністю.

### 3.2 Розробка структурної схеми

Проект стандарту шостого покоління FC Gen 6 на 32 Гбіт/с схвалений FCIA рік назад і вже не міняється, тобто досяг стабільного стану. Крім того, торік завершена робота над стандартом фізичного інтерфейсу нового покоління FC – FC-PI-6. Ратифікація FCIA фізичного інтерфейсу FC-PI-6 (Gen 6 Fibre Channel Physical Interface) стане важливою віхою на шляху до практичного впровадження FC 32 Гбіт/с і розробці відповідних комерційних продуктів. ANSI опублікувало стандарт FC Gen 6, після чого почалося розгортання рішень Fibre Channel нового покоління для побудови більше масштабованих, надійних і високошвидкісних мереж зберігання даних.

На підході стандарт Parallel FC 128 Гбіт/с, у якому пропускна здатність збільшена з 6400 до 25 600 Мбайт/с. Робота над ним повинна завершитися через рік після публікації FC Gen 6. Тоді ж на ринок будуть випущені відповідні продукти. Важливо, що при цьому зберігається повна зворотна сумісність із існуючим устаткуванням Fibre Channel.

В Parallel FC на 128 Гбіт/с використовуються чотири оптичних волокна для передачі даних у прямому напрямку й чотири – у зворотному. Які кабелі й з'єднувачі будуть потрібні для FC на 128 Гбіт/с? Як порти можуть використовуватися модулі QSFP28, CFP2, CFP4 або якісь майбутні чотирьохканальні інтерфейси, а для підключення встаткування – 12-волоконні кабелі із з'єднувачами MPO, активні волоконно-оптичні кабелі (AOC) довжиною до 50 м або кабелі прямого підключення (DAC) до 5 м.

Поряд з підтримкою більше високої швидкості передачі даних (32 Гбіт/с) стандарт FC Gen 6 має наступні особливості:

– Корекція помилок Forward Error Correction (FEC) підвищує надійність з'єднань завдяки автоматичному виявленню й усуненню бітових помилок, які можуть приводити до деградації продуктивності й збоям.

– Убудовані засоби протоколу підвищують енергоефективність (Energy Efficient Fibre Channel), оскільки при неактивному з'єднанні оптичні трансивери переходять у пасивний режим (standby), причому щосекунди це може відбуватися багаторазово. У результаті, по оцінках SNIA, оптичний канал споживає на 40-60% менше електроенергії.

– Підтримка N-Port ID Virtualization (NPIV) спрощує розгортання серверної віртуалізації й масштабування фабрик SAN за рахунок усунення обмежень по числу комутаторів FC і доменів. Завдяки NPIV спрощуються розгортання й міграція віртуальних серверів, тому що керування глобальними іменами WWN (Worldwide Name) переноситься з рівня гіпервізора на рівень протоколу Fibre Channel.

– FC-SP-2 підвищує рівень захищеності Fibre Channel у відповідності зі специфікаціями National Institute of Standards and Technology (NIST) Special Publication 800-131A. У ньому застосовується протокол автентифікації й керування ключами DH-CHAP.

– Підтримка встаткуванням Inter-Switch Link (ISL) Trunking дозволяє агрегувати чотири канали Fibre Channel по 32 Гбіт/с в один канал 128 Гбіт/с між серверами й СЗД. Такими каналами можуть бути чотири мідножильних кабелі або чотири довжини хвилі у волоконно-оптичному з'єднанні. Сьогодні це самий високошвидкісний спосіб обміну даними між серверами й системами зберігання.

– Забезпечується зворотна сумісність із устаткуванням FC на 8 і 16 Гбіт/с (Gen 4 і 5).

У середовищах з високим ступенем віртуалізації генерується значний трафік вводу-виводу. З переносом у віртуальне середовище баз даних Oracle, Microsoft SQL Server, SAP і інших відповідальних додатків детермінованість і надійність Fibre Channel стають незамінними якостями для високопродуктивних мереж зберігання, які повинні забезпечувати цілісність даних і їх стабільну, без втрат, передачу відповідно до SLA. Як очікується, нове покоління FC дозволить створювати більше продуктивні, надійні й масштабовані середовища зберігання

даних, що відповідають вимогам нових архітектур ЦОД, масштабної віртуалізації й флеш-систем з рекордними показниками IOPS.

Fibre Channel дозволяє створювати гнучкі з погляду архітектури рішення. Плани по розвитку цієї технології вселяють в інвесторів упевненість у тім, що їхні вкладення надійно захищені. Високошвидкісні SAN здатні задовольнити зростаючі вимоги серверів і систем зберігання до продуктивності мереж. Іноді вендори вбудовують флеш-пам'ять безпосередньо в сервери, але це дороге й погано масштабоване рішення. FC Gen 6 дає можливість застосовувати традиційну архітектуру із загальним сховищем і ефективніше використовувати ресурси.

Як правильно побудувати мережа зберігання даних поза залежністю від того, яке покоління FC у ній використовується? В Brocade уважають, що SAN повинна відповідати наступним принципам:

– Резервування на фізичному рівні. Здвоєна «фабрика» забезпечує високу доступність SAN. Сервери й СЗД підключаються до двох комутаторів з підтримкою багатоканальної передачі (multipathing). Комутатори не з'єднуються між собою, що забезпечує ізоляцію відмов.

– Правильна топологія. Топологія «кожний з кожним» (Full Mesh) у випадку невеликих фабрик (4–5 комутаторів) дозволяє одержати максимальну продуктивність і доступність. У великих SAN для масштабованості використовується топологія « ядро-границя» (Core-Edge).

– Агрегування з'єднань (Trunking) забезпечує усунення вузьких місць у фабриці й балансування навантаження на апаратному рівні, що дозволяє уникнути багатьох проблем.

– Використання функцій Access Gateway. Цей спрощений режим роботи комутатора FC призначений для підключення великої кількості серверів з невисоким навантаженням. Він підвищує масштабованість і надійність SAN, полегшує керування й часто використовується в убудованих комутаторах FC у шасі модульних серверів. У режимі шлюзу комутатор SAN приймає дані від хост-систем і передає їх на вищестоящі комутатори, що реалізують високорівневі

функції. Тим самим зменшується кількість доменів у фабриці (що позитивно позначається на швидкості її роботи й стабільності), а також число адмініструємих пристроїв. Крім того знімаються проблеми сумісності встаткування різних вендорів.

– Маршрутизація FC підвищує відказостійкість і полегшує керування й нарощування SAN. У територіально розподілених SAN рекомендується не «розтягувати» мережі зберігання між площадками, а використовувати маршрутизацію FC і задіяти на кожній з них пари ізольованих фабрик. У цьому випадку проблеми на одній площадці не будуть поширюватися на іншу.

Для комунікацій між площадками можна застосовувати безпосередньо протокол FC – «далекобійні» трансивери або рішення WDM. У цьому випадку буде підтримуватися синхронна реплікація даних між СЗД. При більших відстанях (до тисяч кілометрів) підійде протокол FCIP (фрейми FC інкапсулюються в IP). При цьому затримка Round Trip не повинна перевищувати 200 мс, а втрати пакетів можуть становити не більше 1%. Для таких комунікацій випускається спеціальне устаткування: маршрутизатор FCIP (FC over IP), наприклад Brocade 7840 або 7800, модулі для DCX, Cisco MDS 9000 SAN Extension over IP Package. За допомогою FCIP (10 Гбіт/с) реалізуються асинхронна реплікація й резервне копіювання даних.

По даним Brocade, отриманим у ході тестування, при відстані більше 75 км краще використовувати не FC, а FCIP. Цей протокол дозволяє застосовувати й більше ефективні алгоритми стиску даних. FC на 10 Гбіт/с (версія FC Gen 5) забезпечує комунікації між площадками через WDM і дозволяє використовувати 10-гігабітні трансивери. Конвергентна інфраструктура, що поєднує SAN і LAN, стала ЦОД стандартом, чому буде сприяти серверна віртуалізація. Конвергентні середовища дозволяють заощадити гроші, спрощують керування, знижують потреби в охолодженні й електроенергії, підвищують гнучкість і при цьому забезпечують низькі затримки, детермінованість і можливості керування мережами, властиві FC.

Зі збільшенням швидкості Ethernet і вдосконалюванням протоколів для передачі даних СЗД, включаючи Data Center Bridging (DCB) і Converged Enhanced Ethernet (CEE), удалося зменшити затримки й знизити втрати пакетів при піковому трафіку. Ці вдосконалення в LAN дозволяють сполучати в одній мережі трафік додатків і трафік СЗД. Технологія FCoE забезпечує передачу даних FC по мережі Ethernet на 10 Гбіт/с і взаємодія з існуючими мережами FC SAN. У числі переваг – уніфікація мережної інфраструктури, більше просте адміністрування й зниження витрат.

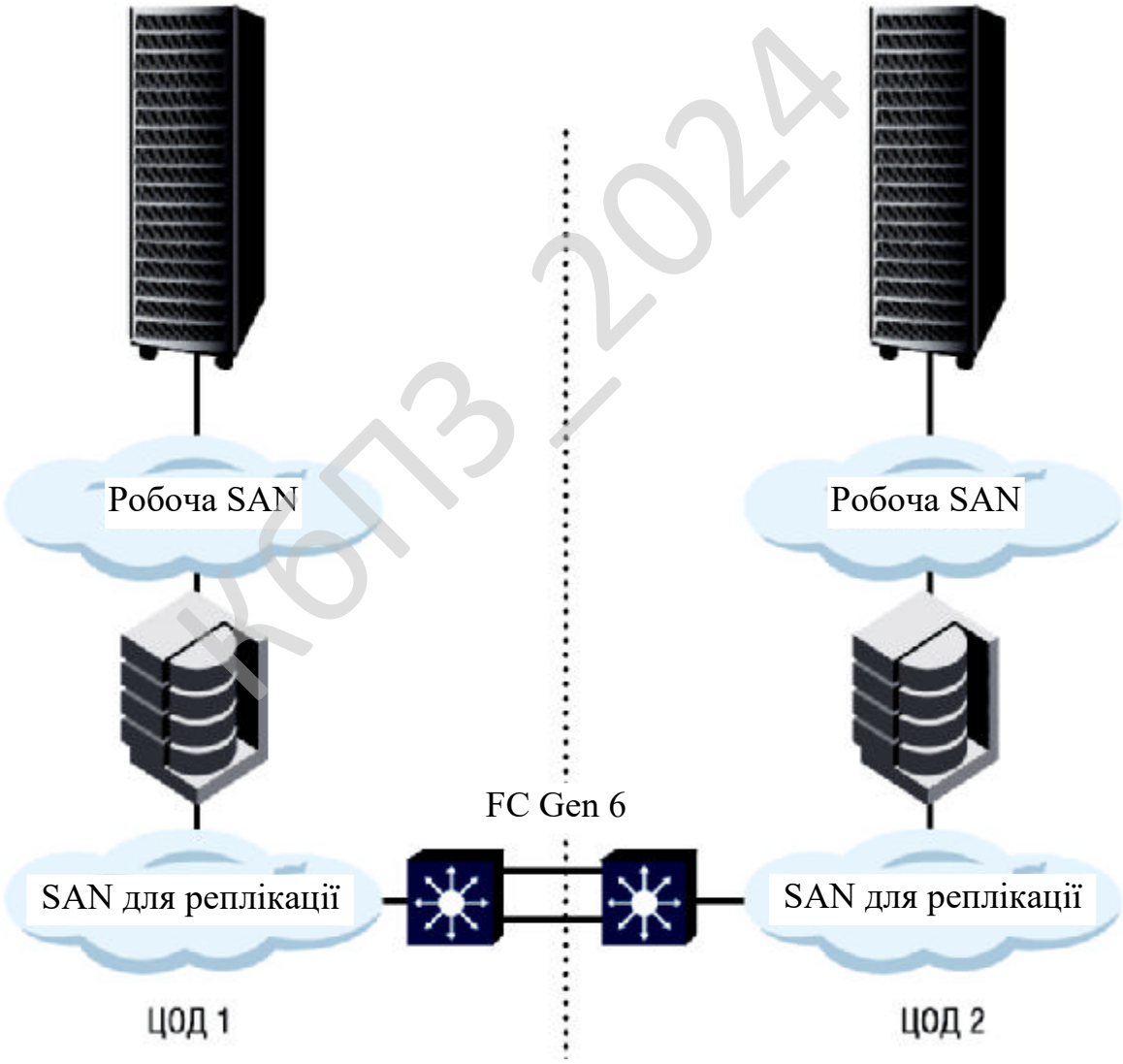




Рисунок 3.1 – Структурна схема системи

Fibre Channel over Ethernet (FCoE) забезпечує передачу трафіку Fibre Channel по локальній мережі Ethernet. Інкапсуляція фреймів FC у кадри Ethernet дозволяє сполучати блоковий трафік систем зберігання з іншим трафіком LAN. У результаті спрощується мережна інфраструктура ЦОД і знижуються витрати, оскільки не потрібно обслуговувати кілька мереж. Однак, як уважають деякі фахівці, незалежно від того, яка технологія використовується, фабрика повинна працювати у виділеній мережі зберігання й ні про яку конвергенцію з мережею передачі даних не може бути мови – ризики переважують виграш у вартості рішення.

Інші експерти впевнені, що віртуальні сервери, що працюють як з ресурсами зберігання, так і з обчислювальними ресурсами, приведуть до відмови від традиційних мереж зберігання. Адаптери HBA в SAN будуть замінятися на мережні адаптери HCA (Host Channel Adapter) або CNA, що підтримують кілька протоколів в одному каналі. Фактично це дозволяє агрегувати канали передачі даних і різко підняти продуктивність.

Історично Ethernet випереджає Fibre Channel по засобах діагностики й аналізу проблем, однак продукти FC у виконанні Brocade пропонують ряд діагностичних інструментів і інших удосконалень, що усувають цей розрив. Деякі з них стали стандартними в поточній версії Fibre Channel, інші будуть доступні в наступній.

Що вибрати – FCoE або Fibre Channel? Однозначної відповіді немає. FC – перевірена технологія, але якщо критично вартість або Ethernet переважніше з погляду стандартизації, то FCoE заслуговує розгляду. Ще один варіант – використання обох технологій: FC для критично важливих даних і додатків, а Ethernet для інших. У великих організаціях рекомендується дотримуватися Fibre Channel. Поряд з вибором протоколу важлива архітектура – комутуюча фабрика забезпечує довільне з'єднання портів, низькі затримки, відказостійкість і балансування навантаження. Fibre Channel уже давно має такі характеристики, тепер їх мають і Ethernet-фабрики.

Для середовища Ethernet ключовим є відмова від Spanning Tree на користь протоколу TRILL або Shortest Path Bridging, інакше властивих комутуючій фабриці характеристик одержати не вдасться. Фабрика передбачає також взаємодію між ЦОД і може бути розподіленої – охоплювати кілька площадок. Трафік FCoE між серверами й СЗД проходить через комутатори, які повинні підтримувати DCB (як, наприклад, Brocade VDX 6730, Dell MXL 10/40Gb, Cisco Nexus 5020, HP StorageWorks 2408 FCoE Converged Network Switch) і мати порти 10Gb.

Якщо на сервері встановлені звичайні мережні карти (NIC), то інкапсуляція FC в Ethernet здійснюється програмно. При інтенсивному трафіку це негативно позначається на продуктивності NIC. Більше високу швидкодію забезпечать конвергентні адаптери CNA з убудованою підтримкою інкапсуляції FCoE, наприклад QLogic 8200 і 8300, Cisco UCS M71 KR-Q, HP CN1000Q Dual Port Converged Network Adapter, Dell 1020 і ін.

Деякі нові дискові масиви мають порти FCoE. У їхньому числі – NetApp FAS3170, EMC CX 4-960 і VNX-F, Dell Compellent SC8000. Для відповідного підключення до серверів успадкованих дискових масивів використовують комутатори з підтримкою FCoE. У цьому випадку з комутаторами СЗД з'єднуються по FC. Ще один важливий момент – тип кабелів, що прокладаються між серверами, комутаторами й дисковими масивами. Для трафіку FCoE (10 Гбіт/с) звичайних кабелів Категорії 6 недостатньо. Для нормальної роботи буде потрібно більше висока категорія або волоконно-оптичні з'єднання. Для підключення серверів до комутаторів То іноді застосовують твінаксиальні кабелі.

Впроваджувати FCoE можна поетапно, у міру відновлення встаткування. Звичайно така міграція починається із заміни серверних мережних адаптерів на CNA. Адаптер CNA одночасно виконує функції FC HBA і Ethernet NIC. Комутатори Ethernet заміняють на продукти з підтримкою DCB або FCoE. Це дозволяє успадкованим серверам працювати як з локальною мережею, так і з директором Fibre Channel, а новим серверам, що підтримують FCoE, – з FC SAN.

У цей час три технології – FC, NAS і iSCSI – використовуються кожна у своїй властиво ніші. Для рішень NAS і iSCSI добре підходить технологія Ethernet Fabric. Хоча основне призначення комутаторів Ethernet Fabric – передача трафіку Ethernet, деякі моделі підтримують і трафік FCoE.

### 3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Представимо опис розробленого програмного комплексу для дослідження мереж зберігання даних SAN на основі стандарту FC Gen 6 із неоднорідним потоком повідомлень на основі аналітичного й імітаційного моделювання.

**Завдання проектування мережі** складається у визначенні пропускних здатностей КЗ із урахуванням неоднорідності трафіку, різноманіття топологій, алгоритмів маршрутизації, варіантів розміщення прикладних програм і наборів даних по вузлах мережі, способів взаємодії користувачів мережі.

**Аналітична модель мереж зберігання даних SAN на основі стандарту FC Gen 6** будується у вигляді розімкнутої СеМО для будь-якої топології мережі, що задається аналітично або графічно. На основі аналітичної моделі вирішується завдання визначення пропускних здатностей КЗ у розподілених мереж зберігання даних SAN на основі стандарту FC Gen 6 при обмеженнях на час доставки пакетів або на вартість мережі.

Програма дозволяє розрахувати характеристики мереж зберігання даних SAN на основі стандарту FC Gen 6, такі як:

– пропускні здатності КЗ;

– час затримки пакетів при передачі по кожному каналу й у мережі в цілому;

– завантаження кожного КЗ;

– інтенсивності потоків пакетів у каналах зв'язку;

– імовірності передачі пакетів від користувачів у мережу, між каналами й з мережі до користувачів.

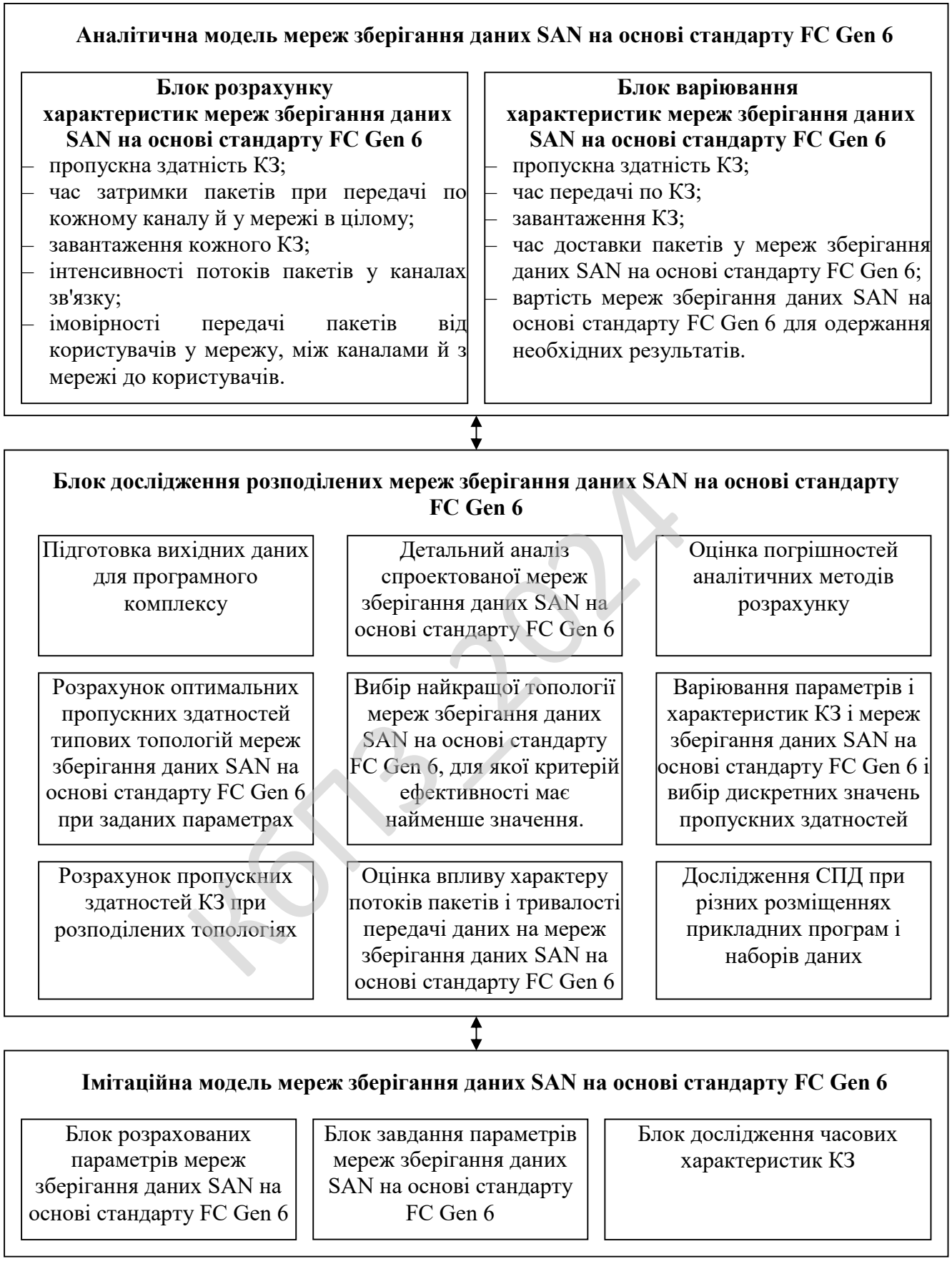


Рисунок 3.2 – Функціональна схема системи

Програма дозволяє варіювати отримані характеристики, такі як:

– пропускні здатності;

– час передачі по КЗ;

– завантаження КЗ;

– час доставки пакетів у мережі зберігання даних SAN на основі стандарту FC Gen 6;

– вартість мереж зберігання даних SAN на основі стандарту FC Gen 6 для одержання необхідних результатів.

Крім того, вона дозволяє вибрати значення пропускних здатностей з дискретного ряду значень, знайдених із традиційної оптимізації безперервних значень пропускних здатностей.

**Імітаційна модель мереж зберігання даних SAN на основі стандарту FC Gen 6** генерується автоматично на основі розрахунку характеристик аналітичної моделі у вигляді розімкнутої експонентної СеМО. Отримані характеристики аналітичної моделі використовуються як параметри імітаційної моделі мереж зберігання даних SAN на основі стандарту FC Gen 6, що представляє собою розімкнуту СеМО, вузли якої відповідають каналам зв'язку, а заявки – пакетам у мереж зберігання даних SAN на основі стандарту FC Gen 6. Число джерел заявок в імітаційній моделі дорівнює числу вузлів у мереж зберігання даних SAN на основі стандарту FC Gen 6, при цьому інтенсивності надходження заявок у моделі визначаються як зовнішні інтенсивності пакетів від користувачів до вузлів мереж зберігання даних SAN на основі стандарту FC Gen 6 відповідно. Аналітична модель, реалізована в програмі, дозволяє розрахувати наступні параметри для імітаційної моделі.

1. Імовірність передачі пакетів від користувача *j* до каналу *k*.

2. Імовірність передачі пакетів від каналу *k* до каналу *h*.

3. Імовірність передачі пакетів від каналу *k* до користувача *j*.

4. Середній інтервал часу між вступниками пакетами від користувача *j* у мереж зберігання даних SAN на основі стандарту FC Gen 6.

5. Середній час передач пакетів у каналах зв'язку *k*.

Крім **розрахованих параметрів** для імітаційної моделі мереж зберігання даних SAN на основі стандарту FC Gen 6 додатково необхідно **задати наступні параметри**:

а) закони розподілів інтервалів часу між вступниками пакетами від користувачів у мереж зберігання даних SAN на основі стандарту FC Gen 6 із коефіцієнтами варіації цих розподілів;

б) закони розподілів часу передачі пакетів у КЗ із коефіцієнтами варіації цих розподілів.

Відзначимо, що імітаційна модель мереж зберігання даних SAN на основі стандарту FC Gen 6 призначена для детального аналізу характеристик функціонування мережі, спроектованої в процесі аналітичного моделювання. При цьому, у випадку відмінності реального характеру процесів надходження пакетів у мережу або передачі пакетів по каналах зв'язку від експонентного, в імітаційній моделі передбачена можливість варіювання законів розподілу часу передачі (довжин обслуговування) пакетів у кожному із КЗ, а також законів розподілу інтервалів часу між вступними в мережу пакетами. Як такі закони в роботі використовувалися наступні розподіли: експонентний, детермінований; гіпоекспоненційний різного порядку й, відповідно, з різними коефіцієнтами варіації; рівномірний; експонентний з ненульовими зсувами; гіперекспонентний; Гамма-розподіл.

Аналогічно **розроблений засіб дослідження часових характеристик каналу зв'язку** шляхом генерування імітаційної моделі каналу зв'язку у вигляді СМО типу G/G/1 з можливістю зміни завантаження каналу й варіювання законів розподілу інтервалів часу між пакетами й часу передачі пакетів по каналі. Даний засіб дозволяє одержати значення часових характеристик каналу зв'язку й оцінити погрішність аналітичних методів розрахунку характеристик каналу зв'язку.

Сформульовано **методику дослідження розподілених мереж зберігання даних SAN на основі стандарту FC Gen 6** із неоднорідним трафіком, що містить наступні етапи.

1. Підготовка вихідних даних для програмного комплексу:

– кількість вузлів мережі і їхнє взаємне розташування;

– навантаження мереж зберігання даних SAN на основі стандарту FC Gen 6, створювана користувачами при роботі із прикладними програмами, наборами даних і в процесі обміну повідомлень;

– обмеження на середній час доставки пакетів або на вартість мереж зберігання даних SAN на основі стандарту FC Gen 6;

– максимальна довжина пакета;

– вибір типу каналів і завдання вартісних коефіцієнтів КЗ.

2. Розрахунок оптимальних пропускних здатностей типових топологій мереж зберігання даних SAN на основі стандарту FC Gen 6 при заданих наступних параметрах:

– типова топологія: зірка, кільце, дерево, повнозв’язна;

– модель взаємодії користувачів мережі: RDA (Remote Data Access), DBS (DataBase Server) і AS (Application Server);

– розподіл прикладних програм і наборів даних по вузлах мереж зберігання даних SAN на основі стандарту FC Gen 6;

– метод маршрутизації.

Після завдання цих параметрів розраховуються оптимальні значення пропускних здатностей КЗ, час передачі пакетів і завантаження каналів.

3. Розрахунок пропускних здатностей КЗ при розподілених топологіях.

4. Вибір найкращої топології мереж зберігання даних SAN на основі стандарту FC Gen 6, для якої обрана залежно від постановки завдання як критерій ефективності характеристика мереж зберігання даних SAN на основі стандарту FC Gen 6 (середній час доставки пакетів або вартість мереж зберігання даних SAN на основі стандарту FC Gen 6) приймає найменше значення.

5. Варіювання параметрів і характеристик КЗ і мереж зберігання даних SAN на основі стандарту FC Gen 6 і вибір дискретних значень пропускних здатностей.

6. Оцінка погрішностей аналітичних методів розрахунку характеристик КЗ.

7. Оцінка впливу характеру потоків пакетів і тривалості передачі даних на характеристики мереж зберігання даних SAN на основі стандарту FC Gen 6:

– вплив третього моменту розподілу вхідного потоку пакетів на характеристики КЗ;

– вплив довжини пакетів на характеристики каналів зв'язку й мереж зберігання даних SAN на основі стандарту FC Gen 6;

– вплив законів розподілів трафіку в мережах і часі передачі пакетів у КЗ на характеристики функціонування мереж зберігання даних SAN на основі стандарту FC Gen 6.

8. Дослідження мереж зберігання даних SAN на основі стандарту FC Gen 6 при різних розміщеннях прикладних програм і наборів даних.

9. Детальний аналіз спроектованої мереж зберігання даних SAN на основі стандарту FC Gen 6, що припускає варіювання параметрів мереж зберігання даних SAN на основі стандарту FC Gen 6, у тому числі: довжини запитів і відповідей прикладних програм і наборів даних, імовірність передачі по основному шляху, способи маршрутизації й т.д.

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



### 3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерської дипломної роботи, наведена на рисунку 3.3. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних

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

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

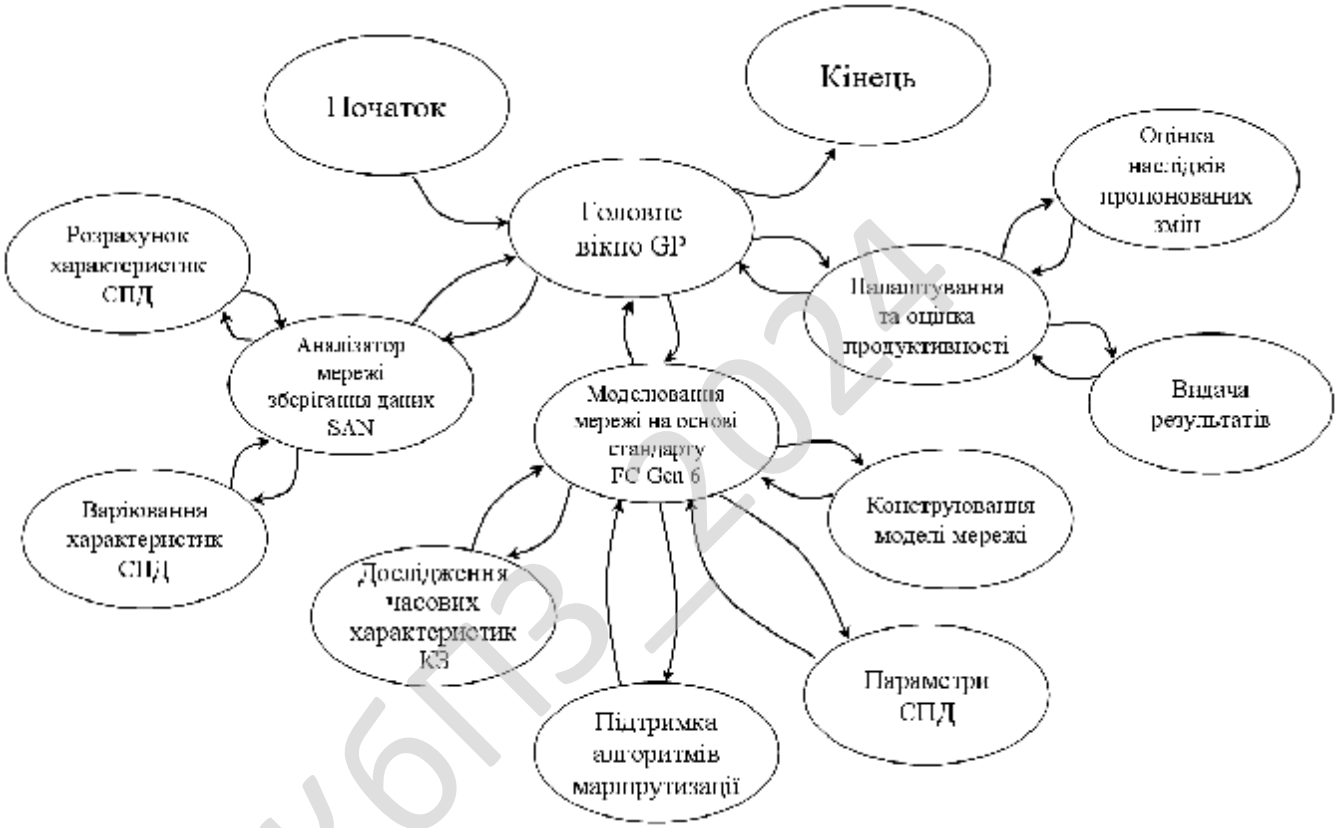


Рисунок 3.3 – Діаграма взаємодії процесів

Діаграми потоків даних містять чотири типи елементів: Процеси які являють собою трансформацію даних в рамках описуваної системи; Сховища даних (репозиторії); Зовнішні по відношенню до системи сутності; Потоки даних між елементами трьох попередніх типів.

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

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

## 4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 зображено роботу підпрограми.

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

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

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю мереж зберігання даних SAN на основі стандарту FC Gen 6.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою.

Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

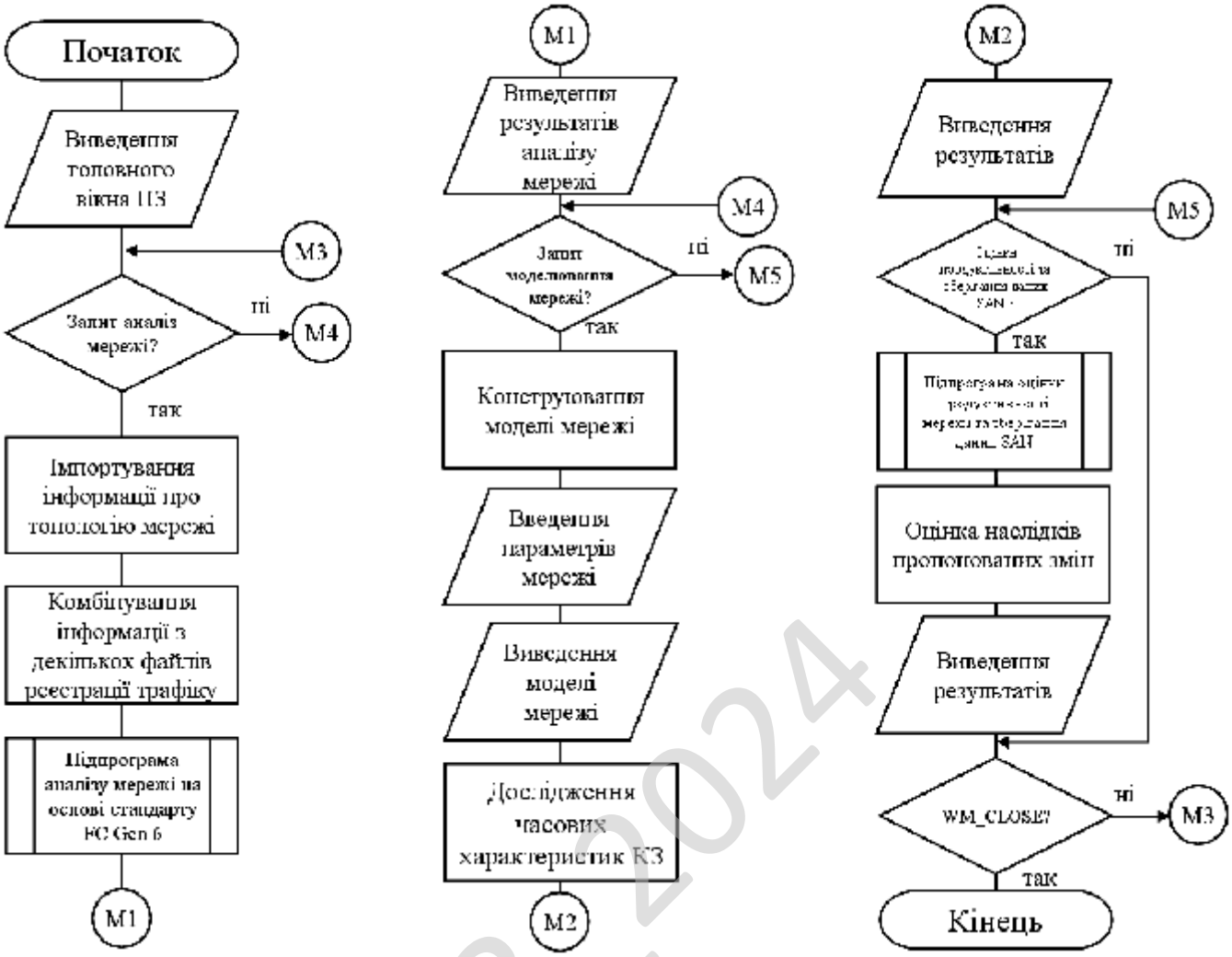


Рисунок 4.1 – Блок-схема основної програми

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

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

Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються.

Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

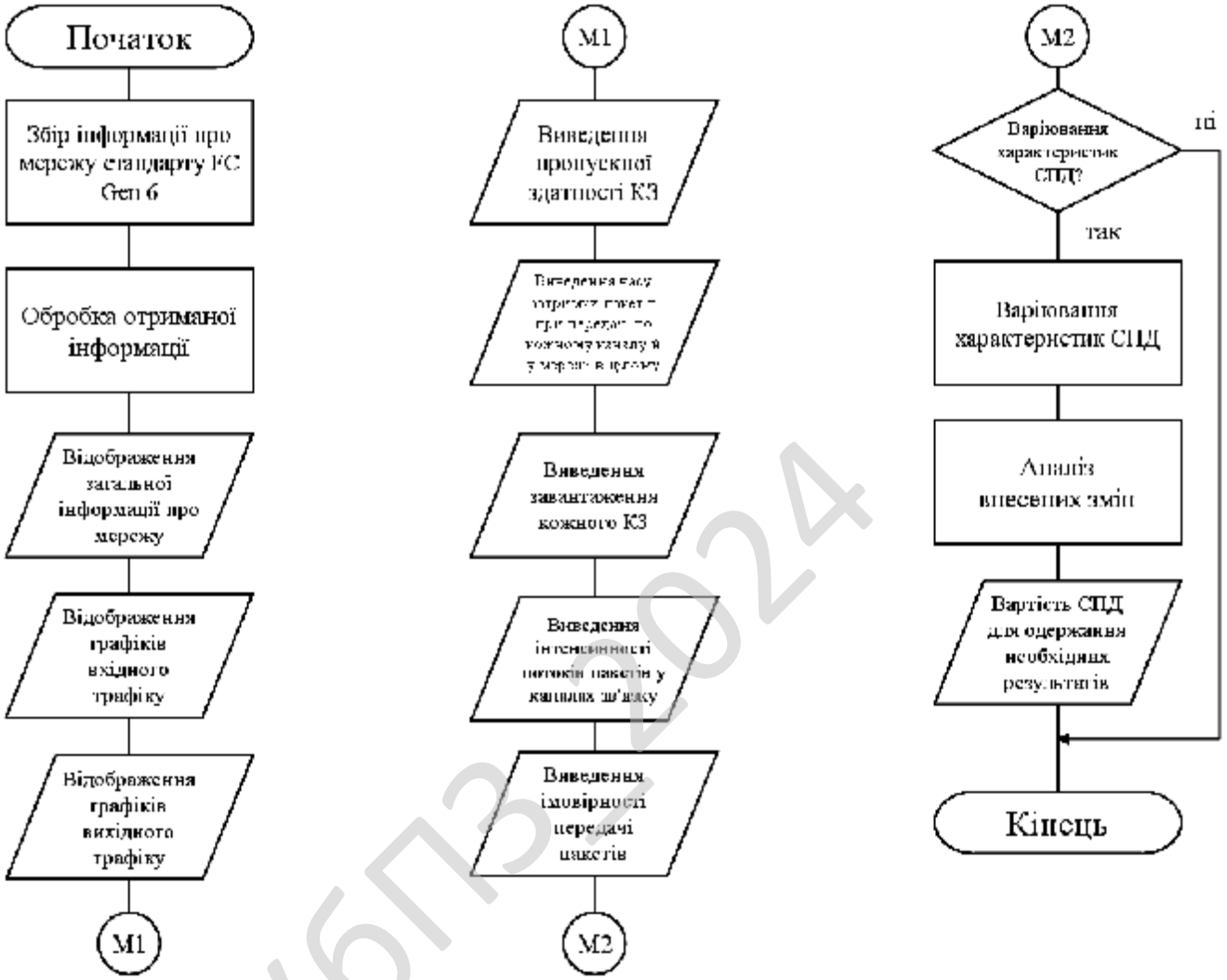




Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

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

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

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

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

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

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

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

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

– діаграма прецедентів (діаграми поведінки типу);

– діаграма класів;

– діаграма компонент.

**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.

Стандарт Fibre Channel Gen 6 (32GFC) забезпечує швидкість передачі даних до 32 Гбіт/с, але, звичайно, симуляція в Python не зможе безпосередньо керувати такою швидкістю. Ми можемо створити Python-програму для імітації передачі даних між "серверами" і "сховищами" у мережі SAN.

Розглянемо частину кода взаємодії системи мереж.

```
import random
import time

class FibreChannelNode:
    def __init__(self, node_id, role):
        self.node_id = node_id
        self.role = role  # 'host' or 'storage'

    def send_data(self, data, target):
print(f"[Node {self.node_id}] Sending data to {target.node_id}: {data}")
time.sleep(random.uniform(0.1, 0.5))
# Симуляція затримки у передачі даних
target.receive_data(data, self)

def receive_data(self, data, source):
print(f"[Node {self.node_id}] Received data from {source.node_id}: {data}")

class FibreChannelSwitch:
    def __init__(self, switch_id):
        self.switch_id = switch_id
        self.connected_nodes = []

def connect_node(self, node):
self.connected_nodes.append(node)
print(f"[Switch {self.switch_id}] Node {node.node_id} connected.")

    def transfer_data(self, data, source_node, target_node_id):
        target_node = next((node for node in self.connected_nodes if node.node_id
== target_node_id), None)
        if target_node:
print(f"[Switch {self.switch_id}] Routing data from {source_node.node_id} to
```

```
{target_node.node_id}")
            source_node.send_data(data, target_node)
        else:
print(f"[Switch {self.switch_id}] Target node {target_node_id} not found.")

# Створення вузлів
host = FibreChannelNode(node_id=1, role='host')
storage = FibreChannelNode(node_id=2, role='storage')

# Створення комутатора
fc_switch = FibreChannelSwitch(switch_id=101)

# Підключення вузлів до комутатора
fc_switch.connect_node(host)
fc_switch.connect_node(storage)

# Передача даних
fc_switch.transfer_data(data="Backup file", source_node=host, target_node_id=2)
```

FibreChannelNode представляє вузол у SAN, який може бути або хостом (сервером), або сховищем. Кожен вузол може передавати і отримувати дані.

FibreChannelSwitch представляє комутатор Fibre Channel, який з'єднує вузли та маршрутизує дані між ними.

Симуляція після підключення вузлів до комутатора, хост надсилає дані сховищу через комутатор.

Додавання функції логічного розділення трафіку (зони безпеки). У SAN-системах є поняття зонування (zoning), яке дозволяє ізолювати трафік між групами пристроїв для підвищення безпеки, розглянемо функцію.

```
class FibreChannelZone:
    def __init__(self, zone_id):
        self.zone_id = zone_id
        self.members = []

    def add_member(self, node):
        self.members.append(node)
print(f"[Zone {self.zone_id}] Node {node.node_id} added to the zone.")

    def is_member(self, node):
        return node in self.members
```

```
class FibreChannelSwitchWithZones(FibreChannelSwitch):
    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.zones = {}

def create_zone(self, zone_id):
        self.zones[zone_id] = FibreChannelZone(zone_id)
print(f"[Switch {self.switch_id}] Zone {zone_id} created.")

def add_node_to_zone(self, zone_id, node):
if zone_id in self.zones:
            self.zones[zone_id].add_member(node)
else:
print(f"[Switch {self.switch_id}] Zone {zone_id} does not exist.")

def transfer_data_in_zone(self, data, source_node, target_node_id,
zone_id):
if zone_id not in self.zones:
print(f"[Switch {self.switch_id}] Zone {zone_id} does not exist.")
            return

if self.zones[zone_id].is_member(source_node):
target_node = next((node for node in self.zones[zone_id].members if
node.node_id == target_node_id), None)
if target_node:
print(f"[Switch {self.switch_id}] Transferring data within Zone {zone_id} from
{source_node.node_id} to {target_node.node_id}")
                source_node.send_data(data, target_node)
            else:
print(f"[Switch {self.switch_id}] Target node {target_node_id} not found in Zone
{zone_id}")
        else:
print(f"[Switch {self.switch_id}] Source node {source_node.node_id} not authorized
in Zone {zone_id}")
```

**Особливості Fibre Channel Gen 6**

Швидкість – Fibre Channel Gen 6 підтримує швидкість передачі до 32 Гбіт/с. Це забезпечує зменшення часу доступу до даних і дозволяє обробляти великі обсяги інформації в реальному часі.

Надійність – SAN на основі Fibre Channel відома своєю стабільністю і

високим рівнем відмовостійкості. Комутатори і з'єднання можуть бути налаштовані для резервування, що мінімізує простої в разі відмови одного з елементів мережі.

Безпека – Fibre Channel підтримує різні методи аутентифікації та шифрування для забезпечення безпечного обміну даними між хостами та сховищами.

Масштабованість – Мережі SAN можуть масштабуватися від невеликих систем до великих корпоративних інфраструктур. З підвищенням попиту на обробку даних можна додавати нові сервери, сховища та комутатори без значних змін в існуючій архітектурі.

Ізоляція трафіку – У SAN трафік даних між хостами і сховищами повністю ізольований від решти мережі, що дозволяє уникнути перевантаження загальної корпоративної мережі і підвищує безпеку даних.

**Архітектура Fibre Channel**

Топологія fabric. Це найпоширеніша топологія, де хости та сховища підключені через комутатори Fibre Channel, утворюючи "тканину" (fabric). Вузли можуть взаємодіяти між собою через комутатори, що дозволяє гнучко керувати маршрутами передачі даних.

Point-to-point. У цьому варіанті хост напряму підключений до сховища через один канал. Цей підхід зазвичай застосовується для невеликих мереж, де висока пропускна здатність не є критичною.

Arbitrated Loop. Топологія кільця, де всі пристрої з'єднані один з одним у вигляді кільця. Вона є менш поширеною і зазвичай використовується у старіших або малих системах.

**Можливості системи SAN на основі Fibre Channel Gen 6**

Зберігання великих обсягів даних. SAN-мережі можуть легко підтримувати велике централізоване сховище, доступне для багатьох серверів одночасно.

Висока швидкість резервного копіювання. Швидкісні з'єднання Gen 6

дозволяють здійснювати резервне копіювання даних практично миттєво, навіть для великих систем баз даних.

Інтенсивні бази даних. Сервери баз даних, які обробляють мільйони запитів за хвилину, можуть використовувати SAN для швидкого доступу до необхідних об'ємів даних без навантаження на локальне сховище.

Віртуалізація. SAN надає можливості для зберігання даних віртуальних машин та керування ресурсами сховищ у великих датацентрах.

### 4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.3).

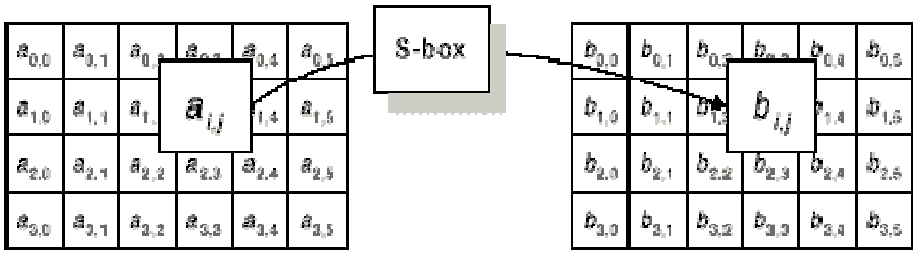


Рисунок 4.3 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.4).

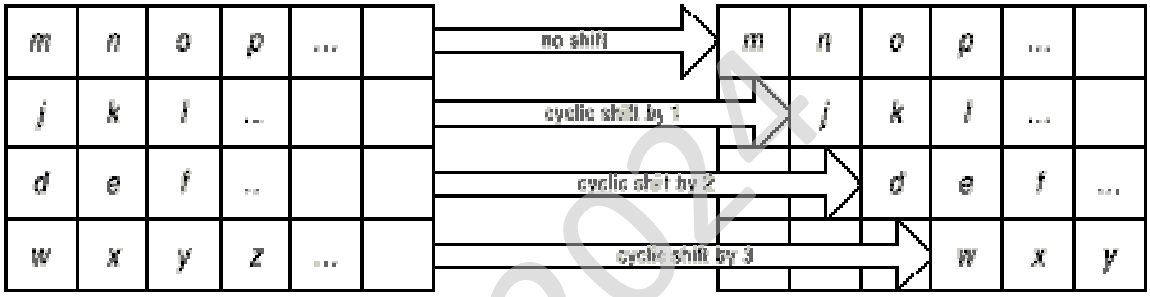


Рисунок 4.4 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.5).

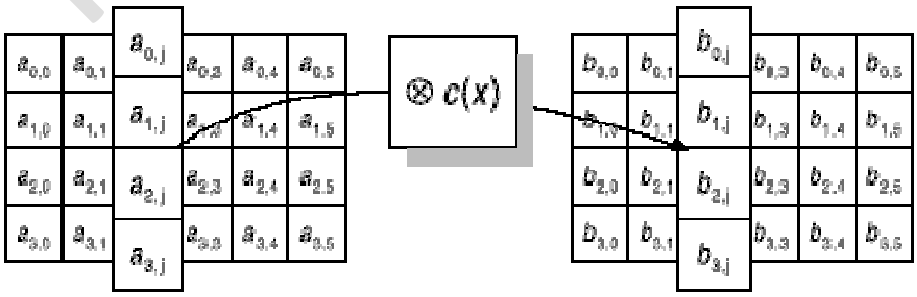


Рисунок 4.5 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.6).

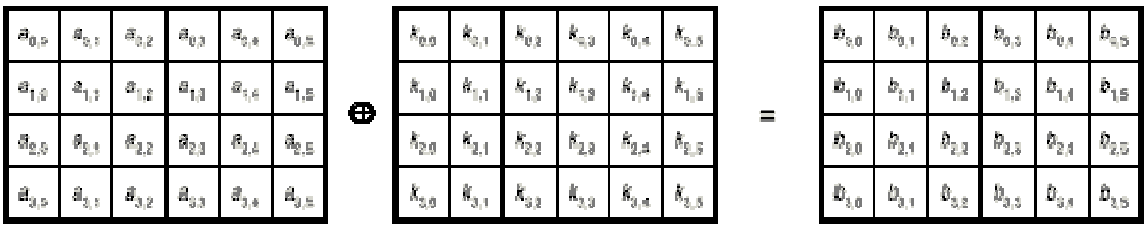

Рисунок 4.6 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

# 5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання магістерської дипломної роботи. Основні дії роботи програми відбуваються у консолі проте, налаштування проходять у інтерфейсі.

Розроблене програмне забезпечення мереж зберігання даних SAN на основі стандарту FC Gen 6 складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Аналізатор мережі; Моделювання мережі; Оцінка продуктивності; Параметри; Довідка.

– Функції представлені у графічному вигляді.

– Розділу вікна підключення мережного інтерфейсу.

– Розділу вікна Інформації про поточний мережний інтерфейс.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Деревовидного представлення зберігання даних SAN на основі стандарту FC Gen 6.

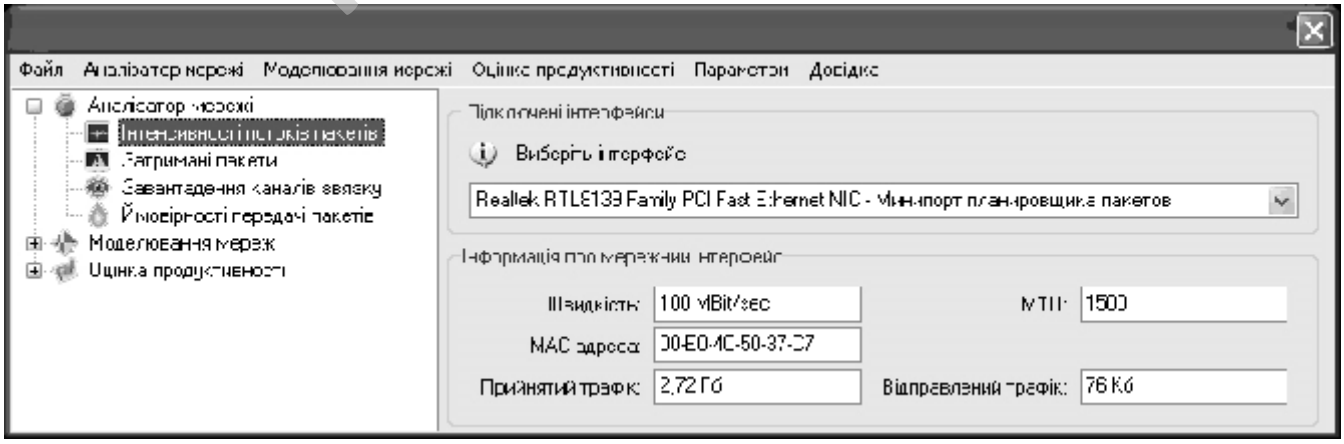


Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

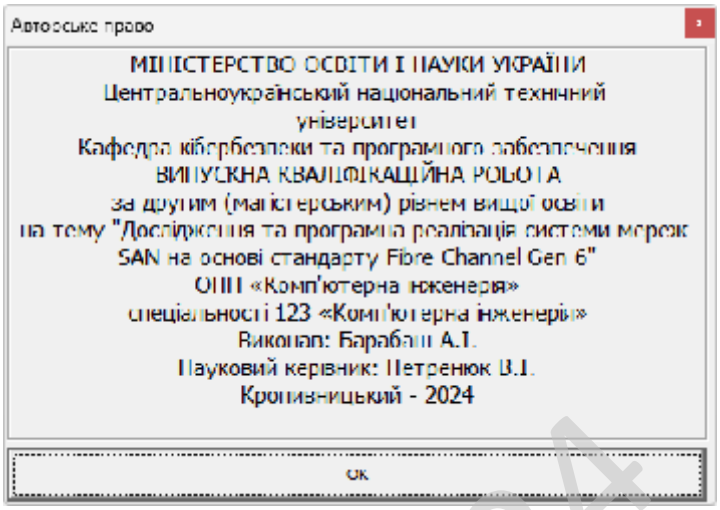


Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

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

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

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

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

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

– У програмі можуть бути пропущені деякі маршрути.

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

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

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

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

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

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

Обрано умови розповсюдження – commercial software.

Програмне забезпечення, створене комерційною організацією з метою отримання прибутку від його використання іншими, наприклад, шляхом продажу копій.

Найважливішою особливістю комерційний програмних продуктів є підтримка великих компаній, прямо зацікавлених у поширенні програм. Багато організацій надають виключно платну підтримку своїх продуктів, такий підхід, як правило, використовують організації надають відкриті вихідні коди. Для продуктів, що розповсюджуються на комерційній основі діють зазвичай безкоштовні служби підтримки, покликані збільшити рівень довіри у клієнтів і потенційних покупців.

Далеко не завжди, але як правило терміни критично важливих змін в комерційних продуктах значно менше, ніж у некомерційних проектів. Це пов'язано з тим, що над комерційним продуктом працюють цілі групи розробників і ця робота є їх основним заняттям. Розробникам-початківцям як правило доводиться шукати додаткові способи заробітку, і це збільшує час, що витрачається на доповнення і зміни програм. Так як основним рушійним фактором створення комерційного ПЗ є одержання прибутку, то комерційні програмні продукти першими заповнюють вільні ніші та пропонують варіанти вирішення завдань відразу по мірі виявлення вакууму в будь-якому секторі ринку.

Окремий вид комерційних програм, коли їх розробка оплачується безпосередньо замовником. Такі програми найчастіше позбавлені всіх переваг комерційних продуктів, оскільки мають обмежений бюджет, але більш адаптовані до вимог замовника, ніж аналоги.

# 6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи мереж SAN на основі стандарту Fibre Channel Gen 6.

*Метою розробки* є дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6.

*Об'єктом дослідження* є процес мереж SAN на основі стандарту Fibre Channel Gen 6.

*Предметом дослідження* є методи мереж SAN на основі стандарту Fibre Channel Gen 6.

*Методи дослідження* базуються на методах теорії побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод мереж SAN на основі стандарту Fibre Channel Gen 6.

– Розроблено вітчизняний продукт мереж SAN на основі стандарту Fibre Channel Gen 6, який має більш широкі можливості, на відміну від існуючих аналогів.

# 7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

## 7.1 Визначення цільової аудиторії кінцевого готового продукту

Результати дослідження та програмної реалізації системи мереж SAN (Storage Area Network) на основі стандарту Fibre Channel Gen 6 можуть зацікавити кілька груп фахівців і організацій. Схематично їх подано на рисунку 7.1.

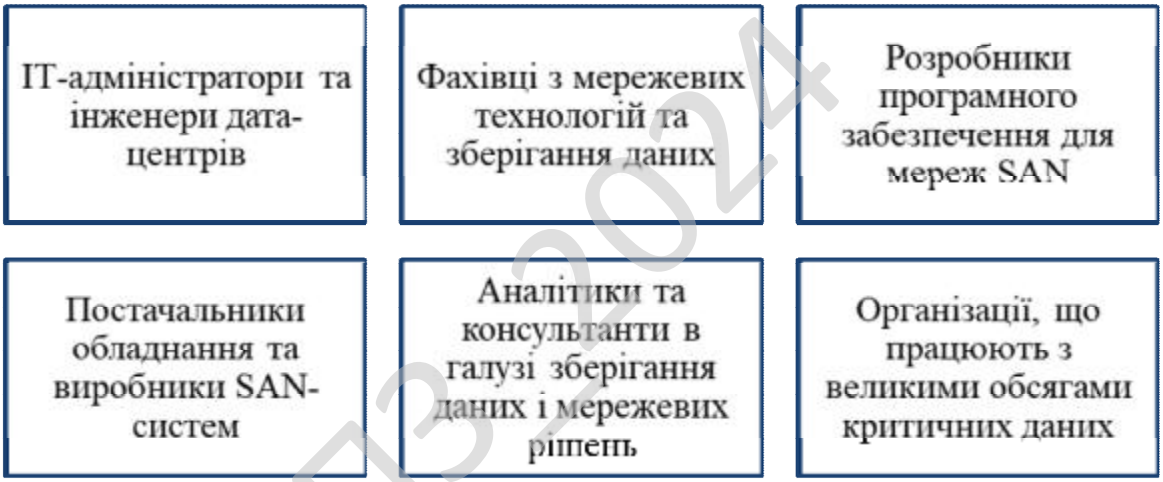


Рисунок 7.1 – Ключові сегменти цільової аудиторії

Таким чином, дослідження та програмна реалізація систем SAN на базі Fibre Channel Gen 6 можуть зацікавити різноманітні категорії професіоналів, зокрема в галузі ІТ-інфраструктури, управління даними, мережевих рішень, а також тих, хто працює над підвищенням продуктивності та надійності систем зберігання даних у різних секторах економіки.

## 7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості програмної реалізації системи мереж SAN на основі стандарту Fibre Channel Gen 6 за допомогою методів експертних оцінок пройде наступні етапи: вибір критеріїв, формування групи експертів, **п**рисвоєння балів за кожним критерієм, підрахунок оцінки. За критерії обираємо: продуктивність (пропускна здатність, швидкість передачі даних), надійність, вартість впровадження та обслуговування, масштабованість системи, сумісність з існуючими рішеннями, рівень безпеки, інноваційність та перспективність стандарту, зручність інтеграції та налаштування. Запрошуємо експертну думку від: ІТ-спеціаліста та адміністратора дата-центрів, фахівця з мережевих технологій та зберігання даних, представника компаній-виробників SAN-рішень, консультанта у сфері інфраструктури зберігання даних.

Таблиця 7.1 – Зведені експертні дані, де 1 – найменша привабливість, 10 – найвища привабливість

| **Критерій** | **Вага критерію (%)** | **Оцінка (1–10)** | **Зважена оцінка** |
|---|---|---|---|
| Продуктивність | 25% | 9 | 2.25 |
| Надійність та відмовостійкість | 20% | 8 | 1.6 |
| Вартість впровадження | 10% | 6 | 0.6 |
| Масштабованість | 15% | 9 | 1.35 |
| Сумісність з існуючими рішеннями | 10% | 7 | 0.7 |
| Рівень безпеки | 10% | 8 | 0.8 |
| Інноваційність | 5% | 9 | 0.45 |
| Зручність інтеграції | 5% | 7 | 0.35 |

Загальна оцінка привабливості = 2.25 + 1.6 + 0.6 + 1.35 + 0.7 + 0.8 + 0.45 + 0.35 = 8.1 з 10 можливих. Оцінка 8.1 свідчить про високу привабливість впровадження SAN на основі Fibre Channel Gen 6 для організацій, які шукають рішення з високою продуктивністю, масштабованістю та надійністю.

### 7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи мереж SAN на основі стандарту Fibre Channel Gen 6 можна використовувати кілька методів залежно від потреб проекту, доступних ресурсів, і рівня деталізації оцінки: метод аналогій, параметричний метод, метод оцінки «знизу вгору», метод трьох оцінок, метод на основі життєвого циклу. Враховуючи високу невизначеність та обмеженість ресурсів в рамках магістерського проєкту оптимальним є метод аналогій.

### 7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Впровадження системи мереж SAN (Storage Area Network) на основі стандарту Fibre Channel Gen 6 може мати значний економічний ефект для організації. Для оцінки економічної ефективності можна використовувати різні підходи, серед яких розрахунок загальної вартості утримання (TCO), повернення на інвестиції (ROI), або періоду окупності (Payback Period).

Таблиця 7.2 – Вхідні дані для розрахунку економічної ефективності від впровадження SAN на основі Fibre Channel Gen 6

| Дані про систему |
|---|
| – компанія вирішує впровадити san на основі fibre channel gen 6 для централізованого управління зберіганням даних.<br>– поточна інфраструктура використовує локальні дискові масиви з низькою продуктивністю, що призводить до частих затримок в доступі до даних і високих витрат на управління.<br>– кількість серверів у системі: 50.<br>– необхідний обсяг зберігання даних: 500 Тб.<br>– річні витрати на обслуговування поточної інфраструктури: 150,000 ₴ |
| Витрати на впровадження нової SAN |
| – вартість обладнання (SAN-комутатор, дискові масиви, кабелі): 300,000₴.<br>– вартість ліцензування програмного забезпечення: 50,000₴.<br>– витрати на інтеграцію та налаштування системи: 30,000₴.<br>– навчання персоналу: 10,000₴. |
| Загальні початкові витрати: 390,000₴. |
| Річні експлуатаційні витрати |
| – обслуговування нової SAN-системи: 50,000₴ (значно менше завдяки автоматизації та централізації).<br>– витрати на енергію та охолодження: 20,000₴. |
| Загальні річні витрати: 70,000₴. |

Продовження таблиці 7.2

| Економія завдяки впровадженню SAN |
|---|
| – швидкість передачі даних зросла завдяки Fibre Channel Gen 6 до 32 Гбіт/с (порівняно з 8 Гбіт/с у старій системі).<br>– це дозволило скоротити час на доступ до даних, що прискорило виконання бізнес-процесів і знизило кількість простоїв.<br>– зменшення експлуатаційних витрат:<br>– річні витрати на обслуговування знизились з 150,000₴ до 70,000₴, що забезпечує економію у 80,000₴ на рік.<br>– скорочення витрат на управління інфраструктурою:<br>– завдяки централізації управління даними в SAN витрати на IT-персонал знизились на 30,000₴ на рік.<br>– З новою SAN-системою додавання нових серверів або розширення обсягу зберігання стало простішим та дешевшим, що скоротило витрати на майбутні апгрейди. |

Розрахунок ключових показників економічної ефективності.

Загальний економічний ефект: економія на обслуговуванні за перший рік: 80,000₴, економія на IT-персоналі за перший рік: 30,000₴.

Загальна економія за перший рік = 80,000₴ + 30,000₴ = 110,000₴.

Інвестиції у впровадження: 390,000₴.

Повернення на інвестиції (ROI). ROI==28.2%

Період окупності (Payback Period). Річна економія = ₴110,000. Період окупності = 3.5 роки

Загальна вартість утримання (TCO). Порівняння витрат за 5 років: стара система (річні витрати: 150,000₴, за 5 років: 150,000 × 5 = 750,000₴), нова SAN-система (початкові витрати - 390,000₴, річні витрати - 70,000₴, за 5 років - 390,000 + (70,000 × 5) = 390,000 + 350,000 = 740,000₴. Таким чином, TCO нової

системи за 5 років трохи менший за стару систему, але нова SAN надає значні переваги в продуктивності, надійності та масштабованості.

Висновки щодо доцільності впровадження:

1. Період окупності становить трохи більше ніж 3,5 років, після чого організація почне отримувати чисту економію.

2. ROI показує, що впровадження SAN на основі Fibre Channel Gen 6 є вигідною інвестицією з поверненням 28.2% протягом першого року.

3. Загальна вартість володіння (TCO) за 5 років є конкурентною порівняно зі старою інфраструктурою, але нова SAN забезпечує значно вищу продуктивність та ефективність.

Таким чином, впровадження SAN на основі Fibre Channel Gen 6 є економічно вигідним рішенням, особливо для компаній з великими обсягами даних і потребами у високій продуктивності та надійності систем зберігання.

### 7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Для успішного просування проєкту програмної реалізації системи мереж SAN на основі стандарту Fibre Channel Gen 6**,** можна використовувати наступний алгоритм, що охоплює ключові етапи маркетингового просування і залучення зацікавлених сторін. Цей підхід дозволить ефективно представити проект потенційним клієнтам, інвесторам та партнерам, а також забезпечити його конкурентоспроможність на ринку.

Такий алгоритм просування допоможе покроково вивести проект програмної реалізації SAN на основі Fibre Channel Gen 6 на ринок. Він враховує як технічні, так і маркетингові аспекти, що дозволить ефективно комунікувати переваги рішення, залучати зацікавлених клієнтів і підтримувати конкурентну позицію на ринку SAN-мереж.

| | |
|---|---|
| **1.** Аналіз ринку та цільової аудиторії | Провести дослідження ринку SAN-рішень, аналізуючи актуальні тренди та потреби організацій |
| | Визначити основні сегменти споживачів |
| | Визначити конкурентів та їх пропозиції |
| | Скласти профілі ключових клієнтів та визначити їхні потреби |
| 2. Створення унікальної торговельної пропозиції (USP) | Виділити ключові технічні переваги Fibre Channel Gen 6 (швидкість до 32 Гбіт/с, низька затримка, підвищена надійність). |
| | Підкреслити конкурентні переваги, такі як легка масштабованість, зниження операційних витрат, підвищення продуктивності. |
| | Створити коротке та запам'ятовуване формулювання основної вигоди для клієнта, що відображає переваги використання саме вашої SAN-системи. |
| 3. Планування маркетингової стратегії | **Контент-маркетинг:** Створення технічної документації, білих книг, статей та аналітики, що пояснюють переваги Fibre Channel Gen 6 для бізнесу. |
| | **Соціальні мережі та блоги:** Ведення блогу про SAN-інфраструктуру, публікація статей на LinkedIn, участь у спеціалізованих форумах та групах для залучення професійної аудиторії. |
| | **Вебінари та презентації:** Проведення онлайн-заходів для інформування потенційних клієнтів про вигоди від переходу на Fibre Channel Gen 6. |
| | **Партнерські програми:** Встановлення партнерств з виробниками апаратного забезпечення та системних інтеграторів для просування спільних рішень. |
| 4. Визначення каналів продажу та партнерства | **Прямі продажі:** Побудова відділу продажів для прямих контактів з великими корпоративними клієнтами. |
| | **Канальні партнери:** Співпраця з системними інтеграторами та VAR (Value Added Resellers), що мають досвід впровадження SAN-систем. |
| | **Дистрибуція через інтернет:** Створення спеціалізованого інтернет-майданчика для продажу SAN-систем та послуг для невеликих та середніх компаній. |
| 5. Визначення бізнес-моделі і цінової політики | Провести аналіз цін конкурентів на схожі SAN-рішення (як на Fibre Channel, так і на інші типи мереж). |
| | Розробити кілька варіантів цінових пропозицій (ліцензування, разова оплата за обладнання, модель підписки з регулярними платежами). |
| | Запровадити програми знижок та бонусів для ранніх клієнтів або лояльних партнерів. |
| 6. Створення демонстраційного проєкту або пілотного рішення | Створити тестовий стенд або демо-систему SAN, яку потенційні клієнти зможуть протестувати на власному обладнанні або у віртуальному середовищі. |
| | Запропонувати спеціальні умови для проведення пілотного впровадження у клієнта. |
| | Публікувати кейси успішних впроваджень для підвищення довіри та демонстрації економічної ефективності. |
| 7. Зворотний зв'язок та аналіз результатів | Аналізувати відгуки від потенційних клієнтів, партнерів та інвесторів. |
| | Оцінити кількість лідів, конверсій, кількість впроваджень і рентабельність проєкту. |
| | Удосконалити маркетингові та продажні стратегії на основі отриманих даних. |
| 8. Моніторинг і підтримка проекту | Запровадити регулярну технічну підтримку для клієнтів, які впровадили SAN-систему. |
| | Розробити програми оновлення і модернізації для постійного підвищення продуктивності. |
| | Відстежувати нові ринкові можливості та вводити нові функціональні можливості для підтримки актуальності продукту. |

Рисунок 7.2 – Алгоритм просування проєкту SAN на основі Fibre Channel Gen 6

### 7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Оптимізація каналів збуту та шляхів реалізації проекту програмної реалізації системи мереж SAN (Storage Area Network) на основі стандарту Fibre Channel Gen 6 вимагає глибокого аналізу ринку, вибору відповідних каналів і стратегії маркетингу, а також інтеграції з інноваційними інструментами.

Оптимізація каналів збуту та шляхів реалізації ПЗ потребуватиме: модернізації існуючих каналів збуту та розвитку мультиканального підходу.

Модернізація каналів збуту передбачатиме: залучити спеціалізовані команди продажів, що можуть безпосередньо працювати з великими корпоративними клієнтами, пропонуючи індивідуальні рішення для їх інфраструктури; встановити партнерські відносини з системними інтеграторами, VAR (Value Added Resellers) та дистриб'юторами, що мають досвід у впровадженні SAN-систем, для збільшення охоплення; створити інтернет-платформу для просування продукту через спеціалізовані інтернет-магазини, що пропонують IT-рішення для бізнесу; розробити модель ліцензування або франчайзингу, що дозволить партнерам розширювати продажі SAN-систем на локальних ринках.

Розвиток мультиканального підходу потребує комбінації фізичних та цифрових каналів збуту та створення порталу самообслуговування для клієнтів, де вони зможуть переглядати продукти, порівнювати конфігурації, отримувати технічну підтримку, здійснювати замовлення та слідкувати за статусом замовлення в режимі реального часу. Також важливо забезпечити технічну підтримку, пропонувати інтегровані рішення та проводити освітні заходи для підвищення обізнаності клієнтів про переваги нових SAN-технологій.

### 7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проекту програмної реалізації системи мереж SAN на основі стандарту Fibre Channel Gen 6 першочергово мають стати фактори наведені на рисунку 7.3.

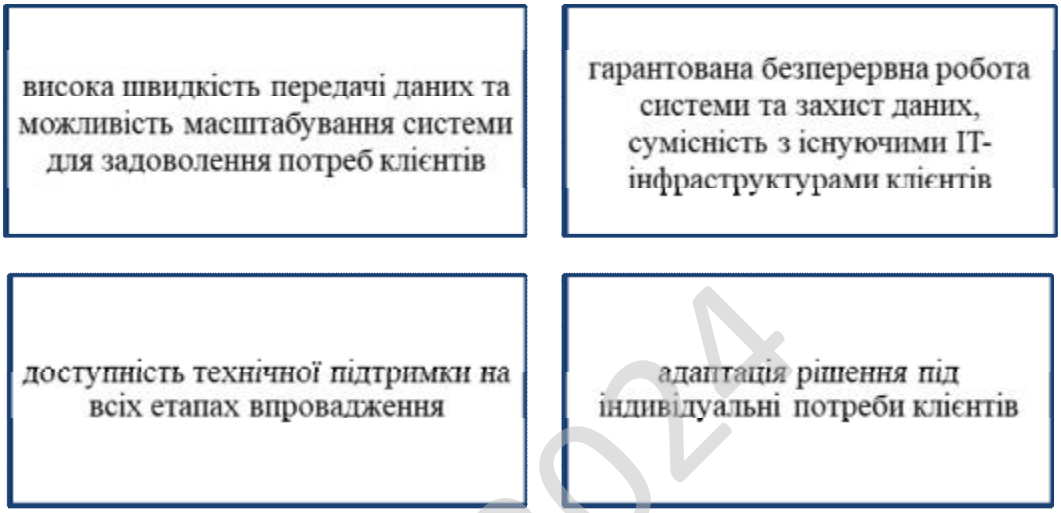


Рисунок 7.3 – Ключові фактори успіху проєкту

Розбудова стратегії просування проєкту з урахуванням наведених факторів дозволить отримати позитивний результат.

# 8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

## 8.1 Вступ

Під час роботи із системою мереж SAN на основі стандарту Fibre Channel Gen 6необхідно дотримуватись вимог з охорони праці при роботі з ПК, а також техніки безпеки та протипожежної безпеки при використанні ЕОМ та комп'ютерної техніки.

Законом України "Про охорону праці" [1] регламентуються загальні положення державної політики в галузі охорони праці, а реалізуються 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» [3], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

У розділі даної магістерської роботи висвітлюються основні питання охорони праці працівників, робота яких пов'язана з роботою за комп'ютером, планування робочого приміщення, де працюють користувачі ПК; параметри мікроклімату, освітленість робочих місць та виробничих приміщень; шумові завади.

Для того, щоб об'єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Робота з комп'ютером характеризується значною розумовою напругою і нервово-емоційним навантаженням операторів, високою напруженістю зорової роботи і достатньо великим навантаженням на м'язи рук при роботі з клавіатурою ЕОМ.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

## 8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо вимоги до приміщень, в яких буде розміщуватись робоче місце. Площа на одне робоче місце повинна становити мінімум 6,0 м$^2$, при цьому об’єм – мінімум 20,0 м$^3$ [2, 4]. Розміщення робочих місць у підвальних приміщеннях, а також на цокольних поверхах заборонено. У відповідності до НПАОП 0.00-7.15-18, приміщення повинні мати природне та штучне освітлення. Приміщення не повинні межувати з іншими приміщеннями, в яких рівні шуму і вібрації перевищують допустимі значення. Покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3-0,5. Для внутрішнього оздоблення приміщень слід використовувати дифузно-відбивні матеріали з коефіцієнтами відбиття для стелі 0,7-0,8, для стін 0,5-0,6 [4]. Саме робоче місце теж повинно відповідати вимогам, що описані в [2, 4]. Конструкція робочого місця повинна забезпечити підтримання оптимальної робочої пози. У відповідності до НПАОП 0.00-7.15-18, обладнання і організація робочого місця працюючих з ЕОМ мають забезпечувати відповідність конструкції всіх елементів робочого місця та їх взаємного, розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності.

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 | 5,4 |
| Довжина | 6 |
| Висота | 2,75 |

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 | 8,1 |
| Обсяг, V | $м^3$ | не менше 20.0 | 24,3 |

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [4].

У зазначеному приміщенні працюють 4 людей. 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 не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [4], але 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» [3] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

### 8.3 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розміри приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значень;

Таким чином можна припустити, що основною причиною можливого 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).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

Приміщення повинні бути оснащені аптечками першої медичної допомоги, а також обов'язковим є щоденне вологе прибирання приміщень.

## 8.4 Техніка безпеки та протипожежна профілактика

Електронно-обчислювальні машини (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у тому числі програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

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;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у тому числі високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

Вимоги безпеки при роботі з ПК визначено в НПАОП 0.00-1.28-18. Згідно вимог електробезпеки, ПК повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань. Не допускається підключати ПК до

звичайної двопровідної електромережі, в тому числі з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення ПК потрібно виконувати за магістральною схемою. Щодо безпеки при роботі з ПК, щодня перед початком роботи необхідно очищати монітор від пилу та інших забруднень. Після закінчення роботи з ПК, він та периферійні пристрої повинні бути відключені від електричної мережі. У разі виникнення певної аварійної ситуації необхідно негайно відключити ПК від електричної мережі. Не допускається виконувати обслуговування, ремонт та налагодження ПК безпосередньо на робочому місці [2]. Згідно з [4], на та під приміщеннями, в яких розміщені ЕОМ, а також у суміжних із ними приміщеннях не дозволяється розташування приміщень категорій А та Б за вибухопожежною небезпекою. Фальшпідлога у приміщеннях з ЕОМ має бути з негорючих матеріалів або матеріалів груп горючості Г1, Г2 з межею вогнестійкості не менше 0,5 години. Простір під нею слід розділяти негорючими діафрагмами на відсіки площею не більше 250 м$^2$. Діафрагми повинні мати межу вогнестійкості не менше 0,75 год. Звукопоглинаюче облицювання стін та стель цих приміщень слід виготовляти з негорючих матеріалів або матеріалів груп горючості Г1, Г2. Персональні комп'ютери після закінчення роботи повинні відключатися від мережі. Не рідше одного разу на квартал необхідно очищати від пилу агрегати та вузли, кабельні канали та простір між підлогами [4]. Приміщення повинні бути забезпечені первинними засобами пожежогасіння, а саме вогнегасниками, що використовуються для локалізації і ліквідації пожеж у їх початковій стадії розвитку.

Вогнегасники слід встановлювати у легкодоступних та помітних місцях (коридорах, біля входів або виходів з приміщень тощо), а також у пожежонебезпечних місцях, де найбільш вірогідна поява осередків пожежі. При цьому необхідно забезпечити їх захист від попадання прямих сонячних променів та безпосередньої (без загороджувальних щитків) дії опалювальних та нагрівальних приладів. Вибір типу та необхідна кількість вогнегасників

визначається відповідно до Типових норм належності вогнегасників. Вогнегасники, допущені до введення в експлуатацію, повинні мати:

– облікові (інвентарні) номери за прийнятою на об'єкті системою нумерації;

– пломби на пристроях ручного пуску;

– бирки та маркувальні написи на корпусі, червоне сигнальне пофарбування згідно з державними стандартами [4].

У відповідності до [4] не дозволяється:

– відкрите прокладання електропроводів і кабелів транзитом через пожежонебезпечні і вибухонебезпечні зони будь-якого класу і ближче 1 м і 5 м від них відповідно, а також у сходових клітках;

– експлуатація кабелів і проводів з пошкодженою або такою, що в процесі експлуатації втратила захисні властивості, ізоляцією;

– залишення під напругою кабелів та проводів з неізольованими струмопровідними жилами;

– застосування саморобних подовжувачів, які не відповідають вимогам ПУЕ, що пред'являються до переносних (пересувних) електропроводок;

– заклеювати шпалерами відкрито прокладені електропроводи і кабелі.

### 8.5 Розрахункова частина

Для забезпечення безпечної роботи необхідне виконання вимог електричної безпеки, оскільки все офісне обладнання заживлюється від електричної мережі. Одним з необхідних засобів електричної безпеки є встановлення захисного заземлення.

Початкові дані, необхідні для розрахунку захисного заземлення:

– допустимий опір розповсюдженню струму в землі від заземлю-вального пристрою $R_{зн}$ = 10 Ом;

– питомий опір грунту в місці встановлення заземлювача $\rho_з$ = 100 Ом/м;

– тип ґрунту – суглинок;

– тип заземлювача – труба, діаметром d=0.05 м і довжиною l =2.5 м; – конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_3$$

де $\phi$ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 2.5 м при глибині закладання від вершини h = 0.5 м $\phi$ = 1.1 для четвертої кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

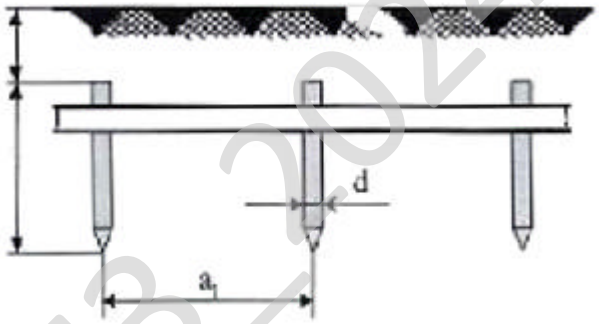


Рисунок 8.1 – Схема розташування заземлювачів

Опір землі:

$$\rho_{рз} = 1{,}1 \cdot 100 = 110 \text{ Ом·м}$$

Опір Rв, розповсюдженню струму в землі від одного вертикального заземлювача:

$$R_B = \frac{\rho_{рз}}{2\pi \cdot l}\left(\ln\frac{2 \cdot l}{d} + 0.5\ln\frac{4t + l}{4t - l}\right)$$

де:

l – довжина заземлювача (l = 2.5 м);

d = 0.05 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.5 + 2.5/2 = 1.75 \text{ м}.$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 2.5}\left(\ln\left(\frac{2 \cdot 2.5}{0.05}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.75 + 2.5}{4 \cdot 1.75 - 2.5}\right)\right) = 34.325 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{34.325}{10} = 3.4 \approx 4 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 2 м. Таким чином коефіцієнт використання вертикального заземлювача за довідковими даними дорівнює $\eta_В = 0,6$.

Кількість вертикальних заземлювачів з урахуванням коефіцієнту використання $\eta_В$ приблизно складає

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{34.325}{10 \cdot 0.6} = 5.72 \approx 6 \text{ шт.}$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 2 \cdot 6 = 12 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.5 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln\frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 12} \cdot \ln\frac{2 \cdot 12^2}{0.04 \cdot 0.5} = 13.976 \text{ Ом}$$

де b = 0.04 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{34.325 \cdot 13.976}{6 \cdot 13.976 \cdot 0.6 + 34.325 \cdot 0.34} = 7.65 \text{ Ом}$$

де $\eta_Г$ – коефіцієнт використання горизонтального заземлювача ($\eta_Г = 0.34$).

Маємо 7.65 Ом < 10 Ом (за потужності генераторів та трансформаторів 100 кВт і менше), отже нормативне обмеження $R_з < R_{з.норм}$ виконується.

# 9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи мереж SAN на основі стандарту Fibre Channel Gen 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 мереж SAN на основі стандарту Fibre Channel Gen 6.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем мереж SAN на основі стандарту Fibre Channel Gen 6.

– Досліджена система мереж SAN на основі стандарту Fibre Channel Gen 6.

– На основі отриманих результатів досліджень створена програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання мереж SAN на основі стандарту Fibre Channel Gen 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 Python. 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 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм AES.

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.

Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Барабаш А.І. Дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 6 // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

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

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

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

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

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

7. Kuznetsov, O., Kryvinska, N., Ilchenko, O., Smirnova, T., Ulianovska, Y. «Comparative Analysis of Cryptocurrency Trading Platforms Using the Analytic Hierarchy Process». *CEUR Workshop Proceedings*, 2023, 3628, pp. 106-115.

8. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

9. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

10. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». *Lecture Notes on Data Engineering and Communications Technologies*, 2023, 178, pp. 208–223.

11. Smirnova, T., Gnatyuk, S., Yudin, O., Sydorenko, V., Polozhentsev, A., «The Model for Calculating the Quantitative Criteria for Assessing the Security Level of Information and Telecommunication Systems». CEUR Workshop Proceedings Volume 3156, 2022, Pages 390-399.

12. Smirnova T., Gnatyuk S., Berdibayev R., Avkurova Zh., Iavich M. «Cloud-Based Cyber Incidents Response System and Software Tools». *Communications in Computer and Information Science*, 2021, vol 1486. Springer, Cham. pp 169-184.

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

14. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». *Journal of theoretical and applied information technology* Vol.98. No 21, 2020, P. 3334-3346.

15. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». *2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT),* Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

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

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

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

19. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», *CEUR Workshop Proceedings* Volume 2608, 2020, Pages 633-645.

20. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». *International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019*; Odessa; Ukraine; 9-13 September 2019. P.22-28.

21. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

22. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

23. Smirnov, O., Odarchenko, R., Abakumova, A., Usik, P., Kundyz, M., «QoE optimization technique for media delivery in 5G networks». *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T),* Kyiv, Ukraine, 8 – 11 October 2019. P.597-601.

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

25. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation

Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

26. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 353-358.

27. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», *2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS)*, Kyiv, Ukraine April 17-19, 2019 P. 347-352.

28. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», *CEUR Workshop Proceedings* Volume 2353, *CEUR Workshop Proceedings* 2019, Pages 618-629.

29. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», *Telecommunications and Radio Engineering*. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

30. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

31. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

32. Смірнова Т.В., Гнатюк С.О., Сидоренко В.М., Юдін О.Ю., Сидоренко С.Ю., «Модель визначення критичності галузевих інформаційно-телекомунікаційних систем». *Проблеми інформатизації та управління,* № 2(70). 2022. С. 28-37.

33. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Смірнов С.А., Поліщук Л.І., «Дослідження стійкості до диференціального криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку,* 2022, № 3(69). С. 93-98.

34. Смірнов О.А., Смірнова Т.В., Якименко Н.М., Поліщук Л.І., Смірнов С.А. «Дослідження статистичної стійкості та швидкісних характеристик запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Вісник Хмельницького національного університету. Серія: «Технічні науки»,* № 2 (307). С. 46-52. 2022.

35. Смірнов О.А., Смірнова Т.В., Константинова Л.В., Смірнов С.А., Якименко Н.М., «Дослідження стійкості до лінійного криптоаналізу запропонованої функції гешування удосконаленого модуля криптографічного захисту в інформаційно-комунікаційних системах» *Системи управління, навігації та зв'язку*, 2022, № 1(67). С. 84-89.

36. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». *Сучасні інформаційні системи*. 2021. Т. 5, № 4. С. 79-95

37. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

38. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», *Кібербезпека: освіта, наука, техніка*. № 3(7). С. 43-62. 2020.

39. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в

комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

40. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у *Кібербезпека та інформаційні технології: монографія.* – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

41. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». *Центральноукраїнський науковий вісник. Технічні науки.* № 2(33). с. 161-172, 2019.

42. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

43. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

44. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

45. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

46. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу

відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

47. Смірнов О.А., Смірнов С.А., Поліщук Л.І., Смірнова Т.В., Конопліцька-Слободенюк О.К. Метод формування антивірусного захисту даних з використанням безпечної маршрутизації метаданих. Кібербезпека: освіта, наука, техніка. – Том 3 № 3. – Київ: КУ ім. Бориса Грінченка. – 2019. – С. 63-87.

48. Смірнов О.А., Гнатюк С.О., Кавун С.В., Терейковський І.А., Жмурко Т.О., Смірнов С.А., Коваленко А.С. Основи безпеки в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2018. – 177 с.

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

50. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". – Випуск 5 (142). – Х.: ХУПС – 2016. – С. 148-152.

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

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

Додаток А
(обов'язковий)

**Технічне завдання**

**Зміст**

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

## 1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи мереж SAN на основі стандарту Fibre Channel Gen 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. № 19-13 від 07.08.2024 року).

## 3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи мереж SAN на основі стандарту Fibre Channel Gen 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 мереж SAN на основі стандарту Fibre Channel Gen 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, працювати в ОС Windows 10/11 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 10/11.

#### 5.8.1 Обладнання

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

#### 5.8.2 Мова програмування

Середовище Python.

### 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 маркетингове та економічне обґрунтування ІТ-проєкту з урахуванням цін на 3 вересня 2024 року.

## 8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз санітарно-гігієнічних умов праці на робочому місці програміста.

## 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 – 99 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 Маркетингове та економічне обґрунтування ІТ-проєкту.

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 02.12.2024 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 16.12.2024 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 мереж SAN на основі стандарту Fibre Channel Gen 6***



Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 26

Літера: РП

Кропивницький – 2024 року

**Основна програма**

```
import random
import time

class FibreChannelNode:
    def __init__(self, node_id, role):
        self.node_id = node_id
        self.role = role
# 'host' or 'storage'

    def send_data(self, data, target):
print(f"[Node {self.node_id}] Sending data to {target.node_id}: {data}")
        time.sleep(random.uniform(0.1, 0.5))
# Симуляція затримки у передачі даних
        target.receive_data(data, self)

    def receive_data(self, data, source):
print(f"[Node {self.node_id}] Received data from {source.node_id}: {data}")

class FibreChannelSwitch:
    def __init__(self, switch_id):
        self.switch_id = switch_id
        self.connected_nodes = []

    def connect_node(self, node):
        self.connected_nodes.append(node)
print(f"[Switch {self.switch_id}] Node {node.node_id} connected.")

    def transfer_data(self, data, source_node, target_node_id):
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
print(f"[Switch {self.switch_id}] Routing data from {source_node.node_id} to
{target_node.node_id}")
            source_node.send_data(data, target_node)
        else:
print(f"[Switch {self.switch_id}] Target node {target_node_id} not found.")

# Створення вузлів
host = FibreChannelNode(node_id=1, role='host')
storage = FibreChannelNode(node_id=2, role='storage')

# Створення комутатора
fc_switch = FibreChannelSwitch(switch_id=101)

# Підключення вузлів до комутатора
fc_switch.connect_node(host)
fc_switch.connect_node(storage)

# Передача даних
fc_switch.transfer_data(data="Backup file", source_node=host, target_node_id=2)

class FibreChannelZone:
    def __init__(self, zone_id):
        self.zone_id = zone_id
        self.members = []

    def add_member(self, node):
        self.members.append(node)
print(f"[Zone {self.zone_id}] Node {node.node_id} added to the zone.")

    def is_member(self, node):
        return node in self.members

class FibreChannelSwitchWithZones(FibreChannelSwitch):
    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.zones = {}
```

```
    def create_zone(self, zone_id):
        self.zones[zone_id] = FibreChannelZone(zone_id)
print(f"[Switch {self.switch_id}] Zone {zone_id} created.")

    def add_node_to_zone(self, zone_id, node):
        if zone_id in self.zones:
            self.zones[zone_id].add_member(node)
        else:
print(f"[Switch {self.switch_id}] Zone {zone_id} does not exist.")

    def transfer_data_in_zone(self, data, source_node, target_node_id, zone_id):
        if zone_id not in self.zones:
print(f"[Switch {self.switch_id}] Zone {zone_id} does not exist.")
            return

if self.zones[zone_id].is_member(source_node):
target_node = next((node for node in self.zones[zone_id].members
if node.node_id == target_node_id), None)

if target_node:
print(f"[Switch {self.switch_id}] Transferring data within Zone {zone_id} from
{source_node.node_id} to {target_node.node_id}")
                source_node.send_data(data, target_node)

else:
print(f"[Switch {self.switch_id}] Target node {target_node_id} not found in Zone
{zone_id}")

else:

print(f"[Switch {self.switch_id}] Source node {source_node.node_id} not
authorized in Zone {zone_id}")
```

**Основна архітектура SAN**

```
import random
import time

class FibreChannelNode:
    """Представляє вузол у мережі SAN: хост або сховище."""

    def __init__(self, node_id, role):
        self.node_id = node_id
        self.role = role
# 'host' або 'storage'
        self.data_received = []

    def send_data(self, data, target):
print(f"[Node {self.node_id}] Відправлення даних до {target.node_id}: {data}")
        time.sleep(random.uniform(0.1, 0.5))
# Симуляція затримки у передачі даних
        target.receive_data(data, self)

    def receive_data(self, data, source):
        self.data_received.append((data, source.node_id))
print(f"[Node {self.node_id}] Отримано дані від {source.node_id}: {data}")

    def get_received_data(self):
        """Повертає всі отримані дані для цього вузла."""
        return self.data_received

class FibreChannelSwitch:
    """Комутатор Fibre Channel, що з'єднує вузли в мережі SAN."""

    def __init__(self, switch_id):
        self.switch_id = switch_id
        self.connected_nodes = []
        self.data_transferred = 0  # Обсяг переданих даних у байтах
        self.latency = random.uniform(0.1, 0.5)
# Симуляція середньої затримки

    def connect_node(self, node):
        self.connected_nodes.append(node)
print(f"[Switch {self.switch_id}] Вузол {node.node_id} підключений.")

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних між вузлами через комутатор."""
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
            data_size = len(data.encode('utf-8'))
# Розмір даних у байтах
            self.data_transferred += data_size
print(f"[Switch {self.switch_id}] Маршрутизація {data_size} байт від
{source_node.node_id} до {target_node.node_id}")
            time.sleep(self.latency)
# Симуляція затримки
            source_node.send_data(data, target_node)
        else:
print(f"[Switch {self.switch_id}] Вузол з ідентифікатором {target_node_id} не
знайдений.")

    def get_transfer_statistics(self):
        """Повертає статистику передачі даних."""
print(f"[Switch {self.switch_id}] Загальний обсяг переданих даних:
{self.data_transferred} байт")
```

**Додавання зонування (Zoning) для сегментації мережі**

```
class FibreChannelZone:
    """Представляє зону у мережі SAN, для ізоляції трафіку між вузлами."""

    def __init__(self, zone_id):
        self.zone_id = zone_id
        self.members = []

    def add_member(self, node):
        self.members.append(node)
print(f"[Zone {self.zone_id}] Вузол {node.node_id} доданий до зони.")

    def is_member(self, node):
        return node in self.members

class FibreChannelSwitchWithZones(FibreChannelSwitch):
    """Комутатор із підтримкою зонування, для ізоляції трафіку в SAN."""

    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.zones = {}

    def create_zone(self, zone_id):
        self.zones[zone_id] = FibreChannelZone(zone_id)
print(f"[Switch {self.switch_id}] Зона {zone_id} створена.")

    def add_node_to_zone(self, zone_id, node):
        if zone_id in self.zones:
            self.zones[zone_id].add_member(node)
        else:
print(f"[Switch {self.switch_id}] Зона {zone_id} не існує.")

    def transfer_data_in_zone(self, data, source_node, target_node_id, zone_id):
        """Передача даних між вузлами в межах зони."""
        if zone_id not in self.zones:
print(f"[Switch {self.switch_id}] Зона {zone_id} не існує.")
            return

        if self.zones[zone_id].is_member(source_node):
            target_node = next((node for node in self.zones[zone_id].members if
node.node_id == target_node_id), None)
            if target_node:
print(f"[Switch {self.switch_id}] Маршрутизація даних у межах зони {zone_id} від
{source_node.node_id} до {target_node.node_id}")
                self.transfer_data(data, source_node, target_node.node_id)
            else:
print(f"[Switch {self.switch_id}] Вузол {target_node_id} не знайдений у зоні
{zone_id}")
        else:
print(f"[Switch {self.switch_id}] Вузол {source_node.node_id} не має доступу до
зони {zone_id}")
```

**Додавання резервування каналів передачі даних**

```
class FibreChannelSwitchWithRedundancy(FibreChannelSwitchWithZones):
    """Комутатор з підтримкою резервування каналів передачі даних."""

    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.redundant_paths = {}
# Словник для зберігання інформації про резервні канали

    def add_redundant_path(self, source_node_id, target_node_id):
        """Додає резервний шлях для вузлів."""
        self.redundant_paths[(source_node_id, target_node_id)] = True
print(f"[Switch {self.switch_id}] Додано резервний канал між вузлами
{source_node_id} і {target_node_id}")

    def transfer_data_with_redundancy(self, data, source_node, target_node_id):
        """Передача даних з використанням резервного каналу, якщо основний канал
відмовляє."""
        if (source_node.node_id, target_node_id) in self.redundant_paths:
            if random.choice([True, False]):
# Симуляція відмови основного каналу
print(f"[Switch {self.switch_id}] Основний канал відмовив. Використання
резервного каналу між вузлами {source_node.node_id} і {target_node_id}")
            else:
print(f"[Switch {self.switch_id}] Використання основного каналу між вузлами
{source_node.node_id} і {target_node_id}")
            self.transfer_data(data, source_node, target_node_id)
        else:
print(f"[Switch {self.switch_id}] Резервний канал між вузлами
{source_node.node_id} і {target_node_id} не знайдено")
```

**Моніторинг продуктивності та безперервність обслуговування**

```
class FibreChannelSwitchWithMonitoring(FibreChannelSwitchWithRedundancy):
"""Комутатор із розширеним моніторингом продуктивності."""

    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.transferred_data_per_node = {}
# Словник для зберігання статистики передачі даних по вузлах

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних між вузлами з оновленням статистики."""
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
            data_size = len(data.encode('utf-8'))
            self.data_transferred += data_size

# Оновлення статистики
if source_node.node_id not in self.transferred_data_per_node:
                self.transferred_data_per_node[source_node.node_id] = 0
            self.transferred_data_per_node[source_node.node_id] += data_size

print(f"[Switch {self.switch_id}] Передано {data_size} байт від вузла
{source_node.node_id} до вузла {target_node.node_id}")
            time.sleep(self.latency)
            source_node.send_data(data, target_node)
        else:
print(f"[Switch {self.switch_id}] Вузол {target_node_id} не знайдено")
    def get_transfer_statistics(self):
"""Повертає загальну статистику передачі даних і по кожному вузлу."""
print(f"[Switch {self.switch_id}] Загальний обсяг переданих даних:
{self.data_transferred} байт")
        for node_id, data_size in self.transferred_data_per_node.items():
print(f"Вузол {node_id}: передано {data_size} байт")
```

**Логування подій та помилок**

```
import logging

class SANLogger:
    """Система логування подій у SAN."""

    def __init__(self, log_file='san_events.log'):
        logging.basicConfig(filename=log_file, level=logging.INFO)

    def log_event(self, message):
        logging.info(f"EVENT: {message}")

    def log_error(self, message):
        logging.error(f"ERROR: {message}")

class FibreChannelSwitchWithLogging(FibreChannelSwitchWithMonitoring):
    """Комутатор із підтримкою логування подій та помилок."""

    def __init__(self, switch_id, logger):
        super().__init__(switch_id)
        self.logger = logger

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних із логуванням подій."""
        try:
            target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
            if target_node:
                data_size = len(data.encode('utf-8'))
                self.data_transferred += data_size
                self.logger.log_event(f"Передано {data_size} байт від
{source_node.node_id} до {target_node.node_id}")
                time.sleep(self.latency)
                source_node.send_data(data, target_node)
            else:
                raise ValueError(f"Вузол з ID {target_node_id} не знайдено.")
        except Exception as e:
            self.logger.log_error(f"Помилка передачі даних: {str(e)}")
print(f"Помилка: {str(e)}")

# Приклад використання системи логування
logger = SANLogger(log_file="san_log.log")
fc_switch = FibreChannelSwitchWithLogging(switch_id=101, logger=logger)

# Підключення вузлів та передача даних з логуванням
fc_switch.connect_node(host1)
fc_switch.connect_node(storage1)
fc_switch.transfer_data(data="Логування події", source_node=host1,
target_node_id=2)
```

```
class FibreChannelNodeWithBandwidth(FibreChannelNode):
    """Вузол із підтримкою обмеження пропускної здатності."""

    def __init__(self, node_id, role, max_bandwidth):
        super().__init__(node_id, role)
        self.max_bandwidth = max_bandwidth  # Максимальна пропускна здатність у
байтах/сек
        self.data_sent_in_period = 0

    def can_send(self, data_size):
        """Перевірка, чи можна передати дані в межах поточної пропускної
здатності."""
        if self.data_sent_in_period + data_size <= self.max_bandwidth:
            return True
        else:
print(f"[Node {self.node_id}] Перевищення пропускної здатності. Затримка
передачі.")
            return False

    def reset_bandwidth(self):
        """Скидання лічильника пропускної здатності (симуляція нового
періоду)."""
        self.data_sent_in_period = 0

class FibreChannelSwitchWithBandwidthManagement(FibreChannelSwitch):
    """Комутатор із підтримкою управління пропускною здатністю."""

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних із перевіркою пропускної здатності."""
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
            data_size = len(data.encode('utf-8'))
            if source_node.can_send(data_size):
                self.data_transferred += data_size
                source_node.data_sent_in_period += data_size
print(f"[Switch {self.switch_id}] Передано {data_size} байт від
{source_node.node_id} до {target_node.node_id}")
                time.sleep(self.latency)
                source_node.send_data(data, target_node)
            else:
print(f"[Switch {self.switch_id}] Передача від {source_node.node_id} до
{target_node.node_id} затримана.")
        else:
print(f"[Switch {self.switch_id}] Вузол з ID {target_node_id} не знайдено.")
```

**Додавання симуляції помилок передачі**

```
class
FibreChannelSwitchWithErrorSimulation(FibreChannelSwitchWithBandwidthManagement)
:
    """Комутатор із підтримкою симуляції помилок передачі даних."""

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних з можливістю втрати або помилки в передачі."""
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
            data_size = len(data.encode('utf-8'))
            if random.random() > 0.95:  # 5% імовірність втрати даних
print(f"[Switch {self.switch_id}] Втрачено пакет даних при передачі від
{source_node.node_id} до {target_node.node_id}")
            elif random.random() > 0.90:  # 10% імовірність помилки передачі
print(f"[Switch {self.switch_id}] Помилка передачі даних від
{source_node.node_id} до {target_node.node_id}")
            else:
                super().transfer_data(data, source_node, target_node_id)
        else:
print(f"[Switch {self.switch_id}] Вузол з ID {target_node_id} не знайдено.")
```

**Забезпечення балансування навантаження**

```
class FibreChannelSwitchWithLoadBalancing(FibreChannelSwitch):
    """Комутатор із підтримкою балансування навантаження."""

    def __init__(self, switch_id):
        super().__init__(switch_id)
        self.path_load = {}  # Словник для зберігання навантаження по кожному
каналу

    def register_path(self, source_node_id, target_node_id):
        """Реєстрація нового шляху з ініціалізацією навантаження."""
        self.path_load[(source_node_id, target_node_id)] = 0

    def transfer_data(self, data, source_node, target_node_id):
        """Передача даних із балансуванням навантаження."""
        target_node = next((node for node in self.connected_nodes if
node.node_id == target_node_id), None)
        if target_node:
            data_size = len(data.encode('utf-8'))
            current_load = self.path_load.get((source_node.node_id,
target_node.node_id), 0)
            if current_load < 1000:  # Ліміт на навантаження по кожному шляху
                self.path_load[(source_node.node_id, target_node.node_id)] +=
data_size
print(f"[Switch {self.switch_id}] Передача даних від {source_node.node_id} до
{target_node.node_id} через основний канал.")
                super().transfer_data(data, source_node, target_node_id)
            else:
                # Знайти інший шлях, якщо є
                alt_path = self.find_alternate_path(source_node, target_node_id)
                if alt_path:
print(f"[Switch {self.switch_id}] Перенаправлення трафіку на альтернативний шлях
від {source_node.node_id} до {target_node.node_id}")
                    super().transfer_data(data, source_node, alt_path)
                else:
print(f"[Switch {self.switch_id}] Неможливо перенаправити трафік, всі шляхи
перевантажені.")
        else:
print(f"[Switch {self.switch_id}] Вузол з ID {target_node_id} не знайдено.")

    def find_alternate_path(self, source_node, target_node_id):
"""Пошук альтернативного шляху для балансування навантаження."""
for (src, tgt), load in self.path_load.items():
            if src == source_node.node_id and tgt == target_node_id and load <
1000:
                return tgt
        return None
```

**Інтерфейс управління системою SAN**

```
class SANManagementConsole:
    """Простий текстовий інтерфейс для управління SAN."""

    def __init__(self, switch):
        self.switch = switch

    def display_menu(self):
 print("\n=== SAN Management Console ===")
 print("1. Підключити новий вузол")
 print("2. Створити нову зону")
 print("3. Додати вузол до зони")
 print("4. Передача даних між вузлами")
 print("5. Отримати статистику передачі")
 print("6. Вийти")

    def connect_node(self):
        node_id = int(input("Введіть ID нового вузла: "))
        role = input("Введіть роль (host/storage): ")
        node = FibreChannelNode(node_id, role)
        self.switch.connect_node(node)

    def create_zone(self):
        zone_id = int(input("Введіть ID нової зони: "))
        self.switch.create_zone(zone_id)

    def add_node_to_zone(self):
        zone_id = int(input("Введіть ID зони: "))
        node_id = int(input("Введіть ID вузла для додавання в зону: "))
        node = next((node for node in self.switch.connected_nodes if
node.node_id == node_id), None)
        if node:
            self.switch.add_node_to_zone(zone_id, node)
        else:
print(f"Вузол {node_id} не знайдений")

    def transfer_data(self):
        source_id = int(input("Введіть ID відправника: "))
        target_id = int(input("Введіть ID отримувача: "))
        data = input("Введіть дані для передачі: ")
        source_node = next((node for node in self.switch.connected_nodes if
node.node_id == source_id), None)
        if source_node:
            self.switch.transfer_data(data, source_node, target_id)
        else:
print(f"Вузол {source_id} не знайдений")

    def run(self):
        while True:
            self.display_menu()
            choice = int(input("Оберіть опцію: "))
            if choice == 1:
                self.connect_node()
            elif choice == 2:
                self.create_zone()
            elif choice == 3:
                self.add_node_to_zone()
            elif choice == 4:
                self.transfer_data()
            elif choice == 5:
                self.switch.get_transfer_statistics()
            elif choice == 6:
print("Вихід з консолі.")
                break
            else:
print("Неправильний вибір.")
```

```
#!/usr/bin/env python3

from __future__ import print_function

import argparse
import atexit
import ctypes
import errno
import fnmatch
import os
import shutil
import site
import ssl
import subprocess
import sys
import tempfile


APPVEYOR = bool(os.environ.get('APPVEYOR'))
PYTHON = sys.executable if APPVEYOR else os.getenv('PYTHON', sys.executable)
PY3 = sys.version_info[0] >= 3
PYTEST_ARGS = "-v -s --tb=short"
if PY3:
    PYTEST_ARGS += "-o "
HERE = os.path.abspath(os.path.dirname(__file__))
ROOT_DIR = os.path.realpath(os.path.join(HERE, "..", ".."))
PYPY = '__pypy__' in sys.builtin_module_names
WINDOWS = os.name == "nt"
if PY3:
    GET_PIP_URL = "https://bootstrap.pypa.io/get-pip.py"
else:
    GET_PIP_URL = "https://bootstrap.pypa.io/pip/2.7/get-pip.py"

sys.path.insert(0, ROOT_DIR)  # so that we can import setup.py

import setup  # NOQA

TEST_DEPS = setup.TEST_DEPS
DEV_DEPS = setup.DEV_DEPS

_cmds = {}
if PY3:
    basestring = str

GREEN = 2
LIGHTBLUE = 3
YELLOW = 6
RED = 4
DEFAULT_COLOR = 7


# ====================================================================
# utils
# ====================================================================


def safe_print(text, file=sys.stdout):
    """Prints a (unicode) string to the console, encoded depending on
    the stdout/file encoding (eg. cp437 on Windows). This is to avoid
    encoding errors in case of funky path names.
    Works with Python 2 and 3.
    """
    if not isinstance(text, basestring):
        return print(text, file=file)
    try:
        file.write(text)
```

```
        except UnicodeEncodeError:
            bytes_string = text.encode(file.encoding, 'backslashreplace')
            if hasattr(file, 'buffer'):
                file.buffer.write(bytes_string)
            else:
                text = bytes_string.decode(file.encoding, 'strict')
                file.write(text)
        file.write("\n")


    def stderr_handle():
        GetStdHandle = ctypes.windll.Kernel32.GetStdHandle
        STD_ERROR_HANDLE_ID = ctypes.c_ulong(0xFFFFFFF4)
        GetStdHandle.restype = ctypes.c_ulong
        handle = GetStdHandle(STD_ERROR_HANDLE_ID)
        atexit.register(ctypes.windll.Kernel32.CloseHandle, handle)
        return handle


    def win_colorprint(s, color=LIGHTBLUE):
        if not WINDOWS:
            return print(s)
        color += 8  # bold
        handle = stderr_handle()
        SetConsoleTextAttribute = ctypes.windll.Kernel32.SetConsoleTextAttribute
        SetConsoleTextAttribute(handle, color)
        try:
            print(s)
        finally:
            SetConsoleTextAttribute(handle, DEFAULT_COLOR)


    def sh(cmd, nolog=False):
        if not nolog:
            safe_print("cmd: " + cmd)
        p = subprocess.Popen(  # noqa S602
            cmd, shell=True, env=os.environ, cwd=os.getcwd()
        )
        p.communicate()
        if p.returncode != 0:
            sys.exit(p.returncode)


    def rm(pattern, directory=False):
        """Recursively remove a file or dir by pattern."""

        def safe_remove(path):
            try:
                os.remove(path)
            except OSError as err:
                if err.errno != errno.ENOENT:
                    raise
            else:
                safe_print("rm %s" % path)

        def safe_rmtree(path):
            def onerror(fun, path, excinfo):
                exc = excinfo[1]
                if exc.errno != errno.ENOENT:
                    raise  # noqa: PLE0704

            existed = os.path.isdir(path)
            shutil.rmtree(path, onerror=onerror)
            if existed:
                safe_print("rmdir -f %s" % path)

        if "*" not in pattern:
            if directory:
                safe_rmtree(pattern)
```

```
            else:
                safe_remove(pattern)
            return

        for root, dirs, files in os.walk('.'):
            root = os.path.normpath(root)
            if root.startswith('.git/'):
                continue
            found = fnmatch.filter(dirs if directory else files, pattern)
            for name in found:
                path = os.path.join(root, name)
                if directory:
                    safe_print("rmdir -f %s" % path)
                    safe_rmtree(path)
                else:
                    safe_print("rm %s" % path)
                    safe_remove(path)


    def safe_remove(path):
        try:
            os.remove(path)
        except OSError as err:
            if err.errno != errno.ENOENT:
                raise
        else:
            safe_print("rm %s" % path)


    def safe_rmtree(path):
        def onerror(fun, path, excinfo):
            exc = excinfo[1]
            if exc.errno != errno.ENOENT:
                raise  # noqa: PLE0704

        existed = os.path.isdir(path)
        shutil.rmtree(path, onerror=onerror)
        if existed:
            safe_print("rmdir -f %s" % path)


    def recursive_rm(*patterns):
        """Recursively remove a file or matching a list of patterns."""
        for root, dirs, files in os.walk('.'):
            root = os.path.normpath(root)
            if root.startswith('.git/'):
                continue
            for file in files:
                for pattern in patterns:
                    if fnmatch.fnmatch(file, pattern):
                        safe_remove(os.path.join(root, file))
            for dir in dirs:
                for pattern in patterns:
                    if fnmatch.fnmatch(dir, pattern):
                        safe_rmtree(os.path.join(root, dir))


    # =====================================================================
    # commands
    # =====================================================================


    def build():
        """Build / compile."""
        # Make sure setuptools is installed (needed for 'develop' /
        # edit mode).
        sh('%s -c "import setuptools"' % PYTHON)

        # "build_ext -i" copies compiled *.pyd files in ./psutil directory in
```

```
    # order to allow "import psutil" when using the interactive interpreter
    # from within psutil root directory.
    cmd = [PYTHON, "setup.py", "build_ext", "-i"]
    if sys.version_info[:2] >= (3, 6) and (os.cpu_count() or 1) > 1:
        cmd += ['--parallel', str(os.cpu_count())]
    # Print coloured warnings in real time.
    p = subprocess.Popen(cmd, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
    try:
        for line in iter(p.stdout.readline, b''):
            if PY3:
                line = line.decode()
            line = line.strip()
            if 'warning' in line:
                win_colorprint(line, YELLOW)
            elif 'error' in line:
                win_colorprint(line, RED)
            else:
                print(line)
        # retcode = p.poll()
        p.communicate()
        if p.returncode:
            win_colorprint("failure", RED)
            sys.exit(p.returncode)
    finally:
        p.terminate()
        p.wait()

    # Make sure it actually worked.
    sh('%s -c "import psutil"' % PYTHON)
    win_colorprint("build + import successful", GREEN)


def wheel():
    """Create wheel file."""
    build()
    sh("%s setup.py bdist_wheel" % PYTHON)


def upload_wheels():
    """Upload wheel files on PyPI."""
    build()
    sh("%s -m twine upload dist/*.whl" % PYTHON)


def install_pip():
    """Install pip."""
    try:
        sh('%s -c "import pip"' % PYTHON)
    except SystemExit:
        if PY3:
            from urllib.request import urlopen
        else:
            from urllib2 import urlopen

        if hasattr(ssl, '_create_unverified_context'):
            ctx = ssl._create_unverified_context()
        else:
            ctx = None
        kw = dict(context=ctx) if ctx else {}
        safe_print("downloading %s" % GET_PIP_URL)
        req = urlopen(GET_PIP_URL, **kw)
        data = req.read()

        tfile = os.path.join(tempfile.gettempdir(), 'get-pip.py')
        with open(tfile, 'wb') as f:
            f.write(data)

        try:
            sh('%s %s --user' % (PYTHON, tfile))
```

```
        finally:
            os.remove(tfile)


def install():
    """Install in develop / edit mode."""
    build()
    sh("%s setup.py develop" % PYTHON)


def uninstall():
    """Uninstall psutil."""
    # Uninstalling psutil on Windows seems to be tricky.
    # On "import psutil" tests may import a psutil version living in
    # C:\PythonXY\Lib\site-packages which is not what we want, so
    # we try both "pip uninstall psutil" and manually remove stuff
    # from site-packages.
    clean()
    install_pip()
    here = os.getcwd()
    try:
        os.chdir('C:\\')
        while True:
            try:
                import psutil  # NOQA
            except ImportError:
                break
            else:
                sh("%s -m pip uninstall -y psutil" % PYTHON)
    finally:
        os.chdir(here)

    for dir in site.getsitepackages():
        for name in os.listdir(dir):
            if name.startswith('psutil'):
                rm(os.path.join(dir, name))
            elif name == 'easy-install.pth':
                # easy_install can add a line (installation path) into
                # easy-install.pth; that line alters sys.path.
                path = os.path.join(dir, name)
                with open(path) as f:
                    lines = f.readlines()
                    hasit = False
                    for line in lines:
                        if 'psutil' in line:
                            hasit = True
                            break
                if hasit:
                    with open(path, "w") as f:
                        for line in lines:
                            if 'psutil' not in line:
                                f.write(line)
                            else:
                                print("removed line %r from %r" % (line, path))


def clean():
    """Deletes dev files."""
    recursive_rm(
        "$testfn*",
        "*.bak",
        "*.core",
        "*.egg-info",
        "*.orig",
        "*.pyc",
        "*.pyd",
        "*.pyo",
        "*.rej",
        "*.so",
```

```
            "*.~",
            "*__pycache__",
            ".coverage",
            ".failed-tests.txt",
        )
        safe_rmtree("build")
        safe_rmtree(".coverage")
        safe_rmtree("dist")
        safe_rmtree("docs/_build")
        safe_rmtree("htmlcov")
        safe_rmtree("tmp")


    def install_pydeps_test():
        """Install useful deps."""
        install_pip()
        install_git_hooks()
        sh("%s -m pip install -U %s" % (PYTHON, " ".join(TEST_DEPS)))


    def install_pydeps_dev():
        """Install useful deps."""
        install_pip()
        install_git_hooks()
        sh("%s -m pip install -U %s" % (PYTHON, " ".join(DEV_DEPS)))


    def test(args=""):
        """Run tests."""
        build()
        sh("%s -m pytest %s %s" % (PYTHON, PYTEST_ARGS, args))


    def coverage():
        """Run coverage tests."""
        # Note: coverage options are controlled by .coveragerc file
        build()
        sh("%s -m coverage run -m pytest %s" % (PYTHON, PYTEST_ARGS))
        sh("%s -m coverage report" % PYTHON)
        sh("%s -m coverage html" % PYTHON)
        sh("%s -m webbrowser -t htmlcov/index.html" % PYTHON)


    def test_process():
        """Run process tests."""
        build()
        sh("%s psutil\\tests\\test_process.py" % PYTHON)


    def test_process_all():
        """Run process all tests."""
        build()
        sh("%s psutil\\tests\\test_process_all.py" % PYTHON)


    def test_system():
        """Run system tests."""
        build()
        sh("%s psutil\\tests\\test_system.py" % PYTHON)


    def test_platform():
        """Run windows only tests."""
        build()
        sh("%s psutil\\tests\\test_windows.py" % PYTHON)


    def test_misc():
        """Run misc tests."""
```

```
    build()
    sh("%s psutil\\tests\\test_misc.py" % PYTHON)


def test_unicode():
    """Run unicode tests."""
    build()
    sh("%s psutil\\tests\\test_unicode.py" % PYTHON)


def test_connections():
    """Run connections tests."""
    build()
    sh("%s psutil\\tests\\test_connections.py" % PYTHON)


def test_contracts():
    """Run contracts tests."""
    build()
    sh("%s psutil\\tests\\test_contracts.py" % PYTHON)


def test_testutils():
    """Run test utilities tests."""
    build()
    sh("%s psutil\\tests\\test_testutils.py" % PYTHON)


def test_by_name(name):
    """Run test by name."""
    build()
    test(name)


def test_last_failed():
    """Re-run tests which failed on last run."""
    build()
    test("--last-failed")


def test_memleaks():
    """Run memory leaks tests."""
    build()
    sh("%s psutil\\tests\\test_memleaks.py" % PYTHON)


def install_git_hooks():
    """Install GIT pre-commit hook."""
    if os.path.isdir('.git'):
        src = os.path.join(
            ROOT_DIR, "scripts", "internal", "git_pre_commit.py"
        )
        dst = os.path.realpath(
            os.path.join(ROOT_DIR, ".git", "hooks", "pre-commit")
        )
        with open(src) as s:
            with open(dst, "w") as d:
                d.write(s.read())


def bench_oneshot():
    """Benchmarks for oneshot() ctx manager (see #799)."""
    sh("%s -Wa scripts\\internal\\bench_oneshot.py" % PYTHON)


def bench_oneshot_2():
    """Same as above but using perf module (supposed to be more precise)."""
    sh("%s -Wa scripts\\internal\\bench_oneshot_2.py" % PYTHON)
```

```
def print_access_denied():
    """Print AD exceptions raised by all Process methods."""
    build()
    sh("%s -Wa scripts\\internal\\print_access_denied.py" % PYTHON)


def print_api_speed():
    """Benchmark all API calls."""
    build()
    sh("%s -Wa scripts\\internal\\print_api_speed.py" % PYTHON)


def print_sysinfo():
    """Print system info."""
    build()
    from psutil.tests import print_sysinfo

    print_sysinfo()


def download_appveyor_wheels():
    """Download appveyor wheels."""
    sh(
        "%s -Wa scripts\\internal\\download_wheels_appveyor.py "
        "--user giampaolo --project psutil" % PYTHON
    )


def generate_manifest():
    """Generate MANIFEST.in file."""
    script = "scripts\\internal\\generate_manifest.py"
    out = subprocess.check_output([PYTHON, script], text=True)
    with open("MANIFEST.in", "w", newline="\n") as f:
        f.write(out)


def get_python(path):
    if not path:
        return sys.executable
    if os.path.isabs(path):
        return path
    # try to look for a python installation given a shortcut name
    path = path.replace('.', '')
    vers = (
        '27',
        '27-32',
        '27-64',
        '310-32',
        '310-64',
        '311-32',
        '311-64',
        '312-32',
        '312-64',
    )
    for v in vers:
        pypath = r'C:\\python%s\python.exe' % v
        if path in pypath and os.path.isfile(pypath):
            return pypath


def parse_args():
    parser = argparse.ArgumentParser()
    # option shared by all commands
    parser.add_argument('-p', '--python', help="use python executable path")
    sp = parser.add_subparsers(dest='command', title='targets')
    sp.add_parser('bench-oneshot', help="benchmarks for oneshot()")
    sp.add_parser('bench-oneshot_2', help="benchmarks for oneshot() (perf)")
    sp.add_parser('build', help="build")
```

```
        sp.add_parser('clean', help="deletes dev files")
        sp.add_parser('coverage', help="run coverage tests.")
        sp.add_parser('download-appveyor-wheels', help="download wheels.")
        sp.add_parser('generate-manifest', help="generate MANIFEST.in file")
        sp.add_parser('help', help="print this help")
        sp.add_parser('install', help="build + install in develop/edit mode")
        sp.add_parser('install-git-hooks', help="install GIT pre-commit hook")
        sp.add_parser('install-pip', help="install pip")
        sp.add_parser('install-pydeps-dev', help="install dev python deps")
        sp.add_parser('install-pydeps-test', help="install python test deps")
        sp.add_parser('print-access-denied', help="print AD exceptions")
        sp.add_parser('print-api-speed', help="benchmark all API calls")
        sp.add_parser('print-sysinfo', help="print system info")
        test = sp.add_parser('test', help="[ARG] run tests")
        test_by_name = sp.add_parser('test-by-name', help="<ARG> run test by name")
        sp.add_parser('test-connections', help="run connections tests")
        sp.add_parser('test-contracts', help="run contracts tests")
        sp.add_parser(
            'test-last-failed', help="re-run tests which failed on last run"
        )
        sp.add_parser('test-memleaks', help="run memory leaks tests")
        sp.add_parser('test-misc', help="run misc tests")
        sp.add_parser('test-platform', help="run windows only tests")
        sp.add_parser('test-process', help="run process tests")
        sp.add_parser('test-process-all', help="run process all tests")
        sp.add_parser('test-system', help="run system tests")
        sp.add_parser('test-unicode', help="run unicode tests")
        sp.add_parser('test-testutils', help="run test utils tests")
        sp.add_parser('uninstall', help="uninstall psutil")
        sp.add_parser('upload-wheels', help="upload wheel files on PyPI")
        sp.add_parser('wheel', help="create wheel file")

        for p in (test, test_by_name):
            p.add_argument('arg', type=str, nargs='?', default="", help="arg")

        args = parser.parse_args()

        if not args.command or args.command == 'help':
            parser.print_help(sys.stderr)
            sys.exit(1)

        return args


    def main():
        global PYTHON
        args = parse_args()
        # set python exe
        PYTHON = get_python(args.python)
        if not PYTHON:
            return sys.exit(
                "can't find any python installation matching %r" % args.python
            )
        os.putenv('PYTHON', PYTHON)
        win_colorprint("using " + PYTHON)
        fname = args.command.replace('-', '_')
        fun = getattr(sys.modules[__name__], fname)  # err if fun not defined
        funargs = []
        # mandatory args
        if args.command in ('test-by-name', 'test-script'):
            if not args.arg:
                sys.exit('command needs an argument')
            funargs = [args.arg]
        # optional args
        if args.command == 'test' and args.arg:
            funargs = [args.arg]
        fun(*funargs)
    if __name__ == '__main__':
        main()
```

```
import argparse
import datetime
import socket
import sys

import psutil
from psutil._common import bytes2human


ACCESS_DENIED = ''
NON_VERBOSE_ITERATIONS = 4
RLIMITS_MAP = {
    "RLIMIT_AS": "virtualmem",
    "RLIMIT_CORE": "coredumpsize",
    "RLIMIT_CPU": "cputime",
    "RLIMIT_DATA": "datasize",
    "RLIMIT_FSIZE": "filesize",
    "RLIMIT_MEMLOCK": "memlock",
    "RLIMIT_MSGQUEUE": "msgqueue",
    "RLIMIT_NICE": "nice",
    "RLIMIT_NOFILE": "openfiles",
    "RLIMIT_NPROC": "maxprocesses",
    "RLIMIT_NPTS": "pseudoterms",
    "RLIMIT_RSS": "rss",
    "RLIMIT_RTPRIO": "realtimeprio",
    "RLIMIT_RTTIME": "rtimesched",
    "RLIMIT_SBSIZE": "sockbufsize",
    "RLIMIT_SIGPENDING": "sigspending",
    "RLIMIT_STACK": "stack",
    "RLIMIT_SWAP": "swapuse",
}


def print_(a, b):
    if sys.stdout.isatty() and psutil.POSIX:
        fmt = '\x1b[1;32m%-13s\x1b[0m %s' % (a, b)
    else:
        fmt = '%-11s %s' % (a, b)
    print(fmt)


def str_ntuple(nt, convert_bytes=False):
    if nt == ACCESS_DENIED:
        return ""
    if not convert_bytes:
        return ", ".join(["%s=%s" % (x, getattr(nt, x)) for x in nt._fields])
    else:
        return ", ".join(
            ["%s=%s" % (x, bytes2human(getattr(nt, x))) for x in nt._fields]
        )


def run(pid, verbose=False):
    try:
        proc = psutil.Process(pid)
        pinfo = proc.as_dict(ad_value=ACCESS_DENIED)
    except psutil.NoSuchProcess as err:
        sys.exit(str(err))

    # collect other proc info
    with proc.oneshot():
        try:
            parent = proc.parent()
            parent = '(%s)' % parent.name() if parent else ''
        except psutil.Error:
            parent = ''
```

```
    try:
        pinfo['children'] = proc.children()
    except psutil.Error:
        pinfo['children'] = []
    if pinfo['create_time']:
        started = datetime.datetime.fromtimestamp(
            pinfo['create_time']
        ).strftime('%Y-%m-%d %H:%M')
    else:
        started = ACCESS_DENIED

# here we go
print_('pid', pinfo['pid'])
print_('name', pinfo['name'])
print_('parent', '%s %s' % (pinfo['ppid'], parent))
print_('exe', pinfo['exe'])
print_('cwd', pinfo['cwd'])
print_('cmdline', ' '.join(pinfo['cmdline']))
print_('started', started)

cpu_tot_time = datetime.timedelta(seconds=sum(pinfo['cpu_times']))
cpu_tot_time = "%s:%s.%s" % (
    cpu_tot_time.seconds // 60 % 60,
    str(cpu_tot_time.seconds % 60).zfill(2),
    str(cpu_tot_time.microseconds)[:2],
)
print_('cpu-tspent', cpu_tot_time)
print_('cpu-times', str_ntuple(pinfo['cpu_times']))
if hasattr(proc, "cpu_affinity"):
    print_("cpu-affinity", pinfo["cpu_affinity"])
if hasattr(proc, "cpu_num"):
    print_("cpu-num", pinfo["cpu_num"])

print_('memory', str_ntuple(pinfo['memory_info'], convert_bytes=True))
print_('memory %', round(pinfo['memory_percent'], 2))
print_('user', pinfo['username'])
if psutil.POSIX:
    print_('uids', str_ntuple(pinfo['uids']))
if psutil.POSIX:
    print_('uids', str_ntuple(pinfo['uids']))
if psutil.POSIX:
    print_('terminal', pinfo['terminal'] or '')

print_('status', pinfo['status'])
print_('nice', pinfo['nice'])
if hasattr(proc, "ionice"):
    try:
        ionice = proc.ionice()
    except psutil.Error:
        pass
    else:
        if psutil.WINDOWS:
            print_("ionice", ionice)
        else:
            print_(
                "ionice",
                "class=%s, value=%s" % (str(ionice.ioclass), ionice.value),
            )

print_('num-threads', pinfo['num_threads'])
if psutil.POSIX:
    print_('num-fds', pinfo['num_fds'])
if psutil.WINDOWS:
    print_('num-handles', pinfo['num_handles'])

if 'io_counters' in pinfo:
    print_('I/O', str_ntuple(pinfo['io_counters'], convert_bytes=True))
if 'num_ctx_switches' in pinfo:
    print_("ctx-switches", str_ntuple(pinfo['num_ctx_switches']))
```

```
if pinfo['children']:
    template = "%-6s %s"
    print_("children", template % ("PID", "NAME"))
    for child in pinfo['children']:
        try:
            print_('', template % (child.pid, child.name()))
        except psutil.AccessDenied:
            print_('', template % (child.pid, ""))
        except psutil.NoSuchProcess:
            pass

if pinfo['open_files']:
    print_('open-files', 'PATH')
    for i, file in enumerate(pinfo['open_files']):
        if not verbose and i >= NON_VERBOSE_ITERATIONS:
            print_("", "[...]")
            break
        print_('', file.path)
else:
    print_('open-files', '')

if pinfo['net_connections']:
    template = '%-5s %-25s %-25s %s'
    print_(
        'connections',
        template % ('PROTO', 'LOCAL ADDR', 'REMOTE ADDR', 'STATUS'),
    )
    for conn in pinfo['net_connections']:
        if conn.type == socket.SOCK_STREAM:
            type = 'TCP'
        elif conn.type == socket.SOCK_DGRAM:
            type = 'UDP'
        else:
            type = 'UNIX'
        lip, lport = conn.laddr
        if not conn.raddr:
            rip, rport = '*', '*'
        else:
            rip, rport = conn.raddr
        line = template % (
            type,
            "%s:%s" % (lip, lport),
            "%s:%s" % (rip, rport),
            conn.status,
        )
        print_('', line)
else:
    print_('connections', '')

if pinfo['threads'] and len(pinfo['threads']) > 1:
    template = "%-5s %12s %12s"
    print_('threads', template % ("TID", "USER", "SYSTEM"))
    for i, thread in enumerate(pinfo['threads']):
        if not verbose and i >= NON_VERBOSE_ITERATIONS:
            print_("", "[...]")
            break
        print_('', template % thread)
    print_('', "total=%s" % len(pinfo['threads']))
else:
    print_('threads', '')

if hasattr(proc, "rlimit"):
    res_names = [x for x in dir(psutil) if x.startswith("RLIMIT")]
    resources = []
    for res_name in res_names:
        try:
            soft, hard = proc.rlimit(getattr(psutil, res_name))
        except psutil.AccessDenied:
            pass
```

```
                else:
                    resources.append((res_name, soft, hard))
            if resources:
                template = "%-12s %15s %15s"
                print_("res-limits", template % ("RLIMIT", "SOFT", "HARD"))
                for res_name, soft, hard in resources:
                    if soft == psutil.RLIM_INFINITY:
                        soft = "infinity"
                    if hard == psutil.RLIM_INFINITY:
                        hard = "infinity"
                    print_(
                        '',
                        template
                        % (RLIMITS_MAP.get(res_name, res_name), soft, hard),
                    )

        if hasattr(proc, "environ") and pinfo['environ']:
            template = "%-25s %s"
            print_("environ", template % ("NAME", "VALUE"))
            for i, k in enumerate(sorted(pinfo['environ'])):
                if not verbose and i >= NON_VERBOSE_ITERATIONS:
                    print_("", "[...]")
                    break
                print_("", template % (k, pinfo['environ'][k]))

        if pinfo.get('memory_maps', None):
            template = "%-8s %s"
            print_("mem-maps", template % ("RSS", "PATH"))
            maps = sorted(pinfo['memory_maps'], key=lambda x: x.rss, reverse=True)
            for i, region in enumerate(maps):
                if not verbose and i >= NON_VERBOSE_ITERATIONS:
                    print_("", "[...]")
                    break
                print_("", template % (bytes2human(region.rss), region.path))


    def main():
        parser = argparse.ArgumentParser(
            description="print information about a process"
        )
        parser.add_argument("pid", type=int, help="process pid", nargs='?')
        parser.add_argument(
            '--verbose', '-v', action='store_true', help="print more info"
        )
        args = parser.parse_args()
        run(args.pid, args.verbose)


    if __name__ == '__main__':
        sys.exit(main())
```

**Приклад використання системи**

```
# Створення вузлів
host1 = FibreChannelNode(node_id=1, role='host')
storage1 = FibreChannelNode(node_id=2, role='storage')
host2 = FibreChannelNode(node_id=3, role='host')

# Створення комутатора з усіма функціями
fc_switch = FibreChannelSwitchWithMonitoring(switch_id=101)

# Підключення вузлів до комутатора
fc_switch.connect_node(host1)
fc_switch.connect_node(storage1)
fc_switch.connect_node(host2)

# Створення зон
fc_switch.create_zone(zone_id=1)
fc_switch.add_node_to_zone(zone_id=1, node=host1)
fc_switch.add_node_to_zone(zone_id=1, node=storage1)

# Передача даних у межах зони
fc_switch.transfer_data_in_zone(data="Резервна копія даних", source_node=host1,
target_node_id=2, zone_id=1)

# Додавання резервного каналу
fc_switch.add_redundant_path(source_node_id=1, target_node_id=2)

# Передача даних з використанням резервного каналу
fc_switch.transfer_data_with_redundancy(data="Критичні дані", source_node=host1,
target_node_id=2)

# Отримання статистики передачі
fc_switch.get_transfer_statistics()
```