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 побудови мережі на основі технології SDN”

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

Керівник проекту
кандидат технічних наук
_____ Улічев О.С.
«____» _____ 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

Морозу Антону Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи побудови мережі на основі технології SDN*

2. Керівник роботи *Улічев Олександр Сергійович, канд. техн. наук*
(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 побудови мережі на основі технології SDN*

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

Мороз А.С. Дослідження та програмна реалізація системи побудови мережі на основі технології SDN. 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 побудови мережі на основі технології SDN.

Метою розробки є дослідження та програмна реалізація системи побудови мережі на основі технології SDN.

Об'єктом дослідження є процес побудови мережі на основі технології SDN.

Предметом дослідження є методи побудови мережі на основі технології SDN.

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

Результат роботи – програмна реалізація системи побудови мережі на основі технології SDN.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, SDN

ABSTRACT

Moroz A.S. Research and software implementation of a network construction system based on SDN technology. 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 network construction system based on SDN technology.

The purpose of the development is the research and software implementation of the network construction system based on SDN technology.

The object of research is the process of building a network based on SDN technology.

The subject of research is methods of building a network based on SDN technology.

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 network construction system based on SDN technology.

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

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

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

The program was developed in the Python environment.

Keywords: computer engineering, SDN

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

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

ВСТУП

Актуальність теми. За два-три роки SDN пройшла шлях від концепції до конкретних технічних рішень. Як показав наш аналіз, виробники пропонують комплексні рішення SDN, цілком готові до впровадження, при цьому вони передбачають еволюційний шлях розгортання SDN у традиційних мережних інфраструктурах. Інакше кажучи, обіцяні новою технологією переваги цілком можна одержати без революційної заміни всього наявного устаткування. На ринку з'явилося досить велике число продуктів, що вписуються в концепцію SDN, а замовники всерйоз стали вивчати питання доцільності розгортання SDN і поступового переходу до програмувальних мереж – саме так ми пропонуємо йменувати SDN. Настав час проаналізувати конкретні рішення SDN, щоб допомогти замовникам у їхньому виборі. Як нам представляється, найкращий спосіб зробити це – сформулювати типову (якщо, звичайно, поняття «типове» застосовно до такої інноваційної області, як SDN) завдання й запропонувати для нього рішення. Ефектною й одночасно ефективною можливістю багатьох рішень SDN є візуалізація трафіку у фізичній і логічній мережах, що дозволяє спростити експлуатацію, прискорити виявлення й усунення несправностей, спростити моніторинг SLA.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи побудови мережі на основі технології SDN.

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

– Огляд існуючих систем побудови мережі на основі технології SDN.

– Дослідження системи побудови мережі на основі технології SDN.

– Програмна реалізація системи побудови мережі на основі технології SDN.

Об'єктом дослідження є процес побудови мережі на основі технології SDN.

Предметом дослідження є методи побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN.

– Розроблено вітчизняний продукт побудови мережі на основі технології SDN, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN, 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

Вигаданий замовник – велика компанія з територіально розподіленою структурою. Один із сегментів своєї мережі передачі даних, що побудована на основі традиційної архітектури, він вирішив перевести на технологію SDN. Замовник розраховує, що впровадження SDN дозволить йому автоматизувати ряд важливих процедур, пов'язаних з експлуатацією мережі. У їхнє число входять керування списками контролю доступу, блокування трафіку певних додатків, користувачів і їхніх груп відповідно до прийнятої політики безпеки, а також налаштування алгоритмів пріоритизації й забезпечення якості обслуговування (QoS), конфігурування нових мережних пристроїв.

1.2 Область застосування

Замовник – велика компанія з територіально розподіленою структурою – використовує мережу передачі даних, побудовану на основі традиційної архітектури. Один з її сегментів вирішено перевести на технологію SDN. Цей сегмент відносно автономний. Використовуване в ньому мережне устаткування морально застаріло, тому планується його повна заміна.

Модернізуємий сегмент охоплює один великий регіональний офіс (300 мережних портів) і чотири філії (по 50 портів) у кожному. Прокладена СКС забезпечує канали зі швидкістю 1 Гбіт/с до робочих місць (90% підключень) і 10 Гбіт/с (10% підключень) до серверів і для передачі даних у ядрі локальної мережі. Зв'язок між віддаленими офісами здійснюється через Інтернет по VPN, канал доступу кожного офісу до Інтернету має пропускну здатність 1 Гбіт/с.

Замовник просить запропонувати рішення для побудови зазначеного сегмента мережі на основі технології SDN.

1. Контролер SDN. Контролер повинен являти собою відказостійку систему, переважно з територіальним поділом пристроїв, що резервують. Бажано, щоб на «південному» інтерфейсі контролера SDN використовувався стандартний протокол OpenFlow (якщо постачальник рекомендує інший протокол, прохання обґрунтувати).

2. Комутатори. Замовник просить постачальника розглянути можливість використання в проекті комутаторів без передвстановленої ОС (так звані Bare Metal Switch). Якщо цей варіант на думку постачальника небажаний, прохання пояснити причину.

3. Функціональність. Замовник розраховує, що рішення SDN дозволять йому автоматизувати наступні процедури:

– керування списками контролю доступу на мережних пристроях;

– блокування певних додатків, користувачів і їхніх груп відповідно до заданих правил безпеки;

– налаштування алгоритмів пріоритизації й забезпечення якості обслуговування (QoS) для існуючих і нових додатків;

Нові додатки класифікуються, їм призначається клас обслуговування, потім політики QoS застосовуються до всієї мережі, а не до окремого пристрою.

– налаштування нових мережних пристроїв.

Контролер самостійно виявляє знову підключений пристрій (комутатор) і конфігурує його.

4.SDN-додатки й мережні сервіси. Які додаткові SDN-додатки й мережні сервіси надає запропоноване рішення?

5. Переваги. Які ще переваги може забезпечити запропоноване вами рішення SDN?

6. Інтеграція із традиційною мережею. Яким образом можлива інтеграція сегмента SDN із сегментами мережі, побудованими по традиційній архітектурі?

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи побудови мережі на основі технології SDN, 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

TamoGraph® Site Survey

TamoGraph Site Survey – потужний і зручний інструмент для збору, візуалізації й аналізу даних у мережах Wi-Fi стандарту 802.11 a/b/g/n/ac. Для впровадження й експлуатації бездротових мереж потрібні професійні програмні продукти, які дозволяють значно спростити виконання таких складних і трудомістких завдань як побудова карт покриття, аналіз інтерференції й рівня сигналу, розподіл Wi-Fi-каналів, і т.п. Саме для рішення цих проблем і призначений TamoGraph.

Більші офіси компаній або банків, готелю, кафе – скрізь, де вже використовується або планується розвертати Wi-Fi мережа, там TamoGraph допоможе істотно скоротити час і витрати на планування й обслуговування мережі, збільшить її продуктивність, розширить покриття, можливо навіть без придбання додаткового устаткування.

Ключові особливості:

– Простий і швидкий збір даних.

– Пасивні, активні й предиктивні інспектування.

– Спектральний аналіз.

– Всебічний аналіз бездротових мереж зі зручними й наочними візуалізаціями рівня сигналу, перешкод, зон покриття точок доступу, швидкості передачі даних, мережних проблем.

– Автоматичне знаходження точок доступу.

– Детальна інформація про кожну точку доступу: робітник канал, максимальна швидкість передачі даних, дані про компанію-виробнику, тип шифрування й т.д.

– Повна підтримка мереж стандартів 802.11ac, 802.11a, 802.11b, 802.11g і 802.11n.

– Інспектування з використанням GPS.

– Широкий вибір шаблонів антен, що дозволяють моделювати точки доступу основних постачальників устаткування Wi-Fi: Cisco, Aruba, Ruckus, Aerohive, Ubiquiti, Meraki, і багатьох інших.

– Докладні звіти в PDF і HTML форматах.

– Приваблива ціна.

Навіщо здійснювати інспектування Wi-Fi мереж

Основна причина, що робить інспектування й профілювання бездротових мереж (site survey) необхідним процесом, полягає в тім, що вгадати поширення радіохвиль украй складно, особливо в закритих приміщеннях. Практично неможливо врахувати всі фактори, які можуть вплинути на продуктивність і якість роботи мережі Wi-Fi. Зміни зовнішнього середовища, навіть така, здавалося б, дріб'язок, як ноутбук зі старим адаптером стандарту 802.11g, за допомогою якого ваш новий працівник увійшов у бездротову мережу офісу, можуть серйозно вплинути на продуктивність мережі. Крім того, з огляду на широке поширення бездротового зв'язку, перешкоди від сусідніх Wi-Fi мереж грають дуже важливу роль. Тому регулярне інспектування мережі з використанням професійних інструментів так необхідно.

Коли проводити інспектування

До розгортання мережі: на цьому етапі інспекція необхідна для того, щоб переконатися в працездатності наміченого плану розгортання мережі. Розміщення тимчасових точок доступу й швидка оцінка характеристик мережі дозволять інженерам оптимізувати розташування точок доступу і їхніх антен,

визначити оптимальну кількість і типи точок доступу, а також уникнути зон невпевненого прийому сигналу.

Після розгортання мережі: на цьому етапі необхідно провести повне профілювання мережі й переконатися в тім, що продуктивність і зона покриття мережі відповідають заявленим при плануванні вимогам. Коли конфігурація й розташування бездротового устаткування остаточно визначено, треба задокументувати дані звіту для використання в майбутньому.

Регулярні контрольні перевірки на постійній основі: обслуговування високошвидкісних мереж із широкими зонами покриття вимагає постійних контрольних перевірок. Нові користувачі, нове обладнання, розширення зони покриття, прилеглі бездротові мережі – все це може вкрай негативно вплинути на роботу вашої мережі Wi-Fi, тому такі перевірки повинні проводитися на регулярній основі.

Кому потрібний TamoGraph:

– Системним адміністраторам
– Операторам Wi-Fi мереж
– Виробникам бездротового устаткування

Системні вимоги

TamoGraph підтримує роботу в операційних системах Windows 7, Windows 8, Windows 8.1, Windows Server 2008 R2, Windows Server 2012 і Windows Server 2012 R2 (як в 32-, так і в 64-бітних версіях). Ви також може проводити інспектування, використовуючи комп'ютери MacBook. Для проведення пасивних інспектувань необхідний сумісний бездротової адаптер.

CommView®

CommView – це програма для перехоплення й аналізу трафіку Інтернету й локальної мережі. Вона збирає інформацію про дані, що проходять через модем (dial-up) або мережну карту й декодує аналізовані дані.

За допомогою CommView ви можете бачити список мережних з'єднань, IP-статистику й досліджувати окремі пакети. IP-пакети декодуються аж до

найнижчого рівня з повним аналізом розповсюджених протоколів. Надається повний доступ до неопрацьованих даних. Перехоплені пакети можуть бути збережені у файл для наступного аналізу. Гнучка система фільтрів дозволяє відкидати непотрібні вам пакети або перехоплювати тільки ті пакети, які ви захочете.

До складу CommView входить модуль VoIP, призначений для поглибленого аналізу, записи й відтворення голосових повідомлень SIP і H.323.

Програма може працювати на комп'ютері під керуванням Windows 7/8/8.1/10 або Windows Server 2008/2012 (підтримуються 32- і 64-бітні версії) і одержувати дані від будь-якого мережного адаптера Ethernet, Wi-Fi або віртуального адаптера (VPN або dial-up).

CommView здійснює повний аналіз більше 100 розповсюджених протоколів.

Наша нова технологія віддаленого моніторингу дозволяє користувачам CommView перехоплювати трафік будь-якого комп'ютера, на якому запущений Remote Agent, поза залежністю від фізичного місця розташування цього комп'ютера. Щоб скористатися цією унікальною можливістю, вам необхідно використовувати CommView Remote Agent. Кликніть тут, щоб одержати додаткову інформацію.

Що ви можете робити за допомогою CommView:

– Перехоплювати інтернет-трафік і/або трафік локальної мережі, що проходить через вашу мережну карту або модем.

– Переглядати перехоплені й декодовані пакети в реальному часі або в offline-режимі.

– Бачити докладну статистику IP-з'єднань: IP-адреси, порти, сесії й т.д.

– Реконструювати TCP-сесії й UDP-потоки.

– Бачити, яке додаток одержує або посилає пакети.

– Переглядати графіки розподілу протоколів, завантаження мережі, списки активних мережних вузлів і їхню статистику.

– Генерувати звіти.

– Робити пошук по рядках або hex-даним по вмісту перехоплених пакетів.

– Експортувати й імпортувати архіви зі збереженими пакетами в/з форматів Sniffer®, EtherPeek™, AiroPeek™, Observer®, NetMon і Wireshark/Tcpdump.

– Налаштовувати попередження, які повідомляють вас про важливі події: підозрілі пакети, високе завантаження мережі, невідомі адреси й т.д.

– Передавати будь-яку IP-адресу в SmartWhois для швидкого й простого одержання інформації про нього.

– Перехоплювати loopback-трафік на локальній машині.

– Багато інших речей!

Кому потрібна CommView:

– Адміністраторам локальних мереж.

– Професіоналам в області мережної безпеки.

– Кожному, хто бажає бачити повну картину мережного трафіку, що проходить через комп'ютер або сегмент локальної мережі, або з'ясувати, чи не є встановлена вчора програма "трояном".

– Програмістам, що розробляють і відлагоджують мережне програмне забезпечення.

SmartWhois®

SmartWhois – це зручний засіб одержання всієї доступної інформації про будь-якому IP-адресі, імені комп'ютера або домені, включаючи країну, штат або провінцію, місто, назву компанії-провайдеру, ім'я адміністратора й контактну інформацію служби технічної підтримки. Програма допоможе знайти відповіді на деякі важливі питання:

– Хто є власником домену?

– Коли був зареєстрований домен?

– Яка контактна інформація власника домену?

– Хто є власником даного блоку IP-адрес?

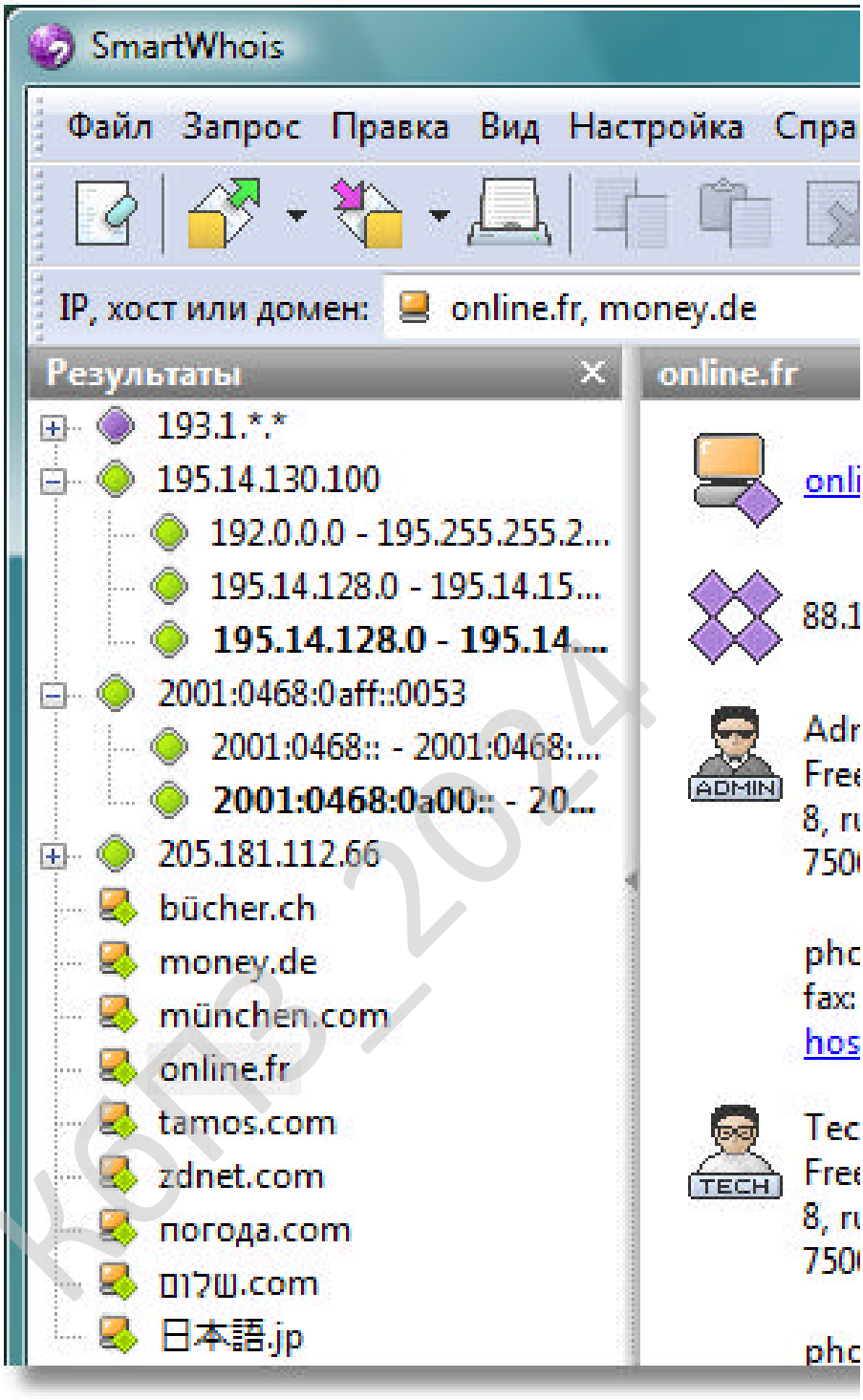

Рисунок 2.1 – Інтерфейс користувача SmartWhois

Програма безпомилково вибере правильну базу даних з більш ніж 100 баз даних по усьому світі й надасть найдетальніші результати протягом декількох секунд. SmartWhois підтримує домени IDN, і ви можете робити запити доменів,

імена яких містять символи національних алфавітів (таких як німецькі умляути або французький гравіс), або ж повністю складаються із символів китайського, івриту, української й іншої мов. Крім цього, SmartWhois підтримує роботу з IPv6-адресами.

Можливості:

– Інтелектуальна робота: програма завжди опитує тільки потрібну базу даних; вам не потрібно витрачати час, опитуючи всі бази даних.

– Інтеграція з Microsoft Internet Explorer і Microsoft Outlook. Швидкий пошук власника домену й IP-адреси в заголовках повідомлень електронної пошти.

– Збереження результатів в архіві: ви можете створити свою базу даних, яку можна переглядати, не підключаючись до Інтернету.

– Завантаження й обробка списків IP-адрес або доменів.

– Кешування отриманих результатів.

– Зіставлення імен і IP-адрес, кешування DNS-запитів.

– Інтеграція з мережним монітором CommView: SmartWhois може бути викликаний з CommView для швидкого одержання інформації.

– Виклик SmartWhois з вашого додатка прямо. Звернетеся до розділу SmartWhois FAQ.

– Запити по масці.

– Консоль Whois для побудови спеціальних запитів.

– Коди країн для довідки.

– Інтерфейс, що налаштовується.

– Підтримка роботи через брандмауера SOCKS5.

Кому потрібний SmartWhois:

– Кожному, хто використовує стандартну утиліту Whois: SmartWhois заощадить масу часу й зробить те, що не можуть зробити стандартні утиліти Whois.

– Людям, які ненавидять спам і хочуть з'ясувати джерело підозрілих електронних повідомлень: переглянете заголовки листа й установите щирого відправника! Ви також можете відправити електронне повідомлення адміністраторові мережі за допомогою одного кличу мишею.

– Веб-майстрам, які хочуть більш докладно вивчити балка-файли й не можуть визначити багато які IP-адреси.

– Авторам Shareware-програм, яким потрібно знати, звідки насправді прийшло замовлення.

– Людям, які хочуть установити джерело підозрілого електронного повідомлення по його заголовках.

Essential NetTools™

Essential NetTools – це набір мережних утиліт для діагностики мереж і моніторингу мережних з'єднань вашого комп'ютера. Це незамінний інструмент для кожного, кому потрібний набір потужних мережних інструментів для щоденного використання. Він містить у собі:

–**NetStat**: відображає список вхідних і вихідних з'єднань вашого комп'ютера, включаючи інформацію з відкритих TCP- і UDP-портам, IP-адресам і стану з'єднань. Від інших утиліт NetStat відрізняє здатність прив'язувати відкриті порти до додатків, що володіють ними. Можливе налаштування системи попереджень на вхідні й вихідні з'єднання.

–**NBScan**: сканер NetBIOS, потужний і швидкий інструмент для дослідження мереж. NBScan може сканувати мережа в заданому діапазоні IP-адрес і становити список комп'ютерів, що мають службу NetBIOS поділюваних ресурсів і таблицю їхніх імен. На відміну від стандартної утиліти, що поставляється з Windows, NBScan забезпечує дружній графічний інтерфейс і легке керування файлами lmhosts, а також паралельне сканування, що дозволяє перевірити мережа класу С менш, ніж за хвилину. NBScan може полегшити виконання щоденних завдань системними інтеграторами, адміністраторами й аналітиками.

–**PortScan**: сканер TCP-портів з розширеними можливостями, що дозволяє сканувати мережа на предмет активних портів. Цей інструмент сканує як у звичайному (повнозв’язному), так і в схованому (half-open) режимах.

–**HostAlive**: модуль моніторингу мережі, що періодично перевіряє, чи активний хост і чи працюють на ньому мережні сервіси, наприклад HTTP- або FTP-сервер.

–**EmailVerify**: перевіряє, чи існує адреса електронної пошти, зв'язуючись по SMTP з відповідним поштовим сервером.

–**Shares**: контролює й веде звіт зовнішніх підключень до поділюваних ресурсів вашого комп'ютера, а також надає швидкий і легкий шлях підключення до віддалених ресурсів.

–**SysFiles**: зручний редактор для п'яти важливих системних файлів: services, protocol, networks, hosts і lmhosts.

–**NetAudit (NetBIOS Auditing Tool)**: дозволяє вам проводити різних перевірок безпеки мережі й/або окремих комп'ютерів, на яких запущена служба доступу до поділюваних ресурсів по NetBIOS. Цей інструмент допоможе вам ідентифікувати потенційні проломи в безпеці.

–**RawSocket**: наділяє вас можливістю встановлювати низькорівневе з'єднання TCP для виявлення проблем з різними мережними службами. Подання даних різними квітами, а також зручний інтерфейс роблять його відмінним інструментом для щоденної роботи й адміністратора, і програміста.

–**WiFiMan**: показує інформацію про встановлені в комп'ютері бездротових адаптерах, доступних бездротових мережах, дозволяє редагувати відповідні профілі підключення.

–**TraceRoute і Ping**: знайомі всім утиліти, які постачені безліччю функцій і наочним поданням результатів, дозволять вам досліджувати Інтернет і виявляти проблеми з'єднань.

–**NSLookup**: дозволяє переводити адреси IP в імена хостів і навпаки, одержувати аліаси й виконувати розширені DNS-запити, такі як MX або CNAME.

–**IPBlackList**: перевіряє, чи включена IP-адреса в різні чорні списки IP-адрес: бази адрес спамерів, відкриті проксі й рілеі електронної пошти. Цей інструмент дасть вам відповідь на питання, чому перевіряєма IP-адреса відкидається деякими мережними ресурсами, такими як поштові сервера.

–**ProcMon**: відображає список активних процесів з повною інформацією про знаходження програми, виробниках, ідентифікаторі процесів, завантажених модулях. Із цим інструментом ви можете переглянути статистику по споживанню процесорного часу, розпізнавати сховані додатки, зупиняти поточні процеси й ефективніше управляти використанням ресурсів комп'ютера.

–**SNMPAudit**: просунутий сканер SNMP-пристроїв. Дозволяє швидко локалізувати SNMP-пристрою в обраному мережному діапазоні й одержати налаштовується виборку, що, даних від кожного з них. Для більше детального вивчення пристрою користувач може використовувати SNMP-браузер.

Інші можливості: створення звіту в HTML, текстовому й CSV-форматах, швидка обмін IP-адресами в різних інструментах; геолокація IP-адрес, докладна інформація про систему, що налаштовується інтерфейс і багато чого іншого.

CommView® for WiFi

CommView for WiFi – це програма для моніторингу й аналізу мережних пакетів у бездротових мережах стандартів 802.11 a/b/g/n/ac, що поєднує в собі продуктивність, гнучкість і зручність використання.

CommView for WiFi може захоплює всі мережні пакети, передані в ефірі, для наступного детального відображення важливої інформації: списку точок доступу й вузлів, статистики по кожному вузлі й каналу, рівня сигналу, списку пакетів і мережних з'єднань, графіків розподілу протоколів і т.д. За допомогою цієї інформації CommView допоможе вам переглядати й докладно аналізувати кожний пакет, виявляти проблеми в роботі мереж, більш оперативно усувати несправності в програмному забезпеченні й устаткуванні.

До складу CommView for WiFi також входить модуль VoIP, призначений для поглибленого аналізу, записи й відтворення голосових повідомлень SIP і H.323.

Пакети можна дешифрувати з використанням користувальницьких ключів WEP або WPA-PSK і декодувати аж до найнижчого рівня. Підтримуючи більше 100 протоколів, CommView for WiFi дозволяє детально вивчати захоплені пакети, використовуючи зручну, деревоподібну систему відображення протокольних рівнів і заголовків пакетів.

Перехоплені пакети можуть бути збережені у файл для наступного аналізу. Гнучка система фільтрів дозволяє відкидати непотрібні вам пакети або перехоплювати тільки ті пакети, які ви захочете. Попередження, що налаштовуються, дозволяють повідомляти користувача про важливі події, таких як підозрілі пакети, високе завантаження мережі або невідомих адрес.

CommView for WiFi – це повнофункціональний і доступний інструмент для адміністраторів бездротових мереж, фахівців в області мережної безпеки, мережних програмістів або тих, хто хоче бачити всю картину трафіку в бездротовій мережі. Ця програма працює в Windows 7/8 / 8.1 / 10 або Windows Server 2008 / 2012 (підтримуються 32- і 64-бітні версії); для роботи необхідний сумісний бездротової мережний адаптер. Ви також можете працювати з CommView for WiFi на комп'ютерах MacBook. Список сумісних адаптерів наведений нижче:

Якщо ваш адаптер не зазначений у цьому списку, будь ласка, кликніть тут.

Що ви можете робити за допомогою CommView for WiFi:

– Сканувати ефір для виявлення станцій і точок доступу Wi-Fi.

– Перехоплювати бездротової трафік 802.11a, 802.11b, 802.11g, 802.11n і 802.11ac.

– Уводити ключі WEP або WPA для дешифрації зашифрованих пакетів.

– Бачити докладну статистику IP-з'єднань: IP-адреси, порти, сесії, і т.д.

– Реконструювати TCP-сесії.

– Налаштовувати попередження, які повідомляють вас про важливі події, таких як підозрілі пакети, високе завантаження мережі, невідомі адреси й т.д.

– Бачити діаграми IP-протоколів і протоколів верхнього рівня.

– Стежити за завантаженням мережі.

– Переглядати перехоплені й декодовані пакети в реальному часі.

– Робити пошук по рядках або hex-даним по вмісту перехоплених пакетів.

– Зберігати й завантажувати пакети.

– Імпортувати й експортувати файли у форматах Sniffer®, EtherPeek™, AiroPeek™, Observer®, NetMon, і Tcpdump/Wireshark.

– Передавати будь-яку IP-адресу SmartWhois для швидкого й простого одержання інформації про нього.

– Здійснювати одночасний захват даних з декількох каналів (при використанні декількох сумісних USB-адаптерів).

– Здійснювати захват пакетів A-MPDU і A-MSDU.

Кому потрібна CommView for WiFi:

– Адміністраторам бездротових мереж.

– Професіоналам в області мережної безпеки.

– Домашнім користувачам, які хочуть відслідковувати активність у своїй бездротовій мережі.

– Програмістам, що розробляють програмне забезпечення для бездротових мереж.

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.

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 побудови мережі на основі технології SDN.

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

Головним елементом будь-якого рішення SDN, безумовно, є контролер. З мережною інфраструктурою контролер взаємодіє через «південні» інтерфейси, основний з них – OpenFlow. Шість із семи представлених контролерів підтримують зазначений протокол. Виключення становить тільки контролер APIC-EM компанії Cisco. На момент підготовки матеріалу в якості «південного» API на цьому контролері був доступний тільки Cisco CLI (відповідно, і працювати він міг тільки з комутаторами й маршрутизаторами Cisco). Однак уже в наступних версіях ПЗ APIC-EM запланована підтримка устаткування інших виробників або комутаторів без передвстановленої ОС (bare-metal switch) за рахунок використання OpenFlow і Cisco OnePK.

Загальний підхід Cisco складається в поділі мережі замовника на логічні домени (ЦОД, WAN, кампус, сервісна інфраструктура й т.д.) і використанні для кожного домену (або груп доменів) спеціалізованого SDN-контролера, що найбільше ефективно вирішує стандартні для обраного домену завдання. Для забезпечення наскрізного сервісу, що вимагає взаємодії декількох інфраструктурних доменів, Cisco поставляє рішення по оркестрації (Network Service Orchestrator, NSO). Для поставленого завдання Cisco запропонувала контролер APIC-EM (Application Policy Infrastructure Controller – Enterprise Module), спеціально розроблений для корпоративних кампусних і розподілених (WAN) мереж. Це ідеологічний і технологічний спадкоємець контролера APIC, використовуваного в рамках архітектури Cisco Application Centric Infrastructure (ACI) для керування інфраструктурою ЦОДу.

Контролер APIC-EM реалізує функціональність керування мережними елементами, залишаючи всі інші завдання зовнішнім системам керування й

сторонніх додатків, взаємодіючим з APIC-EM через «північний» програмний інтерфейс REST.

На відміну від Cisco, компанія NEC є затятим прихильником стандартизованого підходу на основі OpenFlow. Представники NEC називають свій контролер «лідером по відповідності стандартам OpenFlow і сумісності з комутаторами інших виробників», що щорічно підтверджується більшим числом тестів. На «північній» стороні контролер NEC підтримує JSON, XML і SOAP.

Саме собою що розуміє вважають підтримку протоколу OpenFlow і в компанії Huawei. При цьому Huawei створила розширення стандарту OpenFlow – технологію Protocol Oblivious Forwarding (POF), назад сумісну з OpenFlow. Цей підхід забезпечує можливість використовувати як OpenFlow, так і традиційні механізми маршрутизації для передачі й керування трафіком. Таким чином, замовник може здійснити плавний перехід до SDN. Подібну міграцію пропонують і інші компанії (див. нижче).

Для керування SDN-Устаткуванням контролер HP VAN SDN Controller, крім OpenFlow 1.0 і 1.3.1, також підтримує SNMP і NetConf. На «північній» стороні він надає відкриті програмні інтерфейси на Java і REST API для запуску додатків SDN і їхньої інтеграції із зовнішніми системами (наприклад, із системами керування й оркестрації). Крім того, рішення HP підтримує динамічне завантаження й запуск додатків SDN безпосередньо на самому контролері за рахунок використання відкритої архітектури на базі OSGi.

Компанії Brocade і Extreme Neworks запропонували замовникові контролери на базі систем з відкритим вихідним кодом OpenDaylight. Як відзначають у компанії Extreme Neworks, у своєму рішенні One Controller вони поліпшили захист і розширили функціональність платформи OpenDaylight. На «південному» інтерфейсі використовується стандартний протокол OpenFlow v.1.3. ПЗ OneFabric Control Center і OneFabric Connect забезпечують API на «північному» інтерфейсі для підтримки додаткової, розширеної функціональності, зокрема, з використанням компонентів керування мережею –

Netsight, уніфікованого доступу до мережі – NAC, моніторингу роботи додатків у мережі – Purview.

Комерційна версія контролера OpenDayLight від компанії Brocade – Brocade SDN Controller (BSC) – на «південній» стороні, крім OpenFlow 1.0/1.3, підтримує NETCONF, OVSDB, BGP-LS, PCEP. На «північному» інтерфейсі BSC має веб-сервісний RESTful API, для роботи з яким можуть використовуватися високорівневі мови програмування Python, Ruby, Perl.

Замовник одержав і рішення на основі контролера Runos, відмінними рисами якого, за твердженням представників ЦПІКС, є висока продуктивність (пропускна здатність 8 млн подій у секунду, затримка на обробку одного запиту 30 мкс) і зручність розробки. Проект Runos перебуває у відкритому доступі (http://arccn.github.io/runos/) на умовах ліцензії Apache 2.0, що повинне сприяти широкому поширенню контролера, його розвитку й доробці сторонніми розроблювачами. У відкритому доступі перебувають ядро контролера, базовий набір сервісів і додатків (визначення топології, побудова маршруту, статистика й моніторинг, інтерфейс REST, графічний інтерфейс). У комерційній версії контролер Runos має механізми резервування, масштабованості й розподіленого керування.

Важливим моментом є резервування контролера. Не зрячи у своєму завданні замовник особливо обмовило необхідність відказостійкої схеми. І всі постачальники забезпечили її – подробиці нижче в розділах, присвячених конкретним рішенням.

Вибір комутаторів

Незважаючи на те що замовник наполегливо цікавився можливістю використання в проекті комутаторів без передвстановленої ОС (так звані bare metal switch або white box), більшість постачальників зволіли традиційні мережні пристрої, але з підтримкою SDN.

Пояснюючи причину того, що Huawei не пропонує у своїх рішеннях комутатори класу bare metal, тому що при впровадженні SDN важливо

забезпечити наступність архітектури й зберегти працездатність наявних додатків, а існуючі сервіси й протоколи орієнтовані на традиційні мережні засоби.

Представляючи свої SDN-рішення в цілому, компанія HP відзначила можливість побудови мережної інфраструктури з комутаторами на базі відкритої платформи (white box), але рекомендувала їхнє використання в ЦОДух більших масштабів – від 200 To-комутаторів або від 10 000 портів. Такі інфраструктури можна реалізовувати на базі комутаторів HP серії Altoline. На думку фахівців HP, вони найбільш ефективні для ЦОДів, у яких в основному використовуються додатки Open Source, хмарні платформи OpenStack/CloudStack, рішення HPC на базі Hadoop, бази даних NoSQL (Cassandra/HBase) і т.п. Очевидно, що це не випадок нашого замовника.

Хоча більшість постачальників рекомендували замовникові свої ж комутатори, дві компанії, Extreme Networks і NEC, не стали обмежувати його вибір, указавши можливості установки комутаторів інших виробників – головне, щоб вони підтримували стандарти OpenFlow.

У частині вибору комутаторів на тлі інших виділяється пропозиція ЦПІКС. Фахівці цієї компанії рекомендували доповнити свій контролер Runos комутаторами класу white box або комутаторами, побудованими на основі серверів x86. У першому випадку на комутатори встановлюється система Open Networking Linux з розробленим у ЦПІКС агентом OpenFlow 1.3. Такі пристрої підтримують до 48 портів 1Gb, а також чотири порти 10Gb. У другому випадку використовуються традиційні сервери Intel з більшим числом мережних інтерфейсів. Комутація здійснюється спеціальним ПЗ за рахунок ресурсів центрального процесора. Такі комутатори здатні підтримувати до 24 портів 1Gb і до 12 портів 10 Gb с сумарною гарантованою пропускною здатністю 60 Gb на пристрій.

Як відзначають фахівці ЦПІКС, важливою відмінністю комутаторів на базі серверів x86 від комутаторів white box є повна підтримка можливостей протоколу OpenFlow 1.3. У комутаторах white box, як правило, використовуються стандартні

набори мікросхем від Broadcom, які на апаратному рівні не підтримують найчастіше необхідну функціональність OpenFlow, наприклад перезапис IP-адрес.

Додатки SDN

Одне з важливих переваг SDN – можливість використання широкого набору додатків, що реалізують різні мережні сервіси й функції. Такі додатки створюються в тому числі сторонніми розроблювачами й найчастіше надаються замовникам безкоштовно.

Більшість компаній для просування рішень SDN прагнуть сформувати екосистему SDN, важливою частиною якої є розроблювачі додатка. Відзначимо пропонований HP онлайн-магазин SDN-додатків HP SDN App Store (див. урізання « SDN-додатка з магазина»). Як затверджують в HP, користувачі можуть буквально одним кличем мишки завантажувати SDN-додатка з магазина на контролер, відразу ж їх запускати й використовувати.

SDN-додатки з магазина:

– Hyperglance – додаток 3D-візуалізації мережної топології. Воно дозволяє робити моніторинг потоків трафіку в режимі реального часу, а також гнучко маніпулювати ними (перенаправляти, фільтрувати, оптимізувати утилізацію каналів і т.п.) зі зручного графічного інтерфейсу.

– SDN Privatizer – безкоштовний додаток, що реалізує функціональність Private VLAN у масштабах всієї мережі, що дозволяє ізолювати друг від друга різні групи користувачів, у тому числі, підключених до різних комутаторів або навіть розташовуються в різних сегментах мережі.

– BlueCat DNS Director – безкоштовний додаток, що перехоплює DNS-запит користувача й, яка би IP-адреса сервера в ньому не був зазначений, заміняє його на заданий адміністратором адреса корпоративного DNS, тим самим забезпечуючи додатковий рівень безпеки IT-інфраструктури.

В NEC також підкреслюють наявність широкого спектра SDN-додатків від компаній-партнерів. Ці додатки вирішують питання оптимізації мережі

(балансування навантаження, WAN-Оптимізація), аналізу її продуктивності, фільтрації й керування правами доступу (DPI), безпеки (MCE, захист від DDoS і злобливого ПЗ), оптимізації трафіку й т.д. Зі списком інтегрованих з SDN-контролером NEC додатків P-Flow можна ознайомитися, зареєструвавшись в екосистемі NEC SDN Partner Space. У рамках даної ініціативи також здійснюються перевірка на сумісність і тестування комутаторів OpenFlow інших виробників, які надалі можуть використовуватися в мережах SDN під управлінням контролера NEC.

Фахівці NEC звертають увагу на переваги концепції сервісних ланцюжків (Service Chaining), коли різні необхідні користувачеві функції можуть вибиратися й комбінуватися із загального пула для конкретної віртуальної мережі VTN. Це можуть бути класичні для IP-мереж L2/L3 функції контролю доступу, пріоритизації трафіку, керування політиками QoS і т.д., а реалізовані вони можуть бути як у вигляді окремого SDN-додатка, так і на базі апаратного компонента. Такий підхід дозволяє власникові мережі SDN створити набір VTN, архітектурно й функціонально оптимізованих відповідно до вимог користувачів, а також динамічно реагувати на штатні й позаштатні ситуації. Наприклад, система захисту від DDoS-атаки або пристрій фільтрації трафіку можуть включатися в структуру мережі й задіятися тільки у випадку виявлення погрози.

3.2 Розробка структурної схеми

ЦПІКС

Проект ЦПІКС, як уже говорилося, виділяється вибором комутаторів (пристрою white box або на основі серверів x86), у плані ж архітектури він більш-менш типовий. У центральному офісі кінцеві користувачі підключаються до шести граничних комутаторів, кожний з яких з метою резервування приєднується до двох центральних комутаторів 10G. На серверній фабриці функціонують необхідні замовникові мережні сервіси, включаючи контролер Runos. Останній

запускається у двох екземплярах у режимі Active/Standby: у випадку «падіння» першого екземпляра керування передається на резервний контролер.

На відміну від центрального офісу, у філіях не потрібно великої кількості кінцевих портів, тому там досить двох комутаторів. Замість серверної фабрики розгортається один сервер віртуальних машин, на якому й будуть працювати необхідні мережні сервіси.

Додаток SDEnterprise для контролера Runos забезпечить необхідну функціональність, включаючи взаємодію із сервісами VPN, MCE й DPI, з інтернет-провайдером (BGP, MPLS), керування списками контролю доступу ACL і ін.

У центральному офісі й філіях працюють свої екземпляри контролерів SDN. У випадку втрати каналу до центрального офісу мережа філії продовжить функціонувати автономно.

Серед додаткових можливостей, які надає рішення ЦПІКС, – інтеграція з контролером Wi-Fi для безшовного роумінгу й можливість роботи із протоколу dot1x для автентифікації користувачів без твердої прив'язки до порту комутатора. Остання функція дозволяє користувачам мігрувати між мережними пристроями (включаючи точки доступу Wi-Fi), при цьому мережа буде автоматично підбудовуватися під нове розташування користувача.

Brocade

Для рішення завдання фахівці Brocade запропонували використовувати L 3-комутатори сімейства ICX 7000, програмний маршрутизатор Brocade vRouter 5600 і контролер Brocade SDN Controller (далі BSC). У великому офісі передбачається розгорнути дворівневу мережу, у ядрі якої встановити пари високопродуктивних комутаторів ICX 7750 (вони ж можуть використовуватися для підключення серверів), зв'язаних твінаксіальними кабелями на швидкості nx40G. Рівень доступу реалізується на базі ICX 7250. Засоби керування всією мережею (включаючи порти доступу) консолідовані на рівні ядра (архітектура SwitchPort Extender) – вся локальна мережа, по суті, являє собою один комутатор

(«розподілене шасі»). На границі мережі встановлюється відказостійка пара маршрутизаторів vRouter 5600, а у філіях – vRouter 5600 і 48-портовий ICX 7250.

Серед переваг комутаторів Brocade ICX з ОС FastIron (у порівнянні з комутаторами white box, а також пристроями ряду інших виробників) фахівці Brocade назвали підтримку вже згаданої архітектури Switch Port Extender і гібридних портів – той самий порт можна використовувати як для традиційної комутації/маршрутизації, так і для передачі трафіку відповідно до обумовленого контролера правилами. Комутатори можна об'єднати в стек за допомогою стандартних інтерфейсів 1/10/40G Ethernet, причому такий стік може бути розподіленим (до 10 км). Використовувана в комутаторах технологія Po+/Po дозволяє дистанційно (по локальній мережі) подавати електроживлення потужністю до 90 Вт.

На базі BSC можна побудувати відказостійкий кластер із трьох територіально розподілених вузлів. Як варіант можливий створення пула контролерів, схованих за однією віртуальним IP-адресою (VIP).

Як опція в проект може бути включений продукт Brocade Flow Optimizer, що разом з BSC використовується для інтелектуального керування потоками даних, виявлення аномалій і захисту від різних атак. У графічному інтерфейсі FlowOptimizer визначаються профілі трафіку й застосовувані до них правила, задані налаштування автоматично трансформуються в правила OpenFlow і за допомогою контролера в динамічному режимі передаються на мережні пристрої.

CISCO

Для комутації в мережах центрального й віддаленого офісів Cisco запропонувала комутатори серії 2960 з підтримкою Po/UPo або серій 3650/3850, що включають також функції контролера бездротового доступу. У центральному офісі варто передбачити від 8 до 20 комутаторів (по 24 або 48 клієнтських портів) – вибір, тип і кількість пристроїв визначаються топологією СКС, наявністю ЦОДу й вимог по підтримці бездротової мережі. У віддалених офісах пропонується встановити два-три комутатори зазначених серій.

Зв’язність центрального й віддаленого офісів забезпечать маршрутизатори серії ISR 4000. У центральному офісі пропонується встановити відказостійку пари маршрутизаторів Cisco ISR 4451, а у віддалених офісах (залежно від вимог відказостійкості й можливостей каналів глобальної мережі) – один або два маршрутизатори ISR 4431.

Для забезпечення відказостійкості Cisco рекомендувала розмістити контролер APIC-EM на декількох віртуальних машинах, причому ті, у свою чергу, повинні виконуватися на територіально рознесених серверних платформах. Будуть потрібні серверні платформи x86 з гіпервізором VMware ESXi.

Для складання правил, керування життєвим циклом елементів і контролю змін пропонується програмний продукт Prime Infrastructure (PI). Для забезпечення ідентифікації й контролю прав доступу – Identity Service Engine (ISE).

Серед додаткових можливостей, надаваних рішенням на базі APIC-EM, представники Cisco виділили автоматичне виявлення й налаштування нових мережних пристроїв (для цього використовуються протоколи CDP/LLDP, а також функціонал Pn-сервера з боку APIC-EM і PnP-клієнта з боку комутатора або маршрутизатора), взаємодія із системами уніфікованих комунікацій (телефонія, відео, конференції), автоматизоване забезпечення Call Admission Control (CAC), автоматизацію мережної безпеки (при інтеграції із зовнішніми системами). Рішення Cisco забезпечує візуалізацію топології й сервісів, а також застосування й візуалізацію налаштувань QoS, ACL, індивідуальних правил для кожного клієнта.

Cisco планує поставляти APIC-EM безкоштовно з набором убудованих мережних додатків (за нові спеціалізовані додатки буде, імовірно, стягуватися додаткова плата).

Extreme networks

Запропонований Extreme Neworks SDN-контролер One Controller, як уже говорилося, побудований на базі платформи з відкритим вихідним кодом OpenDaylight. Компанія не конкретизувала моделі комутаторів, рекомендувавши

лише свої продукти сімейств Summit і Black Diamond, які використовують мережну ОС EXOS з підтримкою OpenFlow v.1.3. Крім того, у запропонованому рішенні можливе використання будь-яких комутаторів з підтримкою протоколу OpenFlow v.1.0 і вище, що, природно, розширює «волю маневру» замовника.

Фахівці компанії відзначають широкі можливості платформи SDN на базі контролера OneController, зокрема, підтримку відкритого й стандартизованого механізму групових політик, а також інтеграцію OpenDaylight із платформою уніфікованих комунікацій Microsoft Skype for Business.

Крім запитаного замовником функціонала, запропоноване рішення дозволяє реалізувати ряд додаткових SDN-додатків і сервісів, включаючи автоматизацію створення віртуальних мереж, графічний інтерфейс керування трафіком, інжиніринг трафіку для бізнес-додатків, роботу систем безпеки на терабітних швидкостях і ін. Динамічному впровадженню нових сервісів допоможе можливість гнучкого перенапрямку трафіку (вибіркових потоків) на різні компоненти мережної інфраструктури, такі як система аналітики й моніторингу додатків Purview, Captive-портали бездротових мереж Wi-Fi, системи запису IP-Телефонії. Крім того, Extreme пропонує «SDN для Wireless»: повна підтримка концепції SDN і інтеграції сторонніх додатків з бездротовими мережами Wi-Fi від Extreme Networks – IdentiFi (у тому числі пріоритизація трафіку VoIP, інтеграція з рішеннями MDM, BYOD).

HP

Компанія надіслала найдетальнішу відповідь, де була описана її загальна стратегія в частині SDN, представлений весь портфель продуктів SDN і, звичайно, деталізоване рішення конкретного завдання. Відповідно до пропозиції HP, ядро регіонального офісу складуть два модульних комутатори HP 5406R zl2, які забезпечать підключення комутаторів доступу (по 10G), а також комутаторів ЦОДу, граничних маршрутизаторів і опціонального Wi-Fi-контролера HP Aruba. Для рівня доступу – підключення кінцевих пристроїв (ПК, ноутбуків, телефонів, опціональних Wi-Fi-точок HP Aruba) – призначаються сім 48-портових

комутаторів HP 3800 з підтримкою Po+, а для ЦОДу – підключення серверів і СХД (по 10Gb/FCo/iSCSI або 4/8G FC) – двох конвергентних комутатора HP 5900CP.

Для філій на базі HP проробили два варіанти. Перший передбачає установку стека із двох 48-портових комутаторів HP 2920, другий – двох модульних комутаторів HP 5406R zl2. У другому випадку в комутатори встановлюється сервісний модуль HP Advanced Services v2 zl Module із системою віртуалізації VMware vSphere, а локальний контролер SDN у вигляді віртуальної машини з ПЗ HP VAN SDN Controller інсталюється безпосередньо на цей модуль.

Для формування територіально розподіленої мережі запропоновано використовувати маршрутизатори HP MSR3044 (у регіональному офісі) і HP MSR3012 (у філіях). Вони забезпечують контроль доступу й фільтрацію трафіку на границі мережі за допомогою убудованого міжмережного екрана. Для організації VPN-підключення рекомендована технологія HP ADVPN.

Відказостійкий кластер SDN-контролерів HP VAN SDN Controller у центрі HP запропонувала реалізувати на базі серверів HP ProLiant DL360/380 Gen9 – поверх ОС Linux (Ubuntu або RHEL) або на платформі віртуалізації VMware. Можна використовувати й сторонні сервери, однак у цьому випадку HP не може гарантувати заявлені характеристики продуктивності контролера SDN, які були протестовані на серверах HP ProLiant. Кластер контролерів забезпечить резервне керування SDN-інфраструктурою філій у випадку відмови локальних контролерів у філії.

У якості основних HP запропонувала три SDN-додатки: HP Network Protector забезпечує безпека в локальній мережі; HP Network Optimizer – керування QoS у ЛОМ; HP Network Visualizer – моніторинг і діагностику ЛОМ. Крім перерахованих, можна використовувати додаткові додатки з HP SDN App Store.

HUAWEI

Ядром мережі регіонального офісу замовника, за задумом архітекторів Huawei, повинен стати модульний комутатор S7706 із установленими інтерфейсними платами з портами 10 Гбіт/с (для підключення серверів і комутаторів доступу) і 1 Гбіт/с (для підключення робочих місць). Для рівня доступу запропоновані комутатори S 5700-X-LI. Мережі філій будуть будуватися на базі комутаторів S 5720-HI.

Важливим програмним компонентом рішення є Super Virtual Fabric (SVF). SVF – це технологія віртуалізації мережної інфраструктури й уніфікованого керування мережними елементами, користувачами й додатками. Завдяки SVF мережна інфраструктура буде представлена у вигляді єдиного віртуального комутатора із централізованим керуванням конфігураціями, QoS, правилами контролю доступу й автентифікацією користувачів (причому як для головного офісу, так і філій). Крім провідної інфраструктури, SVF дозволяє віртуалізувати і бездротову мережу, при цьому обробка трафіку й автентифікація будуть вироблятися відповідно до загальних правил. Комутатор S7706 виступає в якості головного керуючого комутатора SVF-Parent, а комутатори S 5700-X-LI і S 5720-HI – у якості керованих SVF-Client, тому для останніх не потрібні індивідуальні конфігурація й керування.

Для підключення до територіально розподіленої мережі, побудови тунелів і шифрування трафіку, забезпечення безпеки й надання голосових сервісів у регіональному офісі встановлюються маршрутизатори AR2240, а у філіях – AR1220E. Найближчим часом, відповідно до концепції Agile Branch, маршрутизаторами віддалених офісів також стане можливо управляти із централізованого контролера (споконвічно заявлена робота через контролер Open Daylight, поряд із власним), так що їх не потрібно буде конфігурувати окремо.

Розробляючи свій контролер SDN, Huawei передбачила підтримку кластеризації й модульну структуру ПЗ для забезпечення відказостійкості й високої доступності. Agile Controller має ієрархічну структуру

з декількома компонентами (Management Center (MC), Service Manager (SM), Service Controller (SC)) плюс зовнішні бази даних. Кожний з компонентів може бути зарезервований, а розподілений дизайн дозволяє розмістити частина компонентів безпосередньо у філіях.

Крім забезпечення запитаних замовником базових функцій, Huawei запропонувала ряд додаткових можливостей. Так, завдяки функції Free Mobility користувач одержить єдині політики безпеки, обслуговування й виділення ресурсів, а також сервісні політики, тобто обслуговування буде однаковим поза залежністю від місця, часу, типу термінала або порту доступу. А технологія iPCA дозволить забезпечити наскрізний контроль якості на реальних потоках трафіку й визначити оптимальні шляхи передачі трафіку.

NEC

Для побудови запропонованого SDN-рішення, на базі NEC пропонується платформу NEC ProgrammableFlow, що включає контролер NEC PF6800 і лінійку комутаторів P-Flow. Відказостійкий контролер PF6800 являє собою ПЗ, що виконується на кластері, що організований на базі двох окремих фізичних серверів або віртуальних машин. Як відзначають в NEC, її платформа SDN інтегрована з відкритими платформами SDN/NFV, що розвиваються в рамках проектів OpenStack і OpenDayLight.

Для побудови мережі SDN під управлінням контролера NEC PF6800 можуть використовуватися комутатори NEC або інших виробників, що відповідають стандартам OpenFlow 1.0 і/або OpenFlow 1.3 (для побудови комерційних рішень фахівці NEC рекомендують комутатори з підтримкою OpenFlow 1.3). У лінійку комутаторів NEC P-Flow входять пристрою серій PF52xx, PF53xx і PF54xx різній ємності й продуктивності. Крім того, слід зазначити, що NEC регулярно проводить тестування свого контролера на сумісність із комутаторами інших виробників.

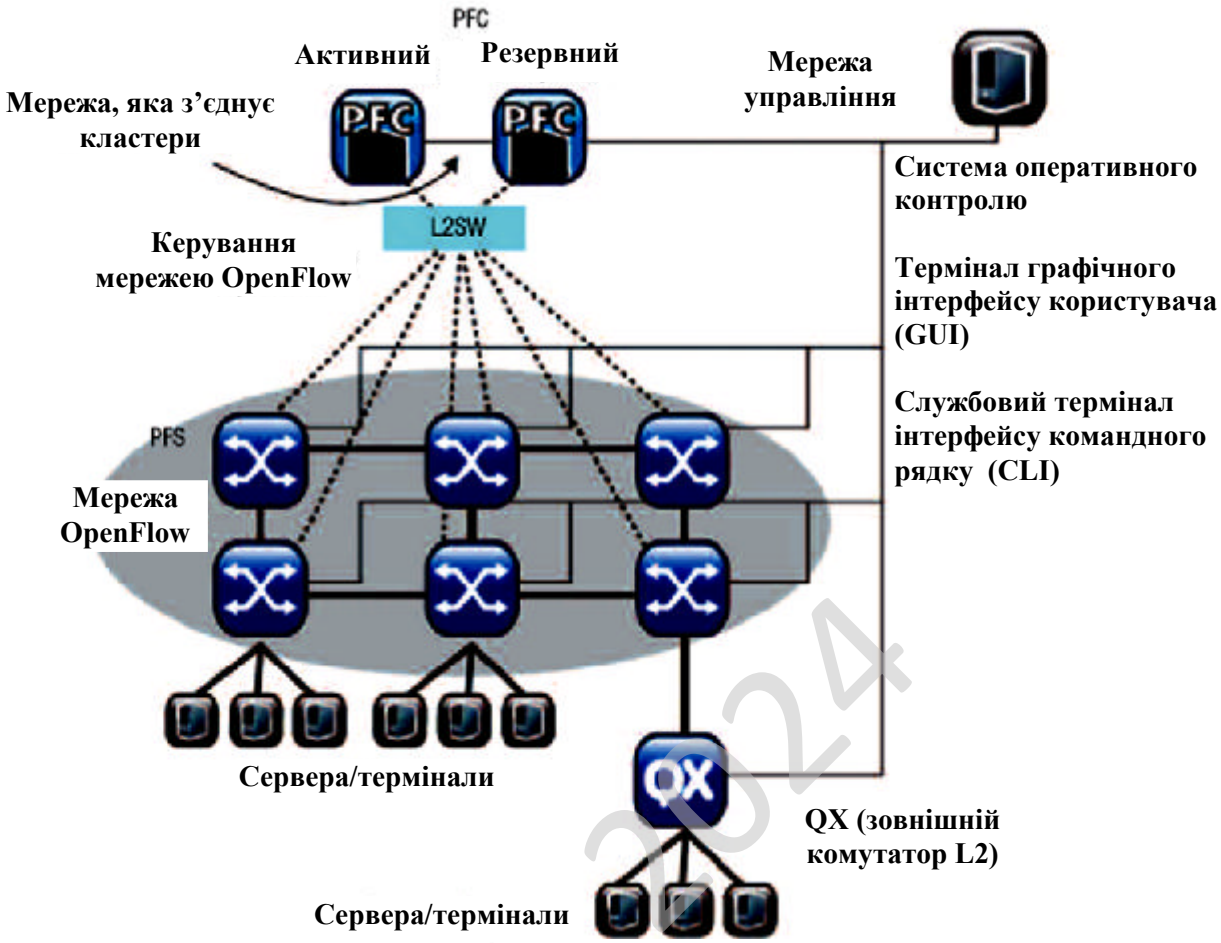

Рисунок 3.1 – Структурна схема системи

Основою запропонованого SDN-рішення, на базі NEC, є віртуалізація мережі VTN: у хмарі комутаторів OpenFlow створюється безліч незалежних мереж L2/L3 IPv4/IPv6, робота яких підтримується автоматично при змінах у фізичній мережі. Користувач може створювати VTN, динамічно змінювати конфігурацію VTN у процесі експлуатації й застосовувати мережні політики незалежно від інших VTN. При цьому для створення своїх VTN він може як скористатися пулом мережних ресурсів, пропонованих оператором мережі SDN і вже інтегрованих з контролером SDN, так і задіяти свої унікальні фізичні пристрої або віртуалізованні (VNF) компоненти.

Серед переваг запропонованого SDN-рішення, на базі NEC є підвищення продуктивності мережі (стабільна робота забезпечується навіть при 100-процентному завантаженні каналів), можливість перебудови фізичної мережі без переривання обслуговування у віртуальних мережах, підвищення безпеки за рахунок повної ізоляції віртуальних мереж другові від друга, збільшення надійності мережі завдяки самовідновленню й автоматичному перерозподілу потоків трафіку відповідно до правил. Ефектної й одночасно ефективної є візуалізація трафіку у фізичній і логічній мережах, що дозволяє спростити експлуатацію, прискорити виявлення й усунення несправностей, спростити моніторинг SLA.

Міграція (гібридні мережі)

Жоден постачальник не пропонує одним махом замінити традиційну мережу на інфраструктуру SDN. Усі підготували сценарії поступової міграції й/або побудови гібридний мереж.

Впровадження SDN не вимагає повної заміни існуючої IP-інфраструктури, а більшість переваг SDN стають доступні при повній або навіть частковій заміні ядра мережі або рівня агрегації. Запропоноване NEC рішення SDN може інтегруватися з IP-мережами на рівнях L2 (MCLAG) і L3 (VRRP/HSRP). При цьому один сегмент SDN може мати кілька підключень рівня L2 і/або L3 до мереж IP, і для всіх підключень може використовуватися єдиний пул мережних сервісів (MCE, балансувальник, DPI, Proxy і ін.), що значно спрощує завдання адміністрування. Для інтеграції декількох сегментів SDN фахівець NEC рекомендує організувати L2 VPN в існуючих мережах передачі даних.

Як зазначено у компанії Huawei, негайний перехід до SDN може привести до втрати інвестицій, частина наявних функцій може бути втрачена або істотно спрощена (наприклад, функції балансувальника або оптимізатора трафіку). Тому її комутатори Agile надають можливість саме міграції на SDN, а не радикального переходу до цієї технології. Вони здатні паралельно підтримувати два режими роботи: традиційна комутація/маршрутизація й SDN.

Фахівці Huawei особливо відзначають те, що в її комутаторах Agile використовуються мережні процесори власної розробки (Ethernet Network Processor, ENP). Традиційні ASIC обробляють дані тільки визначених протоколів, а впровадження нових сервісів (наприклад, з нестандартною інкапсуляцією) тягне апаратні зміни (редизайн мікросхем). На відміну від ASIC, процесори ENP повністю програмувальні, тому замовники можуть уже зараз почати фрагментарно впроваджувати «готові до SDN» пристрою в існуючу інфраструктуру й надалі, просто обновивши прошивання, підтримувати майбутні нові протоколи й сервіси.

По даним Cisco, у випадку вибору її рішення інтеграція із традиційною мережею здійснюється прозоро й без застосування яких-небудь додаткових технологій. Підключення до існуючої мережі рекомендується робити через два опорних маршрутизатори Cisco ISR 4451, розташованих у центральному офісі. Незважаючи на те що ці маршрутизатори перебувають під контролем APIC-EM, для взаємодії із традиційною частиною мережі використовуються стандартні механізми – протоколи канального рівня (залежно від типу ліній зв'язку) і протоколи комутації/маршрутизації відповідно до корпоративного стандарту.

У рішенні HP для реалізації гібридної інфраструктури SDN використовується стандартна функціональність протоколу OpenFlow, а саме інструкції NORMAL і FLOOD, які дозволяють після аналізу трафіку в таблицях OpenFlow на мережному пристрої передати його для наступної обробки в традиційний мережний стек протоколів, настроєних на цьому ж пристрої.

У цілому, завдяки гібридній архітектурі, можна поетапно впроваджувати рішення SDN в існуючих мережах, при цьому дані рішення будуть тісно інтегруватися й взаємодіяти з устаткуванням, що не підтримує SDN і протокол OpenFlow. Це, зокрема, дозволяє використовувати переваги, які надають SDN-додатка, без необхідності повної заміни всього устаткування в мережі.

3.3 Розробка функціональної схеми

Для реалізації системи побудови мережі на основі технології SDN необхідно створити імітаційну модель. Для організації моделі мережі за найпростішою схемою потрібен набір з 14 об'єктів (рисунок 3.2)

На рисунку 3.2 наведена функціональна схема розробленої системи, виходячи зі схеми зображеної на рисунку 3.1. З рисунка 3.2 бачимо, що для реалізації іммітаційної моделі встановлюється два хоста та два маршрутизатора, через які йде трафік. Зафарбована область позначає топологічні елементи мережі (канали й маршрутизатори).

На функціональній схемі вказані усі елементи, які необхідні для передачі пакета по протоколу ARTCP у мережі. До них відносяться:

– об’єкти протоколу ARTCP та CBR;

– хости;

– симплексні канали передачі даних;

– інтерфейси по яким відбувається зєднання між маршрутизаторами;

– маршрутизатори;

– комутуюче поле у маршруторі (таблиця маршрутизації).

У даній роботі при реалізації системи побудови мережі на основі технології SDN удосконалений протокол Adaptive Rate Transmission Control Protocol (ARTCP), який позичає деякі механізми від протоколу TCP. В ARTCP повністю переглянутий алгоритм керування потоком, що і відрізняє його від TCP. Разом з тим, запропонований протокол може забезпечувати сумісність із TCP.

ARTCP відрізняється від стандартного TCP тим, що сегменти відправляються в мережу не у вигляді сплеску в межах вікна, а розділені часовими проміжками, тривалість яких визначається поточним значенням швидкості. Швидкість потоку регулюється не розміром змінного вікна, а значенням швидкості, зміною якої здійснюється адаптація алгоритму відповідно до умов. Механізм ковзного вікна в ARTCP застосовується тільки для

запобігання переповнення буферів одержувача. На розмір вікна в ARTCP не накладено ніяких обмежень у жодному з режимів роботи. Вікно обмежене лише наявністю буферного простору в одержувача, тому для одержувача рекомендується встановлювати вікно на максимальний розмір.

У випадку, коли одержувач обмежений у буферному просторі, ARTCP буде поводитися як звичайний протокол TCP, обмежений вікном. Таким чином, протокол ARTCP сполучить у собі метод ковзного вікна для регулювання керування потоком від краю до краю й метод контролю швидкості для підстроювання під ПрЗд проміжних вузлів з'єднання.

Другою найважливішою відмінністю ARTCP є те, що як сигнал про стан перевантаження або наявності додаткових ресурсів у мережі використовуються темпоральні характеристики потоку – вимір шпаруватості потоку в одержувача й зміни часу RTT. Втрата пакета ніяк не відбивається на роботі ARTCP крім здійснення ретрансляції загубленого пакета механізмом корекції помилок. Як і в TCP об втрати пакета повідомляють дві можливих події: спрацьовування ТПП або послідовне одержання двох підтверджень тих самих даних.

Таким чином, у порівнянні зі своїм попередником, ARTCP має наступні переваги:

1. ARTCP не потрібно доводити мережа до стану перевантаження, щоб визначити доступну частку ПрЗд, тому виключені втрати пакетів пов'язані із цим процесом.

2. ARTCP істотно знижує вимоги до міжмережних пристроїв. По-перше, для нормального функціонування даного протоколу потрібен менший об'єм буферного простору, ніж для TCP, оскільки режим передачі є згладженим. По-друге, ARTCP не вимагає й не залежить від наявності яких або механізмів диспетчеризації або керування чергами, таких як RED або WFQ.

3. ARTCP не інтерпретує втрату пакета як ознаку перевантаження мережі, використовуючи замість цього темпоральні характеристики потоку. Тому ARTCP

повинен особливо ефективно працювати в системах бездротового зв'язку, там де використання TCP неефективно.

4. На відміну від TCP новий протокол не покладається цілком на потік підтверджень у зворотному напрямку для синхронізації процесу передачі. У зв'язку із цим можлива реалізація ARTCP з меншою частотою підтверджень, що не обмежувала б швидкість в асиметричних системах.

Аналіз робіт в області транспортних протоколів, дозволив укласти, що недоліки протоколу TCP досить істотні і є наслідком самого алгоритму, що лежить в основі протоколу TCP. Тому модифікація TCP без заміни його основних алгоритмів не може привести до істотного поліпшення характеристик протоколу.

Тому в цій роботі було вирішено створити вдосконалений алгоритм транспортного протоколу, що залишається, однак, повністю сумісним з архітектурою TCP/IP.

Для того щоб усунути недоліки, властиві TCP, необхідно було знайти спосіб одержання інформації про стан мережі, відмінний від застосування в цих цілях втрат сегментів. Найбільше добре на роль індикатора стану мережі підходять часові характеристики потоку: час RTT і міжсегментні інтервали. З використанням міжсегментних інтервалів можна також визначити частку ПрЗд каналу. Для цього потрібно запам'ятовувати міжсегментні інтервали потоку у відправника й вимірювати їх в одержувача. Порівняння значень інтервалів характеризує стан мережі, а мінімальне значення вимірюваних інтервалів в одержувача дозволяє визначити доступну частку ПрЗд.

Таким чином, виходить наступна схема: установка швидкості потоку відправником за допомогою ретельної диспетчеризації сегментів, вимір швидкості прибуття потоку в одержувача й передача цієї інформації відправникові разом з іншою контрольною інформацією. Різниця старого й нового значень швидкості відправлення потоку ARTCP на кожному кроці задається випадковою змінною, однак, при наявності сигналу про

перевантаження мережі ймовірність зниження швидкості перевищує ймовірність її збільшення на кожному новому кроці.

Нехай τ часовий інтервал між послідовними трансляціями пакетів. Задача функції диспетчеризації сегментів у тому, щоб затримувати відправлення чергового сегмента на час τ після початку передачі попереднього сегмента. Позначимо всі змінні, відносні до відправника індексом s, і r – відносні до одержувача. Отже, τ часовий інтервал між моментами початку відправлення в мережу сегмента i+1 і i-го, а τ_r – інтервал між послідовно прибулими до одержувача сегментами.

Нехай швидкість каналу зв'язку, безпосередньо до якого підключений відправник – R_{ls}, тоді час який потрібен на відправлення одного сегмента (з моменту початку передачі до моменту її закінчення) $t_{ls} = S / R_{ls}$, де s – розмір переданого сегмента.

Очевидно, що максимально можлива швидкість потоку:

$$R^{\max}_s = R_{ls} = S / \tau^{\min}_s. \qquad (3.1)$$

Мінімальне значення міжсегментного інтервалу в цьому випадку буде $\tau^{\min}_s = t_{ls}$, коли пакети відправляються в мережу без затримок з максимальною швидкістю канального рівня. Шляхом зміни τ у межах $[\tau^{\min}, \infty)$, ARTCP може контролювати швидкість потоку в межах $[R_{ls}, 0)$. Ілюстрація принципу керування швидкістю наведена на рисунку 3.1.

Затримка відправлення готових сегментів виробляється за допомогою системного таймера. Наприклад, для повного використання ПрЗд каналу в 512 Кб/с при розмірі сегмента в 1000 байт кожний сегмент необхідно відправляти із затримкою τ =0.015625 с, щоб швидкість потоку склала 64 пакета/с.

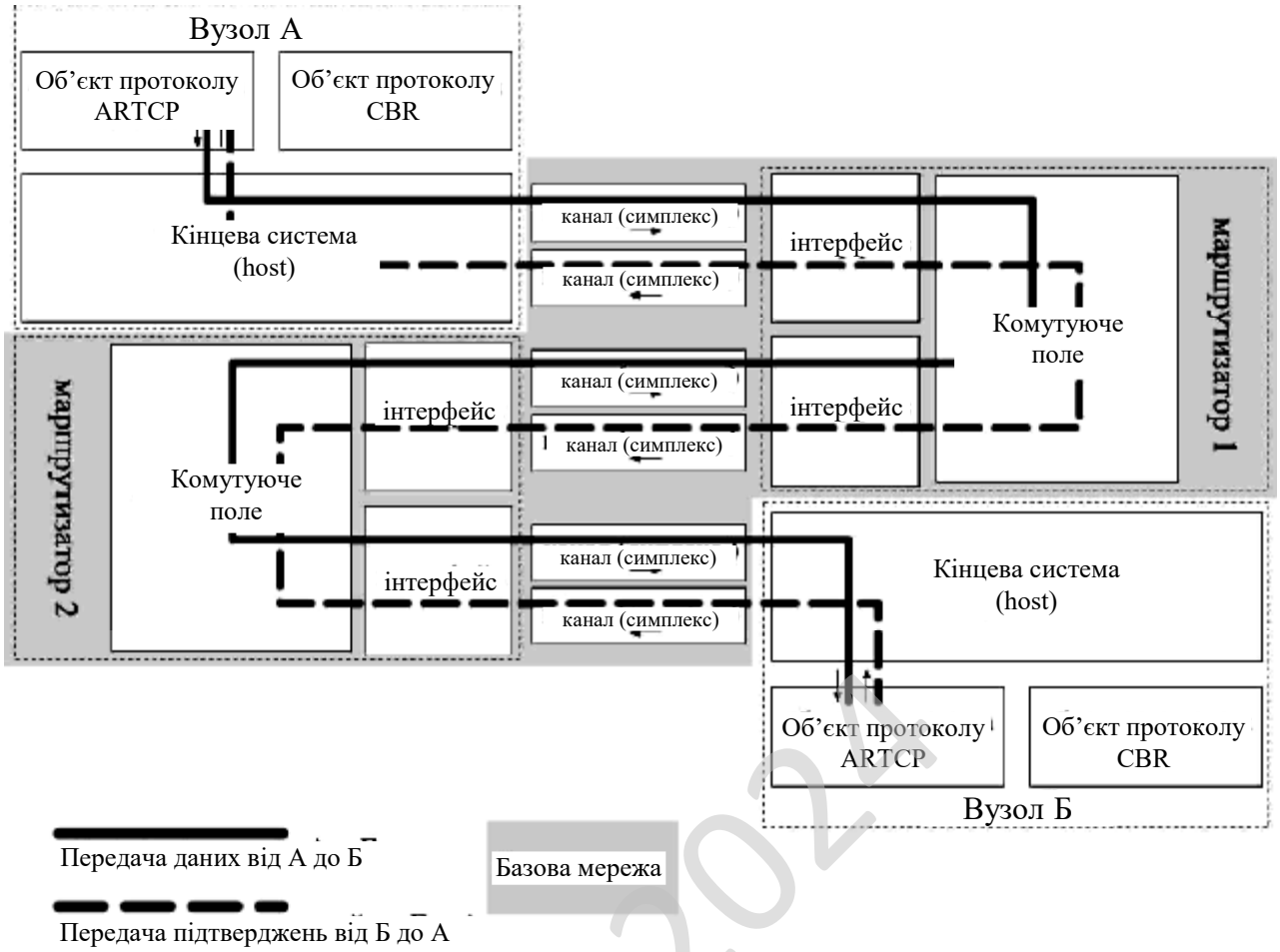

Рисунок 3.2 – Функціональна схема системи

Формат повідомлень використовуваних ARTCP може в точності збігатися з форматом пакета TCP. Стандарт TCP [4] передбачає наявність додаткових полів у заголовку сегмента між стандартним заголовком і полем даних. ARTCP може передавати додаткову інформацію в цих полях, що буде гарантувати сумісність із TCP. Усього протокол ARTCP вимагає використання лише двох нових полів: значення попереднього порядкового номера "PS" у напрямку від відправника до одержувача й значення шпаруватості "TI" у напрямку від одержувача до відправника. Значення "TI" можна передавати у вигляді опції часової мітки [6], а значення "TI" вимагає поля, що дозволяє помістити порядковий номер сегмента.

Диспетчер сегментів відправляє сегменти на лінію через строго задані міжсегментні часові інтервали. Значення інтервалів визначаються швидкістю відправлення потоку, що задається функцією адаптації.

Очевидно, що швидкість прийому потоку одержувачем не може бути вище швидкості обслуговування потоку на ділянці з найменшої ПрЗд, через яку проходить з'єднання. Таким чином, знаючи швидкість прибуття потоку до одержувача, можна визначити доступну пропускну здатність мережі. Для коректного виміру швидкості необхідно не враховувати випавші з потоку, тобто загублені сегменти, а також сегменти, що доставляються мережею в зміненому порядку. Для виконання цієї умови в поле "PS" кожного сегмента, що відправляється, записується порядковий номер (або зсув) від попереднього сегмента.

Одержавши сегмент *i*, одержувач обчислює різницю поточного часу й часу прибуття попереднього (*j*) сегмента τ_R і у випадку, якщо поле "PS" *i-го* сегмента містить значення *j*, поміщає $R_r = s/\tau_R$ у поле "TI" підтвердження наступного в протилежному напрямку (рисунок 3.1). Одержувач витягає значення поля "TI" з одержуваних підтверджень і використовує його для керування швидкістю передачі.

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

3.4 Розробка діаграми процесів

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. Діаграма процесів розробленої системи зображена на рисунку 3.3. При детальному її розгляді можна побачити як саме відбувається взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

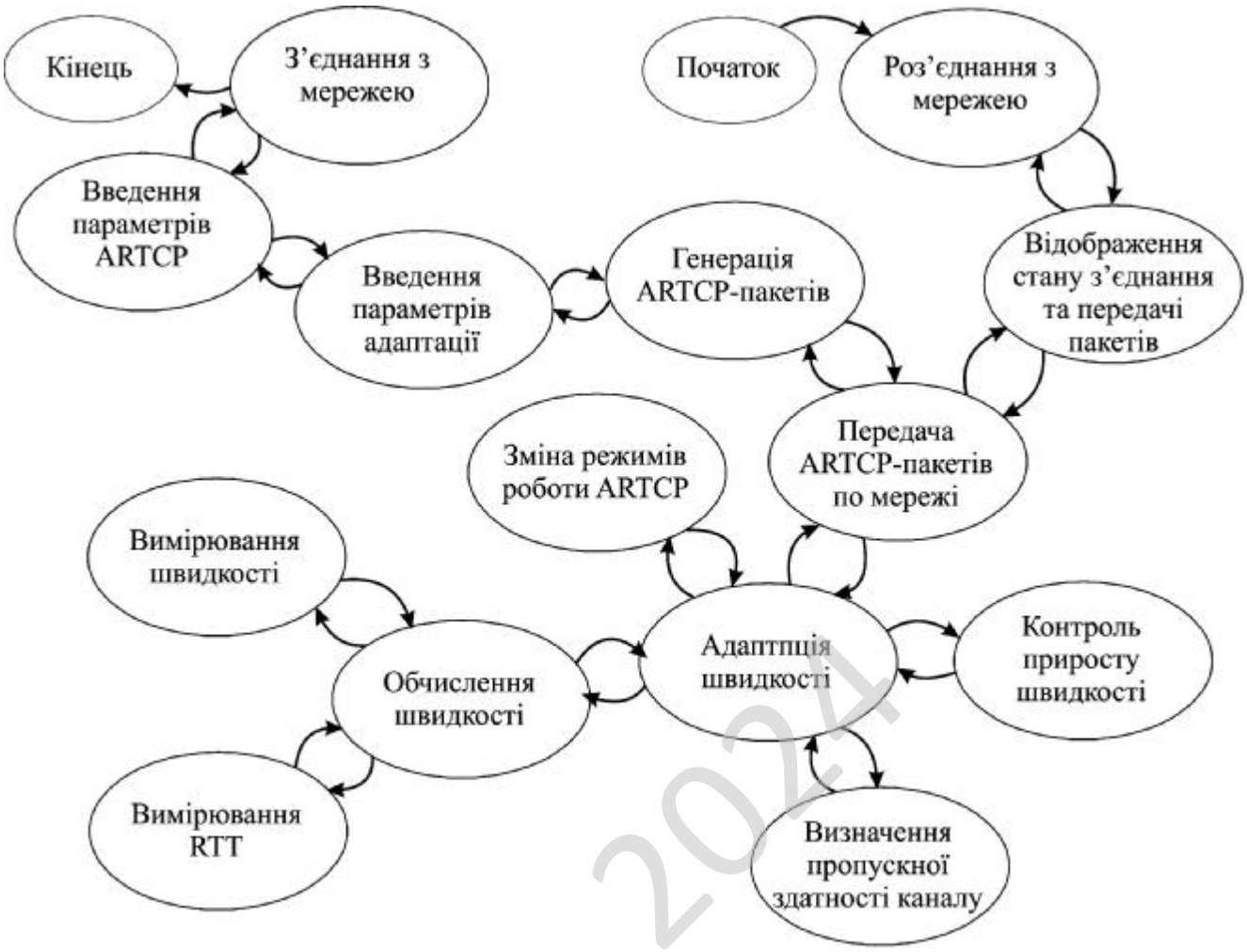

Рисунок 3.3 – Діаграма взаємодії процесів

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

Блок-схеми є основою ПЗ. Тому від точності і детальності проробки 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, також те, що при розробці програми слід надати особливу увагу модулю побудови мережі на основі технології SDN.

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

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

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

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

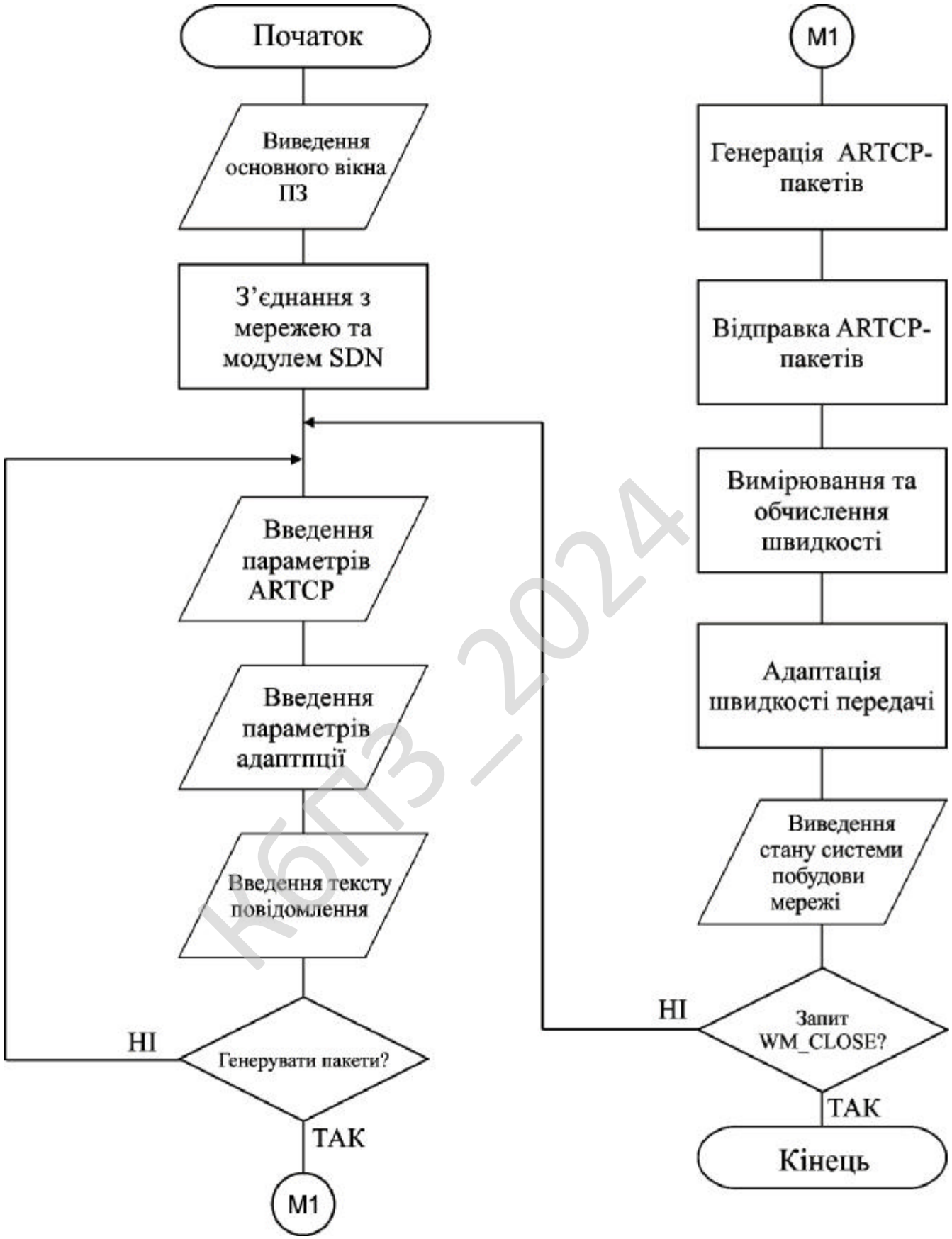

Рисунок 4.1 – Блок-схема основної програми

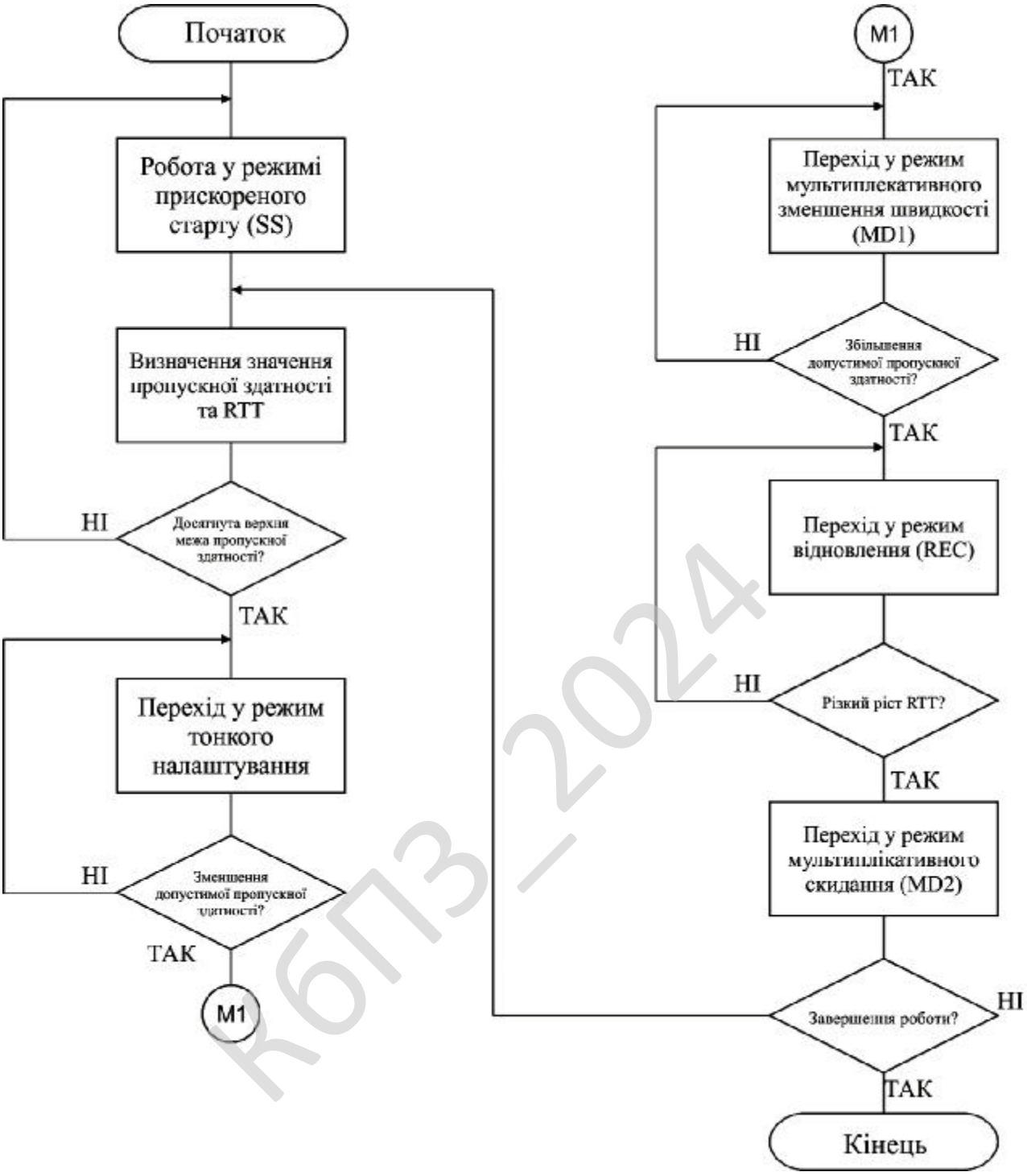

Рисунок 4.2 – Блок-схема роботи підпрограми

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

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

Розглянемо використані технології та їх основні компоненти що підтверджують правильність використаних проектних рішень.

Redmine – вільне серверне ПЗ для управління проектами та відстежування помилок. До системи входить календар-планувальник та діаграми Ганта для візуального представлення ходу робіт за проектом та строків виконання. Redmine написано на мові Ruby і є ПЗ розробленим з використанням відомого веб-фреймворку Ruby on Rails, що означає легкість в розгортанні системи та її адаптації під конкретні вимоги. Для кожного проекту можна вести свої вікі та форуми.

Функціональні можливості:

– Ведення декількох проектів.

– Гнучка система доступу з використанням ролей.

– Система відстеження помилок.

– Діаграми Ганта та календар.

– Ведення новин проекту, документів та управління файлами.

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

– Власна Wiki для кожного проекту.

– Форуми для кожного проекту.

– Облік часових витрат.

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

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

– Створення записів про помилки на основі отриманих листів

– Підтримка LDAP автентифікації.

– Можливість самореєстрації нових користувачів.

– Багатомовний інтерфейс (у тому числі українська мова).

– Підтримка СКБД: MySQL, PostgreSQL, SQLite.

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

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

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

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

Система відстеження помилок Багтрекер – прикладна програма для допомоги розробникам програмного забезпечення (програмістам, тестувальникам тощо) враховувати і контролювати помилки, знайдені у програмах, питання щодо

функціональності, рішення та оновлення, побажання користувачів, а також стежити за процесом їх виконання.

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

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

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

Отже, система відстеження помилок у найпростішому варіанті – це процес, що включає в себе виявлення помилки, її опис, виправлення і перевірку цього виправлення, тобто процес «стеження» за багом протягом всього як його життєвого циклу, так і життєвого циклу розробки в цілому.

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

– номер (ідентифікатор) дефекту;

– хто повідомив про дефект;

– дата і час виявлення дефекту;

– версія продукту, в якій виявлено дефект;

– серйозність (критичність) дефекту та пріоритет рішення;

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

– відповідальний за усунення дефекту;

– обговорення можливих рішень та їх наслідків;

– поточний стан виправлення дефекту;

– версії продукту, в якій дефект виправлений.

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

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

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

Типовий життєвий цикл дефекту:

1. Новий – дефект зареєстрований тестувальником.

2. Призначений – призначений відповідальний за виправлення дефекту.

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

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

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

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

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

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

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

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

У корпоративному середовищі, система відстеження помилок може

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

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеГрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15-30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані. Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner'а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting). Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?;

– Що буде зроблено з моменту поточної наради до наступної?;

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3ма годинами.

Основою системи була архітектура SDN, яка включала три головні компоненти:

– Контролер (Control Plane).

– Інфраструктура передачі даних (Data Plane).

– Програми управління мережею (Application Plane).

Основною задачею системи було відокремлення керування мережею від інфраструктури передачі даних для спрощення адміністрування, збільшення гнучкості та масштабованості.

Контролер виконував роль центрального елемента, що керував усіма маршрутизаторами та комутаторами мережі. Його функцією було прийняття рішень про передачу трафіку, управління мережею та контроль стану обладнання.

Інфраструктура передачі даних включала комутатори, маршрутизатори та інші мережеві елементи, які безпосередньо виконували пересилку даних. Вони підпорядковувались командам контролера.

Програмні інтерфейси для управління мережею дозволяли адміністраторам здійснювати налаштування політик маршрутизації, безпеки та управління трафіком.

Модулі системи

1. SDN контролер.

Контролер був реалізований на Python з використанням модуля ryu. Цей модуль надавав API для роботи з комутаторами та управління мережевими потоками. Контролер керував потоками даних через OpenFlow протокол. Базова конфігурація контролера включала обробку запитів на підключення, контроль таблиць маршрутизації та управління трафіком.

2. Комутатори та маршрутизатори (Data Plane).

Всі комутатори і маршрутизатори працювали за допомогою протоколу OpenFlow, що забезпечував можливість прямого управління контролером. При побудові мережі використовувалися фізичні комутатори з підтримкою OpenFlow, а також віртуальні комутатори (наприклад, OVS – Open vSwitch).

3. Програма управління мережею (Application Plane).

Створювались модулі для динамічної маршрутизації трафіку та забезпечення високої пропускної здатності. Програмне забезпечення контролювало стан мережі в реальному часі, збирало статистику про пропускну здатність каналів та здійснювало автоматичне балансування навантаження.

Архітектура системи

1. Система базувалася на централізованому контролері, який отримував інформацію від мережевих пристроїв та приймав рішення щодо управління мережею.

2. Всі пристрої передавали свої таблиці стану контролеру, який обробляв їх і приймав рішення про налаштування правил пересилання трафіку.

3. Програмна частина контролера була відповідальною за обробку запитів від комутаторів, маршрутизаторів та інших елементів, а також за налаштування правил маршрутизації.

4. Комутатори та маршрутизатори виконували команди, що надходили від контролера, і передавали дані між мережевими вузлами.

Код системи побудови мережі SDN

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
from ryu.lib.packet import ethernet
from ryu.lib.packet import ether_types

class SimpleSwitch13(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

    def __init__(self, *args, **kwargs):
        super(SimpleSwitch13, self).__init__(*args, **kwargs)
        self.mac_to_port = {}

    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        in_port = msg.match['in_port']
        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]
        dst = eth.dst
```

```
        src = eth.src
        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})
self.logger.info("packet in %s %s %s %s", dpid, src, dst, in_port)
# Learn a mac address to avoid FLOOD next time.
        self.mac_to_port[dpid][src] = in_port
        if dst in self.mac_to_port[dpid]:
          out_port = self.mac_to_port[dpid][dst]
        else:
            out_port = ofproto.OFPP_FLOOD
        actions = [parser.OFPActionOutput(out_port)]
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
            self.add_flow(datapath, match, actions)
        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data
        out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
        in_port=in_port, actions=actions, data=data)
        datapath.send_msg(out)
    def add_flow(self, datapath, match, actions):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                actions)]
        mod = parser.OFPFlowMod(datapath=datapath, priority=1,
                                  match=match, instructions=inst)
        datapath.send_msg(mod)
```

Додаткові функції дозволять реалізувати:

– Управління балансуванням трафіку.

– Моніторинг стану мережі.

– Динамічне оновлення таблиць маршрутизації.

Додавання функцій балансування трафіку:

```
from ryu.base import app_manager
from ryu.controller import ofp_event
from ryu.controller.handler import MAIN_DISPATCHER, set_ev_cls
from ryu.ofproto import ofproto_v1_3
from ryu.lib.packet import packet
```

```
from ryu.lib.packet import ethernet, ipv4
from ryu.lib.packet import ether_types

class LoadBalancer(app_manager.RyuApp):
    OFP_VERSIONS = [ofproto_v1_3.OFP_VERSION]

def __init__(self, *args, **kwargs):
        super(LoadBalancer, self).__init__(*args, **kwargs)
        self.mac_to_port = {}
        self.server_pool = ['10.0.0.2', '10.0.0.3', '10.0.0.4']
# IP адреси серверів
        self.current_server = 0
    @set_ev_cls(ofp_event.EventOFPPacketIn, MAIN_DISPATCHER)
    def _packet_in_handler(self, ev):
        msg = ev.msg
        datapath = msg.datapath
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        in_port = msg.match['in_port']
        pkt = packet.Packet(msg.data)
        eth = pkt.get_protocols(ethernet.ethernet)[0]
        if eth.ethertype == ether_types.ETH_TYPE_LLDP:
# LLDP пакети ігнорували
            return
        if eth.ethertype == ether_types.ETH_TYPE_IP:
            ip_header = pkt.get_protocols(ipv4.ipv4)[0]
            dst_ip = ip_header.dst
            if dst_ip in self.server_pool:
# Перемикання на інший сервер для балансування
new_dst_ip = self.server_pool[self.current_server]
self.current_server = (self.current_server + 1) % len(self.server_pool)
self.logger.info(f"Балансування трафіку на сервер {new_dst_ip}")
# Оновлення IP в таблиці маршрутизації
        ip_header.dst = new_dst_ip

        dst = eth.dst
        src = eth.src
        dpid = datapath.id
        self.mac_to_port.setdefault(dpid, {})
        self.mac_to_port[dpid][src] = in_port
        if dst in self.mac_to_port[dpid]:
            out_port = self.mac_to_port[dpid][dst]
```

```
        else:
            out_port = ofproto.OFPP_FLOOD
        actions = [parser.OFPActionOutput(out_port)]
        if out_port != ofproto.OFPP_FLOOD:
            match = parser.OFPMatch(in_port=in_port, eth_dst=dst)
            self.add_flow(datapath, match, actions)
        data = None
        if msg.buffer_id == ofproto.OFP_NO_BUFFER:
            data = msg.data
        out = parser.OFPPacketOut(datapath=datapath, buffer_id=msg.buffer_id,
                                  in_port=in_port, actions=actions, data=data)
        datapath.send_msg(out)

    def add_flow(self, datapath, match, actions):
        ofproto = datapath.ofproto
        parser = datapath.ofproto_parser
        inst = [parser.OFPInstructionActions(ofproto.OFPIT_APPLY_ACTIONS,
                                             actions)]
        mod = parser.OFPFlowMod(datapath=datapath, priority=1,
                                match=match, instructions=inst)
        datapath.send_msg(mod)
```

Обґрунтування проектних рішень

1. Використання протоколу OpenFlow дозволяло забезпечити гнучкість та простоту управління мережею за рахунок централізованого контролера.

2. Контролер базувався на модулі ryu, який є стандартом у сфері SDN і дозволяв легко інтегрувати програмне забезпечення з існуючим мережевим обладнанням.

3. Вибір архітектури з централізованим контролером забезпечував можливість швидкої масштабованості мережі, а також спрощення обслуговування та оновлення конфігурацій.

4. За допомогою створених модулів маршрутизації та управління трафіком вдалося досягти оптимального розподілу навантаження в мережі та підвищення загальної продуктивності мережі

5. Додавання функцій балансування трафіку покращило продуктивність мережі, дозволивши рівномірно розподіляти запити до серверів, що підтверджувалося зниженням навантаження на кожен окремий сервер.

6. Система моніторингу надавала статистичні дані про стан портів та мережі в реальному часі, що дозволяло оперативно виявляти проблеми та реагувати на збої.

7. Динамічне оновлення правил маршрутизації дозволило мережі бути гнучкою до змін у навантаженні та топології, що збільшило її стабільність та ефективність.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8

бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;
– Лінійне перетворення π;
– Байтова перестановка τ;
– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо розроблене ПЗ системи побудови мережі на основі технології SDN яке зображено на рисунку 5.1. Процес розрахунків здійснюється в консольному додатку з передачею отриманих результатів у користувацький інтерфейс. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

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

– Функції представлені у графічному вигляді (іконки).

– Розділ генерація пакетів.

– Розділ параметрів.

– Розділ обробки даних.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Функціональних кнопок ПЗ.

Вигаданий замовник – велика компанія з територіально розподіленою структурою. Один із сегментів своєї мережі передачі даних, що побудована на основі традиційної архітектури, він вирішив перевести на технологію SDN. Замовник розраховує, що впровадження SDN дозволить йому автоматизувати ряд важливих процедур, пов'язаних з експлуатацією мережі. У їхнє число входять керування списками контролю доступу, блокування трафіку певних додатків, користувачів і їхніх груп відповідно до прийнятої політики безпеки, а також налаштування алгоритмів пріоритизації й забезпечення якості обслуговування (QoS), конфігурування нових мережних пристроїв.

Розроблена програма має дуже простий і інтуїтивно зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним

середовищем Windows без особливих складностей освоїть і цю програму, оскільки її інтерфейс інтуїтивно зрозумілий.

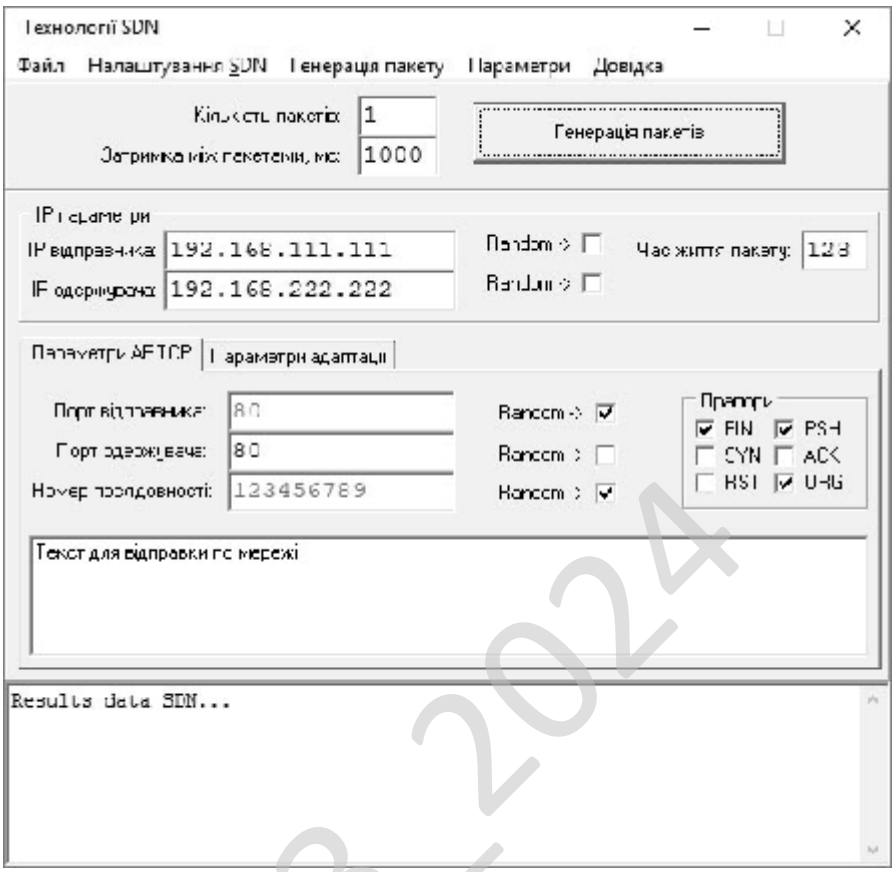

Рисунок 5.1 – Головне вікно ПЗ

Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

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

Загалом процес розгортання складається з кількох взаємопов'язаних дій із можливими переходами між ними. Ця активність може відбуватися як з боку виробника так і з боку споживача. Оскільки кожна програмна система є унікальною, то усі процеси та процедури під час розгортання важко передбачити. Тому, "розгортання" можна трактувати як загальний процес відповідно до певних вимог та характеристик. Розгортання може здійснюватись програмістом і в процесі розробки програмного забезпечення.

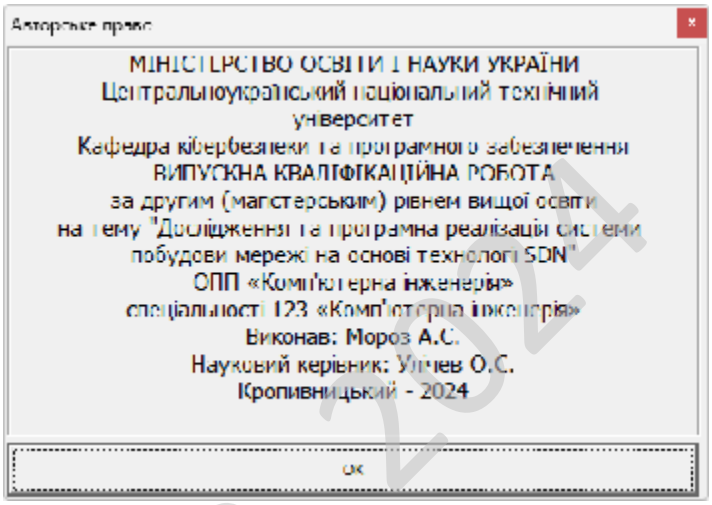

Рисунок 5.2 – Авторське право

До діяльностей пов'язаних із розгортанням програмного забезпечення відносять:

– Випуск.

– Встановлення та активація.

– Деактивація.

– Адаптація.

– Обновлення.

– Вмонтування.

– Відстежування версій.

– Видалення.

– Вилучення з обігу.

При впровадженні програмного забезпечення потрібно урахувати наступні дії:

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

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

– Виконання робіт з загальної стандартизації існуючої діяльності організації, коли виділяються кращі практики виконання процедур і включаються в ІТ рішення за принципом найбільшої корисності для більшості учасників. Відсоток таких процедур щодо загального обсягу автоматизації може бути невеликий, але це надає процесу побудови рішення вагу в організації за рахунок збільшення його необхідності.

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

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

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

Проводилось тестування 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 побудови мережі на основі технології SDN.

Метою розробки є дослідження та програмна реалізація системи побудови мережі на основі технології SDN.

Об'єктом дослідження є процес побудови мережі на основі технології SDN.

Предметом дослідження є методи побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN.

– Розроблено вітчизняний продукт побудови мережі на основі технології SDN, 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 побудови мережі на основі технології SDN (Software-Defined Networking) можуть бути цікавими різним категоріям фахівців і організацій (рис. 7.1).

Інженери та архітектори мереж –	•їм будуть корисні нові способи оптимізації мережевої інфраструктури, спрощення управління та підвищення ефективності роботи мережі.
Фахівці з інформаційної безпеки –	•SDN дозволяє впроваджувати політики безпеки на програмному рівні, що допомагає швидше реагувати на загрози та підвищити контроль над мережевими потоками.
Інтернет-провайдери та телекомунікаційні компанії –	•їм можуть бути цікаві способи спрощення управління мережею, зменшення витрат та швидка адаптація мережі до нових вимог клієнтів.
Розробники програмного забезпечення –	•такі дослідження надають нові можливості для розробки програмних продуктів, які можуть використовувати переваги SDN у своїй роботі.
Навчальні заклади та студенти –	•дослідження і прикладна реалізація систем SDN будуть корисними для вивчення нових концепцій, підготовки курсів та лабораторних робіт.
Великі компанії та підприємства –	•особливо ті, що використовують складні мережі, оскільки SDN дозволяє легше масштабувати і керувати такими мережами.
Дослідники та інженери в галузі інноваційних технологій –	•вони можуть використовувати нові ідеї з SDN для створення нових продуктів або вдосконалення наявних.

Рисунок 7.1 – Цільова аудиторія

Впровадження SDN пропонує гнучкість і підвищену ефективність мережевих ресурсів, що робить ці результати цінними для широкого спектру професіоналів, які працюють з сучасними мережевими технологіями.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Для оцінки привабливості програмної реалізації системи побудови мережі на основі технології SDN можна використовувати метод експертних оцінок, наприклад, за допомогою методу ранжування або методу попарного порівняння. Матимем наступний порядок виконання експертної оцінки.

Спочатку визначимо критерії, які експерти будуть використовувати для оцінки привабливості (рис. 7.2).

Рисунок 7.2 – Критерії експертної оцінки

Залучаємо групу експертів у сфері мережевих технологій, програмної інженерії та інформаційної безпеки (наприклад, 5-7 осіб). Експерти присвоюють

ваги кожному критерію за важливістю, наприклад, за 10-бальною шкалою. Далі середнє значення за кожним критерієм можна обчислити.

Таблиця 7.1 – Зведені результати експертних оцінок

Критерії	**Експерт 1**	**Експерт 2**	**Експерт 3**	**Середній бал**
Ефективність управління	8	7	9	8
Безпека	9	8	9	8.67
Продуктивність	7	8	8	7.67
Скалабельність	8	7	8	7.67
Вартість реалізації	6	5	7	6
Сумісність	7	8	7	7.33

Експерти оцінюють привабливість SDN за кожним критерієм, також за 10-бальною шкалою, виходячи з можливостей та функціональних характеристик реалізації SDN.

Таблиця 7.2 – Зведені результати експертних оцінок

Критерій	**Вага**	**Оцінка реалізації SDN**	**Продукт оцінки (вага * оцінка)**
Ефективність управління	8	9	72
Безпека	8.67	8	69.36
Продуктивність	7.67	8	61.36
Скалабельність	7.67	7	53.69
Вартість реалізації	6	7	42
Сумісність	7.33	8	58.64

Сума всіх продуктів ваги на оцінки дає загальну оцінку привабливості SDN: 72+69.36+61.36+53.69+42+58.64=357.0572 + 69.36 + 61.36 + 53.69 + 42 + 58.64 = 357.0572+69.36+61.36+53.69+42+58.64=357.05. Результати можуть бути

інтерпретовані наступним чином. Отримана оцінка перевищує певний поріг (наприклад, 300), то програмна реалізація SDN вважається привабливою для впровадження. За потреби можна порівняти з іншими альтернативами для обґрунтованого вибору. Такий підхід дозволяє кількісно оцінити привабливість проекту SDN на основі експертних думок та порівняти його з іншими рішеннями.

7.3 Вибір методу оцінки вартості ПЗ

Для оцінки вартості програмної реалізації системи побудови мережі на основі технології SDN доцільно розглянути 2 наступні методи: порівняння з аналогами та метод експертних оцінок. Метод аналогій базується на порівнянні вартості розробки з аналогічними проєктами, де вже використовували SDN. Метод підходить, якщо є доступ до інформації про вартість та специфіку подібних проєктів. Аналіз аналогій дає приблизну вартість та дозволяє оцінити, наскільки масштабним і ресурсомістким є проєкт. Метод експертних оцінок – це залучення експертів до прийняття рішення. Залучення експертів дозволяє оцінити вартість, враховуючи досвід і практику в сфері SDN. Експерти можуть оцінити загальні витрати на розробку, інфраструктуру, ліцензії, навчання персоналу та обслуговування. Цей метод може дати обґрунтовану оцінку вартості, якщо у проєкті є певні невизначеності або ризики. Це дасть детальніший розрахунок витрат, що дозволяє врахувати особливості кожного компоненту системи та можливі ризики, які можуть виникнути під час реалізації проєкту.

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

Впровадження системи побудови мережі на основі технології SDN може підвищити економічну ефективність компанії за рахунок факторів наведених на рисунку 7.3.

Рисунок 7.3 – Економічна ефективність від реалізації проєкту для клієнта

Загальна економічна ефективність включає зменшення витрат на обладнання, скорочення операційних витрат, оптимізацію ресурсів, зменшення втрат через простої, додатковий дохід від нових послуг.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Для успішного просування проєкту програмної реалізації системи побудови мережі на основі технології SDN можна використовувати алгоритм наведений на рисунку 7.4.

Рисунок 7.4 – Алгоритм просування проєкту

Цей алгоритм спрямований на поступове привернення уваги цільової аудиторії, підвищення її обізнаності про переваги SDN та активне залучення потенційних клієнтів через демонстрацію економічної та функціональної ефективності проєкту.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для оптимізації каналів збуту та шляхів реалізації проєкту програмної реалізації системи мережі на основі технології SDN, варто зосередитися на стратегіях запропонованих нижче:

– розширення каналів дистрибуції через партнерства;

– модель SaaS для оптимізації збуту;

– автоматизація продажів і впровадження каналів електронної комерції;

– ліцензування та білінг через модель на основі підписок;

– акцент на технічну підтримку та навчання через інтеграторів;

– прямі продажі великим корпоративним клієнтам;

– запуск партнерської програми для залучення реселерів;

– пропозиція програмного забезпечення з відкритим вихідним кодом (Freemium модель);

– інвестування в пошукову оптимізацію (SEO) та контент-маркетинг.

Кожна з цих стратегій допоможе оптимізувати канали збуту, розширити аудиторію та підвищити прибутковість проєкту SDN. Обираючи правильне поєднання з цих методів, ви зможете знизити витрати, збільшити доходи та охопити різноманітні сегменти ринку.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключові фактори успіху проєкту програмної реалізації системи побудови мережі на основі технології SDN (Software-Defined Networking) включають

технічні, операційні, бізнесові та маркетингові аспекти, які забезпечують ефективне впровадження, використання та комерціалізацію продукту (рис. 7.5).

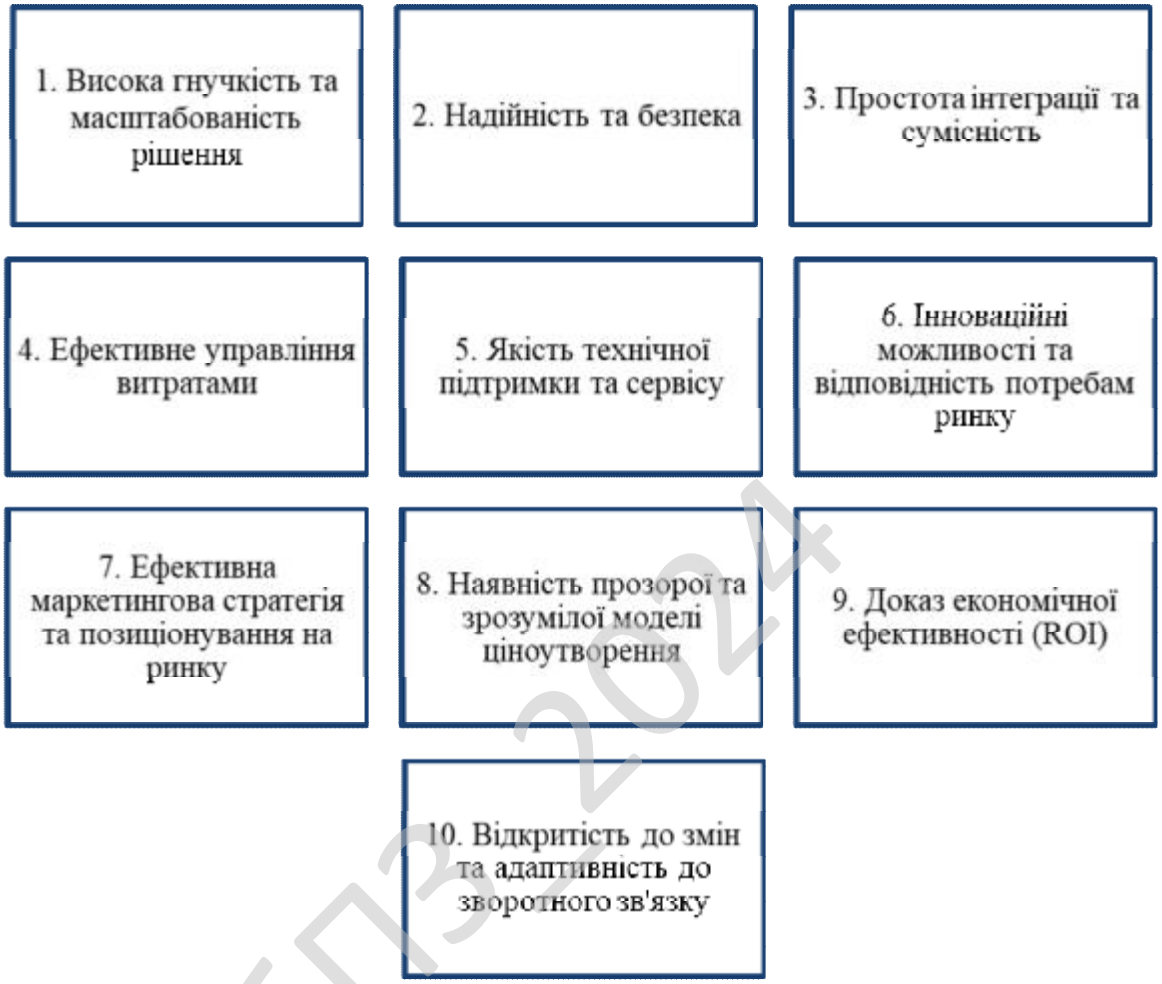

Рисунок 7.5 – Ключові фактори успіху проєкту

Дотримання цих факторів допоможе підвищити конкурентоспроможність проєкту на основі SDN, забезпечити лояльність клієнтів та сприяти успішному впровадженню технології.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Аналізуючи умови працівників ІТ-сфери, на перший погляд, може здатися, що працівники сфери інформаційних технологій не підпадають під ризи на виробництві, та якщо більш глибоко розглянути умови і специфіку праці фахівців сфері it-індустрії, можна виявити ряд факторів які будуть мати негативний влив як на стан охорони праці, так і на самого it-фахівця зокрема. Сюди можна віднести як невідповідність освітлення, так і високий рівень шуму, що негативно позначатимуться як на емоційному так і на фізичному стані фахівця, призводитимуть до зниження ефективності праці та виробничих травм. Також, важливим моментом охорони праці ІТ-фахівця є врахування його психологічних можливостей (швидкість реакції, особливості пам'яті та уваги, емоційний стан тощо). Для того, щоб забезпечити ефективну роботу it-фахівця, потрібно враховувати та максимально компенсувати такі негативні фактори як: надмірне нервово-емоційне навантаження, довготривалі статичні перевантаження, обмежена рухова активність. Всі ці чинники призводить до різноманітних відхилень у стані здоров’я, зокрема до перевтоми, зниження фізичної та розумової працездатності, неврозів, захворювань серцево-судинної системи тощо. Метою даного розділу є огляд конкретних умов праці спеціаліста у сфері it-індустрії. Завданнями для даного розділу є: аналіз умов праці на робочому місці фахівця it-індустрії, розробка конкретних рекомендацій щодо покрашення умов праці фахівців it-індустрії, огляд пожежної безпеки на ІТ-підприємстві та розрахунок системи загального штучного освітлення виробничого приміщення де працюють ІТ – фахівці.

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	3
Довжина	4,6
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,9
Об'єм, V	$м^3$	не менше 20.0	20,7

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

У зазначеному приміщенні працюють двоє 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» [2], 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» 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.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину.

Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Вологість,%	Швидкість повітря, м/с	Температура,°С	Вологість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-24	40-55	0,12
Тепла	23-25	50-70	0,1	24-25	50-65	0,9

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер Xerox WorkCentre 3025BI (3025VBI), електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми україни "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи

приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.3 Розробка заходів з умов поліпшення охорони праці

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

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

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних 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) [9].

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.

8.4 Розрахункова частина

Питання охорони праці та правила безпеки при роботі з офісною, комп'ютерною та мережевою технікою розглянуті у працях вітчизняних вчених [11].

Запорукою безпечної роботи є 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 грунту – суглинок;

– тип заземлювача – труба, діаметром d=0.045 м і довжиною l =2.2 м; – конструкція заземлювача – розташування заземлювачів по контуру. Розрахунок проводимо за стандартною методикою [7].

Визначимо розрахунковий опір землі:

$$\rho_{рз} = \phi \cdot \rho_з$$

де ϕ – коефіцієнт сезонності, що враховує коливання питомого опору при зміні вологості ґрунту протягом року; при використанні заземлювача довжиною l = 2.2 м при глибині закладання від вершини h = 0.6 м ϕ = 1.1 для четвертої

кліматичної зони.

Схема розташування заземлювачів показана на рисунку 8.1.

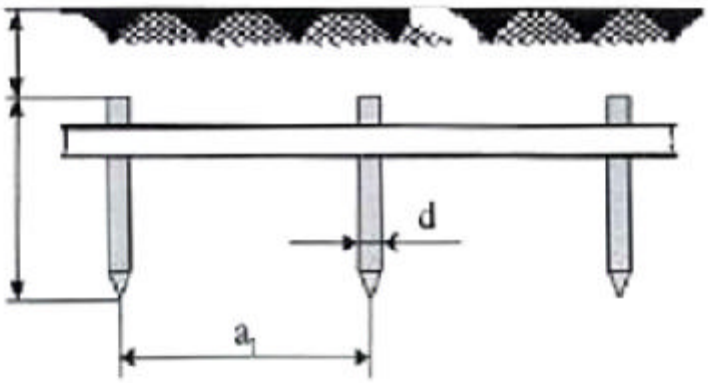

Рисунок 8.1 – Схема розташування заземлювачів

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 = 2.2 м);

d = 0.045 м – діаметр заземлювача при U<1 кВ та при S<100 кВА;

t – відстань від поверхні до середини заземлювача:

$$t = h + l/2 = 0.6 + 2.2/2 = 1.7 \text{ м.}$$

$$R_в = \frac{110}{2 \cdot 3.14 \cdot 2.2}\left(\ln\left(\frac{2 \cdot 2.2}{0.045}\right) + 0.5 \cdot \ln\left(\frac{4 \cdot 1.7 + 2.2}{4 \cdot 1.7 - 2.2}\right)\right) = 38.9 \text{ Ом}$$

Визначаємо потрібну кількість заземлювачів:

$$n' = \frac{R_в}{R_{зн}} = \frac{38.9}{10} = 3.9 \approx 4 \text{ шт.}$$

Коефіцієнт використання вертикальних заземлювачів враховує ефект екранування. При вибраному значенні k = a/l, де a – відстань між вертикальними заземлювачами, м; k = 1 при a = 1.8 м. Коефіцієнт використання вертикального

заземлювача за довідковими даними дорівнює $\eta_В = 0,6$.

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

$$n = \frac{R_в}{R_{зн} \cdot \eta_в} = \frac{38.9}{10 \cdot 0.6} = 6.48 \approx 7 \; шт.$$

Довжина горизонтального заземлювача, необхідна для розміщення вертикальних заземлювачів по контуру

$$L = a \cdot n = 1.8 \cdot 7 = 12.6 \text{ м}$$

Опір горизонтального заземлювача $R_Г$, Ом, прокладеного на глибині h = 0.6 м від поверхні землі буде

$$R_г = \frac{R_{рз}}{2 \cdot 3.14 \cdot L} \cdot \ln \frac{2 \cdot L^2}{b \cdot h} = \frac{110}{2 \cdot 3.14 \cdot 12.6} \cdot \ln \frac{2 \cdot 12.6^2}{0.06 \cdot 0.6} = 13.57 \text{ Ом}$$

де b = 0.06 м – ширина сталевої смуги, з якої виготовлений заземлювач.

Обчислюємо загальний опір:

$$R_з = \frac{R_в \cdot R_г}{n \cdot R_г \cdot \eta_в + R_в \cdot \eta_г} = \frac{38.9 \cdot 13.57}{6 \cdot 13.57 \cdot 0.6 + 38.9 \cdot 0.34} = 7.93 \text{ Ом}$$

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

Маємо 7.93 < 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 побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем побудови мережі на основі технології SDN.

– Досліджена система побудови мережі на основі технології SDN.

– На основі отриманих результатів досліджень створена програмна реалізація системи побудови мережі на основі технології SDN.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання побудови мережі на основі технології SDN.

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 CRYPTON.

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. Мороз А.С. Дослідження та програмна реалізація системи побудови мережі на основі технології SDN // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

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

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

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

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

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

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

8. 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.

9. 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.

10. 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.

11. 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.

12. 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.

13. 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.

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

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. 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.

21. 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.

22. 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.

23. 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.

24. 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.

25. 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.

26. 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.

27. 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.

28. 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.

29. 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.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. 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

38. 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.

39. 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.

40. 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.

41. 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.

42. 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.

43. 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.

44. 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).

45. 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

46. 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.

47. 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.

48. 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.

49. 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.

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

51. 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.

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN.

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 побудови мережі на основі технології SDN;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище 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 – 97 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

_____ Улічев О.С.

Дослідження та програмна реалізація системи побудови мережі на основі технології SDN

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2024 року

Основна програма

```
# Імпорт необхідних бібліотек
import requests
from flask import Flask, request, jsonify

# Створення Flask додатку для роботи з контролером SDN
app = Flask(__name__)

# Створення класу для SDN контролера
class SDNController:
    def __init__(self):
        self.switches = {}
        self.flows = []
        self.devices = []

    # Додавання нового комутатора до мережі
    def add_switch(self, switch_id):
        self.switches[switch_id] = []

    # Додавання нового пристрою до мережі
    def add_device(self, device_id, switch_id):
        if switch_id in self.switches:
            self.devices.append({'device_id': device_id, 'switch_id':
switch_id})
        else:
            raise Exception("Switch does not exist")

    # Додавання нового правила потоку до комутатора
    def add_flow(self, switch_id, flow_rule):
        if switch_id in self.switches:
            self.switches[switch_id].append(flow_rule)
            self.flows.append({'switch_id': switch_id, 'flow_rule': flow_rule})
        else:
            raise Exception("Switch does not exist")

    # Виведення всіх правил потоку для певного комутатора
    def get_flows(self, switch_id):
        if switch_id in self.switches:
            return self.switches[switch_id]
        else:
            raise Exception("Switch does not exist")

    # Видалення правила потоку з комутатора
    def delete_flow(self, switch_id, flow_rule):
        if switch_id in self.switches:
            if flow_rule in self.switches[switch_id]:
                self.switches[switch_id].remove(flow_rule)
        else:
            raise Exception("Switch does not exist")

# Створення екземпляра SDN контролера
controller = SDNController()

# Додавання нового комутатора через API
@app.route('/add_switch', methods=['POST'])
def add_switch():
    data = request.get_json()
    switch_id = data.get('switch_id')
    controller.add_switch(switch_id)
    return jsonify({'status': 'switch added'}), 200

# Додавання нового пристрою до комутатора через API
@app.route('/add_device', methods=['POST'])
def add_device():
    data = request.get_json()
    device_id = data.get('device_id')
```

```
    switch_id = data.get('switch_id')
    try:
        controller.add_device(device_id, switch_id)
        return jsonify({'status': 'device added'}), 200
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 400

# Додавання нового правила потоку через API
@app.route('/add_flow', methods=['POST'])
def add_flow():
    data = request.get_json()
    switch_id = data.get('switch_id')
    flow_rule = data.get('flow_rule')
    try:
        controller.add_flow(switch_id, flow_rule)
        return jsonify({'status': 'flow added'}), 200
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 400

# Отримання всіх правил потоку для комутатора через API
@app.route('/get_flows/<switch_id>', methods=['GET'])
def get_flows(switch_id):
    try:
        flows = controller.get_flows(switch_id)
        return jsonify({'flows': flows}), 200
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 400

# Видалення правила потоку через API
@app.route('/delete_flow', methods=['POST'])
def delete_flow():
    data = request.get_json()
    switch_id = data.get('switch_id')
    flow_rule = data.get('flow_rule')
    try:
        controller.delete_flow(switch_id, flow_rule)
        return jsonify({'status': 'flow deleted'}), 200
    except Exception as e:
        return jsonify({'status': 'error', 'message': str(e)}), 400

# Запуск сервера для SDN контролера
if __name__ == '__main__':
    app.run(debug=True)

# Функція для оновлення топології мережі
def update_topology():
    topology = {}
    for device in controller.devices:
        switch_id = device['switch_id']
        if switch_id not in topology:
            topology[switch_id] = []
        topology[switch_id].append(device['device_id'])
    return topology

# Маршрут для отримання топології мережі
@app.route('/get_topology', methods=['GET'])
def get_topology():
    topology = update_topology()
    return jsonify({'topology': topology}), 200

# Функція для динамічного налаштування політик на комутаторах
def apply_dynamic_policies():
    for switch_id, flows in controller.switches.items():
        for flow in flows:
            if "high_priority" in flow:
                # Приклад правила для пріоритетного трафіку
                requests.post(f'http://localhost:5000/add_flow',
json={'switch_id': switch_id, 'flow_rule': flow})
```

```
# Налаштування маршруту для динамічного оновлення політик
@app.route('/apply_policies', methods=['POST'])
def apply_policies():
    apply_dynamic_policies()
    return jsonify({'status': 'policies applied'}), 200

# Резервування пропускної здатності для певних потоків
def reserve_bandwidth(flow_rule):
    if "bandwidth" in flow_rule:
        # Приклад дій для резервування пропускної здатності
        print(f"Reserving {flow_rule['bandwidth']} for flow {flow_rule['id']}")

# Застосування резервування пропускної здатності до всіх потоків
@app.route('/reserve_bandwidth', methods=['POST'])
def reserve_bandwidth_route():
    for flow in controller.flows:
        reserve_bandwidth(flow['flow_rule'])
    return jsonify({'status': 'bandwidth reserved'}), 200

# Маршрут для моніторингу стану мережі
@app.route('/monitor_network', methods=['GET'])
def monitor_network():
    # Тут можна реалізувати додаткові перевірки на стан мережі
    return jsonify({'status': 'network stable'}), 200

# Функція для автоматичного відновлення мережевих з'єднань після відмови
def recover_network():
    # Приклад дій для відновлення мережі
    for device in controller.devices:
        # Перевірка з'єднань і відновлення, якщо потрібно
        print(f"Recovering connection for device {device['device_id']}")

# Маршрут для запуску процедури відновлення мережі
@app.route('/recover_network', methods=['POST'])
def recover_network_route():
    recover_network()
    return jsonify({'status': 'network recovered'}), 200
```

Файл vlan.py

```
# Імпорт необхідних бібліотек
from flask import Flask, request, jsonify

# Створення класу для роботи з VLAN
class VLANManager:
    def __init__(self):
        self.vlans = {}

    # Додавання VLAN до комутатора
    def add_vlan(self, vlan_id, switch_id):
        if switch_id not in self.vlans:
            self.vlans[switch_id] = []
        self.vlans[switch_id].append(vlan_id)

    # Отримання списку VLAN для певного комутатора
    def get_vlans(self, switch_id):
        if switch_id in self.vlans:
            return self.vlans[switch_id]
        else:
            return []

# Створення екземпляра VLAN менеджера
vlan_manager = VLANManager()

# API для роботи з VLAN
app = Flask(__name__)

# Додавання нового VLAN
@app.route('/add_vlan', methods=['POST'])
def add_vlan():
    data = request.get_json()
    vlan_id = data.get('vlan_id')
    switch_id = data.get('switch_id')
    vlan_manager.add_vlan(vlan_id, switch_id)
    return jsonify({'status': 'vlan added'}), 200

# Отримання списку VLAN для комутатора
@app.route('/get_vlans/<switch_id>', methods=['GET'])
def get_vlans(switch_id):
    vlans = vlan_manager.get_vlans(switch_id)
    return jsonify({'vlans': vlans}), 200

# Запуск сервера
if __name__ == '__main__':
    app.run(debug=True)
```

Файл qos.py

```
# Імпорт необхідних бібліотек
from flask import Flask, request, jsonify

# Створення класу для управління QoS
class QoSManager:
    def __init__(self):
        self.qos_rules = {}

    # Додавання QoS правила
    def add_qos_rule(self, switch_id, qos_rule):
        if switch_id not in self.qos_rules:
            self.qos_rules[switch_id] = []
        self.qos_rules[switch_id].append(qos_rule)

    # Отримання всіх QoS правил для певного комутатора
    def get_qos_rules(self, switch_id):
        if switch_id in self.qos_rules:
            return self.qos_rules[switch_id]
        else:
            return []

# Створення екземпляра менеджера QoS
qos_manager = QoSManager()

# API для роботи з QoS
app = Flask(__name__)

# Додавання нового QoS правила
@app.route('/add_qos', methods=['POST'])
def add_qos():
    data = request.get_json()
    switch_id = data.get('switch_id')
    qos_rule = data.get('qos_rule')
    qos_manager.add_qos_rule(switch_id, qos_rule)
    return jsonify({'status': 'qos rule added'}), 200

# Отримання всіх QoS правил
@app.route('/get_qos/<switch_id>', methods=['GET'])
def get_qos(switch_id):
    qos_rules = qos_manager.get_qos_rules(switch_id)
    return jsonify({'qos_rules': qos_rules}), 200

# Запуск сервера
if __name__ == '__main__':
    app.run(debug=True)
```

Файл database.py

```
# Імпорт бібліотеки для роботи з SQLite
import sqlite3

# Створення бази даних для VLAN та QoS
class DatabaseManager:
    def __init__(self, db_name="network.db"):
        self.connection = sqlite3.connect(db_name)
        self.create_vlan_table()
        self.create_qos_table()

    # Створення таблиці VLAN
    def create_vlan_table(self):
        with self.connection:
            self.connection.execute("""
                CREATE TABLE IF NOT EXISTS vlans (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    switch_id TEXT NOT NULL,
                    vlan_id TEXT NOT NULL
                )
            """)

    # Створення таблиці QoS
    def create_qos_table(self):
        with self.connection:
            self.connection.execute("""
                CREATE TABLE IF NOT EXISTS qos (
                    id INTEGER PRIMARY KEY AUTOINCREMENT,
                    switch_id TEXT NOT NULL,
                    qos_rule TEXT NOT NULL
                )
            """)

    # Додавання запису до VLAN
    def add_vlan(self, switch_id, vlan_id):
        with self.connection:
            self.connection.execute("INSERT INTO vlans (switch_id, vlan_id)
VALUES (?, ?)", (switch_id, vlan_id))

    # Додавання запису до QoS
    def add_qos_rule(self, switch_id, qos_rule):
        with self.connection:
            self.connection.execute("INSERT INTO qos (switch_id, qos_rule)
VALUES (?, ?)", (switch_id, qos_rule))

    # Отримання всіх VLAN для певного комутатора
    def get_vlans(self, switch_id):
        with self.connection:
            return self.connection.execute("SELECT vlan_id FROM vlans WHERE
switch_id = ?", (switch_id,)).fetchall()

    # Отримання всіх QoS правил для певного комутатора
    def get_qos_rules(self, switch_id):
        with self.connection:
            return self.connection.execute("SELECT qos_rule FROM qos WHERE
switch_id = ?", (switch_id,)).fetchall()

# Приклад використання
db_manager = DatabaseManager()
db_manager.add_vlan("switch1", "vlan10")
db_manager.add_qos_rule("switch1", "high_priority")

print("VLANs for switch1:", db_manager.get_vlans("switch1"))
print("QoS rules for switch1:", db_manager.get_qos_rules("switch1"))
```

Файл encryption.py

```
# Імпорт необхідних бібліотек
from cryptography.fernet import Fernet

# Клас для шифрування трафіку
class TrafficEncryptionManager:
    def __init__(self):
        self.key = Fernet.generate_key()
        self.cipher = Fernet(self.key)

    # Шифрування повідомлення
    def encrypt_traffic(self, data):
        return self.cipher.encrypt(data.encode())

    # Розшифрування повідомлення
    def decrypt_traffic(self, encrypted_data):
        return self.cipher.decrypt(encrypted_data).decode()

# Приклад використання
encryption_manager = TrafficEncryptionManager()
encrypted_data = encryption_manager.encrypt_traffic("Example network data")
decrypted_data = encryption_manager.decrypt_traffic(encrypted_data)

print("Encrypted:", encrypted_data)
print("Decrypted:", decrypted_data)
```

Файл authentication.py

```
# Імпорт необхідних бібліотек
from flask import Flask, request, jsonify

# Створення класу для аутентифікації пристроїв
class DeviceAuthenticationManager:
    def __init__(self):
        self.authorized_devices = {}

    # Додавання пристрою до списку авторизованих
    def authorize_device(self, device_id, auth_token):
        self.authorized_devices[device_id] = auth_token

    # Перевірка аутентифікації пристрою
    def is_authorized(self, device_id, auth_token):
        return self.authorized_devices.get(device_id) == auth_token

# Створення екземпляра менеджера аутентифікації
auth_manager = DeviceAuthenticationManager()

# API для роботи з аутентифікацією
app = Flask(__name__)

# Додавання нового авторизованого пристрою
@app.route('/authorize_device', methods=['POST'])
def authorize_device():
    data = request.get_json()
    device_id = data.get('device_id')
    auth_token = data.get('auth_token')
    auth_manager.authorize_device(device_id, auth_token)
    return jsonify({'status': 'device authorized'}), 200

# Перевірка аутентифікації
@app.route('/check_auth', methods=['POST'])
def check_auth():
    data = request.get_json()
    device_id = data.get('device_id')
    auth_token = data.get('auth_token')
    if auth_manager.is_authorized(device_id, auth_token):
        return jsonify({'status': 'authorized'}), 200
    else:
        return jsonify({'status': 'unauthorized'}), 403

# Запуск сервера
if __name__ == '__main__':
    app.run(debug=True)
```

Файл monitoring.py

```
# Імпорт необхідних бібліотек
import psutil

# Функція для моніторингу стану мережі
def monitor_network():
    # Отримання інформації про мережеві інтерфейси
    network_info = psutil.net_if_stats()
    network_data = {}
    for interface, stats in network_info.items():
        network_data[interface] = {
            'is_up': stats.isup,
            'speed': stats.speed,
            'duplex': stats.duplex,
            'mtu': stats.mtu
        }
    return network_data

# Приклад використання
network_status = monitor_network()
for interface, data in network_status.items():
    print(f"Interface: {interface}")
    for key, value in data.items():
        print(f"  {key}: {value}")
```

Файл load_balancing.py

```
# Імпорт необхідних бібліотек
from random import choice

# Клас для реалізації балансування навантаження
class LoadBalancer:
    def __init__(self):
        self.servers = []

    # Додавання сервера до списку для балансування
    def add_server(self, server_ip):
        self.servers.append(server_ip)

    # Вибір сервера для обробки запиту
    def select_server(self):
        if not self.servers:
            raise Exception("No servers available")
        return choice(self.servers)

# Приклад використання
load_balancer = LoadBalancer()
load_balancer.add_server("192.168.1.1")
load_balancer.add_server("192.168.1.2")

selected_server = load_balancer.select_server()
print(f"Request will be handled by server: {selected_server}")
```

Файл web_interface.py

```
# Імпорт необхідних бібліотек
from flask import Flask, render_template, request, jsonify

# Створення Flask додатку
app = Flask(__name__)

# Головна сторінка з веб-інтерфейсом
@app.route('/')
def index():
    return render_template('index.html')

# Маршрут для додавання нового комутатора через веб-інтерфейс
@app.route('/add_switch', methods=['POST'])
def add_switch():
    switch_id = request.form.get('switch_id')
    # Логіка додавання комутатора через SDN контролер
    # Виклик API або функції SDN контролера
    return jsonify({'status': 'switch added'})

# Маршрут для отримання топології
@app.route('/get_topology', methods=['GET'])
def get_topology():
    # Логіка отримання топології з контролера
    return jsonify({'topology': 'network topology data'})

# Запуск веб-додатку
if __name__ == '__main__':
    app.run(debug=True)

# HTML для веб-інтерфейсу (index.html)
'''
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SDN Controller</title>
</head>
<body>
    <h1>SDN Network Management</h1>
    <form action="/add_switch" method="post">
        <label for="switch_id">Switch ID:</label>
        <input type="text" id="switch_id" name="switch_id" required>
        <button type="submit">Add Switch</button>
    </form>

    <h2>Network Topology</h2>
    <button onclick="getTopology()">Get Topology</button>
    <div id="topology"></div>

    <script>
        function getTopology() {
            fetch('/get_topology')
            .then(response => response.json())
            .then(data => {
                document.getElementById('topology').innerText =
JSON.stringify(data.topology);
            });
        }
    </script>
</body>
</html>
'''
```

Файл multicast.py

```
# Імпорт необхідних бібліотек
from flask import Flask, request, jsonify

# Створення класу для управління мультикастом
class MulticastManager:
    def __init__(self):
        self.multicast_groups = {}

    # Додавання нового мультикаст групи
    def add_multicast_group(self, group_id, devices):
        self.multicast_groups[group_id] = devices

    # Отримання пристроїв для мультикаст групи
    def get_multicast_devices(self, group_id):
        return self.multicast_groups.get(group_id, [])

# Створення екземпляра менеджера мультикасту
multicast_manager = MulticastManager()

# Створення API
app = Flask(__name__)

# Додавання мультикаст групи
@app.route('/add_multicast_group', methods=['POST'])
def add_multicast_group():
    data = request.get_json()
    group_id = data.get('group_id')
    devices = data.get('devices')
    multicast_manager.add_multicast_group(group_id, devices)
    return jsonify({'status': 'multicast group added'}), 200

# Отримання пристроїв для мультикаст групи
@app.route('/get_multicast_devices/<group_id>', methods=['GET'])
def get_multicast_devices(group_id):
    devices = multicast_manager.get_multicast_devices(group_id)
    return jsonify({'devices': devices}), 200

# Запуск сервера
if __name__ == '__main__':
    app.run(debug=True)
```

Файл firmware_update.py

```
# Імпорт необхідних бібліотек
import requests

# Клас для оновлення прошивок
class FirmwareUpdateManager:
    def __init__(self):
        self.devices = {}

    # Додавання пристрою для оновлення
    def add_device(self, device_id, firmware_version):
        self.devices[device_id] = firmware_version

    # Оновлення прошивки для всіх пристроїв
    def update_firmware(self, new_version):
        for device_id in self.devices:
            self.devices[device_id] = new_version
            print(f"Updated {device_id} to firmware version {new_version}")

# Приклад використання
firmware_manager = FirmwareUpdateManager()
firmware_manager.add_device("device1", "v1.0")
firmware_manager.add_device("device2", "v1.0")
firmware_manager.update_firmware("v2.0")
```

Файл cloud_integration.py

```
# Імпорт необхідних бібліотек
import requests

# Клас для інтеграції з хмарною платформою
class CloudIntegrationManager:
    def __init__(self, api_url):
        self.api_url = api_url

    # Надсилання запиту для управління через хмарну платформу
    def send_request(self, endpoint, data):
        response = requests.post(f"{self.api_url}/{endpoint}", json=data)
        return response.json()

# Приклад використання
cloud_manager = CloudIntegrationManager(api_url="https://cloud-sdn-
platform.com")
response = cloud_manager.send_request(endpoint="add_switch", data={"switch_id":
"switch1"})
print(response)
```

Файл dynamic_routing.py

```
# Імпорт необхідних бібліотек
import networkx as nx

# Клас для динамічного обчислення маршрутів
class DynamicRoutingManager:
    def __init__(self):
        self.network = nx.Graph()

    # Додавання вузла та ребра для створення мережевої топології
    def add_link(self, src, dst, weight):
        self.network.add_edge(src, dst, weight=weight)

    # Обчислення найкоротшого шляху між двома вузлами
    def calculate_shortest_path(self, src, dst):
        return nx.shortest_path(self.network, source=src, target=dst,
weight='weight')

# Приклад використання
routing_manager = DynamicRoutingManager()
routing_manager.add_link('switch1', 'switch2', 1)
routing_manager.add_link('switch2', 'switch3', 2)
routing_manager.add_link('switch1', 'switch3', 3)

path = routing_manager.calculate_shortest_path('switch1', 'switch3')
print(f"Shortest path from switch1 to switch3: {path}")
```

Файл anomaly_detection.py

```
# Імпорт необхідних бібліотек
from sklearn.ensemble import IsolationForest

# Клас для виявлення аномалій в мережевому трафіку
class AnomalyDetectionManager:
    def __init__(self):
        self.model = IsolationForest(contamination=0.1)

    # Навчання моделі на основі нормального трафіку
    def train(self, data):
        self.model.fit(data)

    # Виявлення аномалій в нових даних
    def detect_anomalies(self, new_data):
        predictions = self.model.predict(new_data)
        return predictions

# Приклад використання
anomaly_manager = AnomalyDetectionManager()
normal_data = [[10], [15], [14], [13], [10]]  # Нормальні показники трафіку
anomaly_manager.train(normal_data)

new_data = [[20], [5], [14]]  # Нові дані для перевірки
anomalies = anomaly_manager.detect_anomalies(new_data)
print(f"Anomaly detection results: {anomalies}")
```

Файл openflow_support.py

```
# Імпорт бібліотек для підтримки OpenFlow
from pox.lib.util import dpidToStr
from pox.core import core
import pox.openflow.libopenflow_01 as of

# Клас для підтримки OpenFlow
class OpenFlowManager:
    def __init__(self):
        self.connection = None

    # Обробка підключення комутатора
    def _handle_ConnectionUp(self, event):
        self.connection = event.connection
        print(f"Switch connected: {dpidToStr(event.dpid)}")

    # Додавання правила потоку через OpenFlow
    def add_flow(self, src_ip, dst_ip, action_output_port):
        if self.connection:
            msg = of.ofp_flow_mod()
            msg.match = of.ofp_match(nw_src=src_ip, nw_dst=dst_ip)
            msg.actions.append(of.ofp_action_output(port=action_output_port))
            self.connection.send(msg)

# Ініціалізація OpenFlow
def launch():
    core.openflow.addListenerByName("ConnectionUp",
OpenFlowManager()._handle_ConnectionUp)
```

Файл distributed_sdn.py

```
# Імпорт необхідних бібліотек
from flask import Flask, request, jsonify

# Клас для управління розподіленою архітектурою SDN
class DistributedSDNManager:
    def __init__(self):
        self.controllers = []

    # Додавання нового SDN контролера
    def add_controller(self, controller_ip):
        self.controllers.append(controller_ip)

    # Взаємодія з іншими контролерами
    def sync_with_controllers(self):
        for controller in self.controllers:
            # Логіка синхронізації між контролерами
            print(f"Syncing with controller: {controller}")

# Приклад використання
distributed_manager = DistributedSDNManager()
distributed_manager.add_controller("192.168.1.100")
distributed_manager.add_controller("192.168.1.101")
distributed_manager.sync_with_controllers()
```

Файл virtualization_integration.py

```
# Імпорт бібліотеки для управління віртуалізацією
import libvirt

# Клас для інтеграції з віртуалізацією
class VirtualizationManager:
    def __init__(self):
        self.conn = libvirt.open()

    # Створення нової віртуальної машини
    def create_vm(self, xml_config):
        domain = self.conn.createXML(xml_config, 0)
        return domain

    # Перезапуск віртуальної машини
    def restart_vm(self, domain_name):
        domain = self.conn.lookupByName(domain_name)
        domain.reboot()

# Приклад використання
vm_manager = VirtualizationManager()
xml_config = "<domain>...</domain>"  # XML конфігурація віртуальної машини
domain = vm_manager.create_vm(xml_config)
print(f"VM created: {domain.name()}")
```