Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра інформаційних технологій та комп'ютерної інженерії
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

студента Каніболоцького Володимира Володимировича
(ПІБ)
академічної групи 123-19з-1
(шифр)
спеціальності 123 Комп'ютерна інженерія
(код і назва спеціальності)
за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему «Комп'ютерна система моніторингу доступності хостів компанії "Van Ommeren" з детальним опрацюванням побудови, налаштування та безпеки корпоративної мережі»
(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	проф. Гнатушенко В.В			
розділів:				
розробка апаратної частини	доц. Бешта Д.О.			
розробка корпоративної мережі	ас. Панферова Я.В.			

Рецензент				

Нормоконтролер	проф. Цвіркун Л.І.			

Дніпро
2023

ЗАТВЕРДЖЕНО:

завідувач кафедри
інформаційних технологій
та комп'ютерної інженерії
(повна назва)

_____ Гнатушенко В.В.
(підпис) (прізвище, ініціали)
« 16 » 05 2023 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня бакалавр

студента Каніболоцького В.В. **академічної групи** 123-19з-1
(прізвище та ініціали) (шифр)
спеціальності 123 Комп'ютерна інженерія
за освітньо-професійною програмою 123 Комп'ютерна інженерія
(офіційна назва)
на тему «Комп'ютерна система моніторингу доступності хостів компанії "Van Ommeren" з реалізацією побудови, налаштування та безпеки корпоративної мережі» затверджену наказом ректора НТУ «Дніпровська політехніка» від 11.04.2023 № 256-с

Розділ	**Зміст**	**Термін виконання**
Стан питання та постановка завдання	На основі матеріалів виробничих практик, інших науково-технічних джерел розглянути призначення та завдання систем моніторингу IT-інфраструктури	10.05.2023
Розробка апаратної частини	Розробити вимоги до функцій, виконуваними системою на основі Network Controller, зробити вибір постачальника рішень для Network Controller	17.05.2023
Розробка корпоративної мереж	Побудувати в Packet Tracer модель корпоративної мережі компанії з Network Controller, виконати налаштування та перевірку роботи системи	24.05.2023
Розробка компонента системи	Розробити програму для моніторингу хостів, яка сповіщатиме в Telegram, коли мережний пристрій недоступний	31.05.2023

Завдання видано _____ проф. Гнатушенко В.В
(підпис керівника) (прізвище, ініціали)

Дата видачі 19.04.2023

Дата подання до екзаменаційної комісії 20.06.2023

Прийнято до виконання _____ Каніболоцький В.В .
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 86 с., 30 рис., 11 табл., 2 додатки, 16 джерел.

МОНІТОРИНГ, NETWORK CONTROLLER, REST API, СИСТЕМА МОНІТОРИНГУ, TELEGRAM , PACKET TRACER, CISCO, PYTHON

Об’єкт розробки – комп'ютерна система компанії ""Van Ommeren" " з реалізацією системи моніторингу доступності хостів, налаштування та безпеки комп’ютерної мережі.

Головною метою даної роботи є створення системи моніторингу, яка забезпечить підвищення надійності комп'ютерної мережі компанії "Van Ommeren" і покращить ефективність процесу адміністрування.

У рамках дослідження було розглянуто область моніторингу мережі, визначено його поняття та важливість для підприємства. Також проведений аналіз типової структури системи моніторингу і існуючих систем цього типу.

Були встановлені вимоги до системи, а також розроблена апаратна частина комп'ютерної системи.

Розроблено макет робочої мережі усієї мережі в Packet Tracer, додано в мережу Network Controller (NC) та створена програма на Python для моніторингу хостів, яка сповіщатиме в Telegram, коли мережний пристрій недоступний або знов приєднався до мережі.

У рамках даного дослідження було розглянуто предметну область, встановлено концепцію моніторингу комп'ютерних мереж та розкрито його важливість у діяльності підприємств. Також була проаналізована типова структура системи моніторингу та існуючі системи цього типу. В результаті були визначені вимоги до розробки власної системи моніторингу, а також використання месенджера Telegram для сповіщення.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КС – комп’ютерна система

LAN – локальні мережі

WLAN – бездротова локальна мережа

API – прикладни́й програ́мний інтерфе́йс (Application Programming Interface)

VLSM – маски підмережі змінної довжини (англ. Variable Length Subnet Mask)

DHCP – протокол динамічного налаштування вузла (англ. Dynamic Host Configuration Protocol)

NC – Network Controller

DHCP – Dynamic Host Configuration Protocol

GUI – Graphical User Interface

HTTP – Hyper Text Transfer Protocol

JSON – JavaScript Object Notation

NAT – Network Address Translation

OSPF – Open Shortest Path First Protocol

SDN – Software Defined Networking

ВСТУП

У сучасному світі практично неможливо уявити роботу фірми або компанії без комп'ютерної мережі. Мережа стала важливим інструментом, який дозволяє підприємствам оптимізувати свою роботу та виробництво, об’єднавши всі свої комп’ютери в єдину систему. Таким чином, це вже не розкіш, а необхідність. Однак із зростанням складності мереж стало надзвичайно важливо мати систему моніторингу мережі, яка може допомогти контролювати роботу всієї мережі та своєчасно реагувати на проблеми.

Моніторинг мережі – це процес контролю доступності, надійності, роботи та продуктивності складних мереж. Це включає відстеження та аналіз компонентів мережі, таких як маршрутизатори, комутатори, брандмауери і хости, а також з'єднань між ними.

Системи, призначені для моніторингу мережі використовують програмні та апаратні засоби для контролю різних аспектів мережі та її функціонування, зокрема, таких як трафік, використання пропускної здатності та час безвідмовної роботи. Ці системи здатні виявляти пристрої та інші елементи, що складають мережу або підключені до неї, і забезпечувати оновлення їх стану.

Моніторинг мережі сприяє зменшенню середнього часу відновлення (MTTR) для системних адміністраторів та дозволяє вчасно приймати необхідні заходи для вирішення проблем у реальному часі. При виявленні проблем система моніторингу повідомляє системних адміністраторів безпосередньо або через систему підтримки, щоб вони могли якомога швидше знайти вирішення проблеми.

Отже, наявність надійної системи моніторингу мережі є критично важливою для будь-якого підприємства, яке хоче забезпечити безперебійну роботу своєї мережі та додатків, які вона підтримує. Інвестуючи в надійну систему моніторингу мережі, компанії можуть оптимізувати свою роботу та виробництво, надаючи своїм клієнтам високоякісні послуги.

Зі збільшенням розміру сучасної мережі та частотою змін, необхідних для бізнесу, управління та автоматизація мереж за допомогою інтерфейсу командного рядка (CLI) є неефективним та схильним до помилок. Новий підхід, який використовує програмування на основі API, дозволяє масштабно вносити зміни в управління мережевими пристроями.

Ця інновація забезпечується завдяки мережевим контролерам, які встановлюють в корпоративній мережі.

Мережевий контролер – це спеціалізований компонент мережевої системи, який призначений для збору даних і управління конфігураціями компонентів мережі.

Впровадження цієї інновації, набагато зменшило витрати на підтримку мережевого обладнання і структури загалом, так як адміністратор може отримати доступ та вести моніторинг не лише в межах певного офісу, а й одержувати інформацію від інших підрозділів та філіалів. Також плюсами є те, що підвищилася загальна продуктивність та рівень безпеки.

Мета даної роботи полягає в розробці системи, яка забезпечуватиме моніторинг стану мережних вузлів з метою забезпечення надійності комп'ютерної мережі компанії "Van Ommeren" і підвищення ефективності процесу адміністрування.

Для досягнення мети моєї кваліфікаційної роботи бакалавра передбачаються наступні завдання::

– провести докладний аналіз існуючих систем моніторингу;

– визначити конкретні вимоги до системи моніторингу комп'ютерної мережі у компанії "Van Ommeren";

– спроектувати систему моніторингу, яка перевіряє доступність мережних пристроїв компанії «Van Ommeren»;

– реалізувати систему, яка була спроектована, щоб надсилати повідомлення про стан мережних пристроїв у Telegram.

1 СТАН ПИТАННЯ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Характеристика і структура об’єкта впровадження

Комп'ютерна мережа – це з'єднання комп'ютерів, пристроїв і інших пристроїв для обміну даними та ресурсами. Існує багато різних типів комп'ютерних мереж, таких як локальні мережі (LAN), місцеві мережі (MAN), глобальні мережі (WAN) та бездротові мережі (WLAN).

Модернізацію комп'ютерної мережі замовила нам компанії "Van Ommeren". Компанія "Van Ommeren" на ринку послуг (підприємство в області логістики, транспорту і торгівлі) з 2001 року. За цей період накопичено великий досвід роботи як на території України, так і за її межами.

Враховуючи поточну ситуацію з глобальною пандемією та військовими діями в Україні, в логістиці розпочалася справжня IT-революція. Варто також зазначити, що логістичні компанії використовують IT-платформи для обміну митними та вантажними платежами.

Транспорт, логістика та інформаційні технології стають життєвою необхідністю будь-яких суб’єктів господарювання. Глобальним трендом сучасної логістики є "правило 7R". Продукт потрібної якості повинен бути доставлений у потрібній кількості, у потрібний час, у потрібне місце, потрібному клієнту і за потрібною ціною. Без належним чином розвиненої логістики переміщення товарів і людей неможливе. Транспортно-логістична інфраструктура пов'язує виробництво і споживання, обслуговує процес руху товарів, створює умови, необхідні для задоволення попиту, скорочує час реалізації товарів, полегшує обіг капіталу, знижує витрати на переміщення товарів і ціни та створює ефективне конкурентне середовище. З метою координації діяльності всіх видів транспорту, промисловості та ефективного розвитку регіонів України необхідно побудувати логістичну систему на основі

регіональних розподільчих логістичних центрів, які стануть точками зростання для нашої країни [1].

Одна з провідних компаній Південно-Східного регіону України з більш, ніж 21-м досвідом надання логістичних і торгівельних послуг по всій території України та за її межами. Основні відомості про компанію-замовника: ТОВ "Van Ommeren"; має в своєму складі 41 працівник; адреса головного офісу 49000, Україна, Дніпропетровська обл., м. Дніпро, вул. Набережна Заводська, 50.

В комп'ютерній мережі компанії «Van Ommeren» виділені такі сервери файл-сервер і Інтернет-сервер. Кожна робоча станція в робочих групах повинна мати доступ до всіх серверів. Разом з цим необхідно організувати підключення до мережі Інтернет.

Основними видами трафіку в мережі будуть локальний, робота в Інтернет та з файл-сервером.

Фахівці компанії досвідчені в сферах логістики, торгівлі, а також господарського, корпоративного, інвестиційного, податкового, митного, антимонопольного та міжнародного господарства.

Компанія "Van Ommeren" успішно співпрацює з великими українськими підприємствами, міжнародними компаніями, державними структурами та IT-корпораціями.

Для розробки проєкту комп’ютерної системи для компанії "Van Ommeren" із застосуванням сучасних мережних технологій, необхідно проаналізувати структурні підрозділи, що будуть поєднані мережею.

1.2 Організаційно-управлінська структура об’єкта впровадження

Компанія "Van Ommeren" має організаційну структуру, представлену на рисунку 1.1.

Компанія має лінійно-функціональну організаційну структуру. Тобто вона організована суворо ієрархічно, характеризується розподілом сфер відповідальності та єдиноначальності, а діяльність структурних підрозділів є

спеціалізованою і визначається їхніми основними функціональними характеристиками.

Організаційно-управлінська структура компанії представлена трьома рівнями управління: вищим, середнім і низовим.

Вищому рівню управління відповідає підсистема менеджменту.

Вищий рівень управління очолює директор компанії. Він відповідає за загальне керівництво діяльністю компанії Van Omuren і підзвітний за ті операції, які знаходяться безпосередньо в його зоні відповідальності.

Генеральному директору безпосередньо підпорядковуються всі посадові особи, які входять до складу компанії.

Проміжні рівні структури управління забезпечують виконання функцій, що входять до складу підсистеми. До проміжного рівня належать керівники всіх структурних підрозділів, які забезпечують функціонування компанії та виконання виробничих завдань, поставлених на вищих рівнях.

Нижній (операційний) рівень управління включає персонал усіх структурних підрозділів, який безпосередньо виконує завдання, поставлені вищим і середнім рівнями управління.

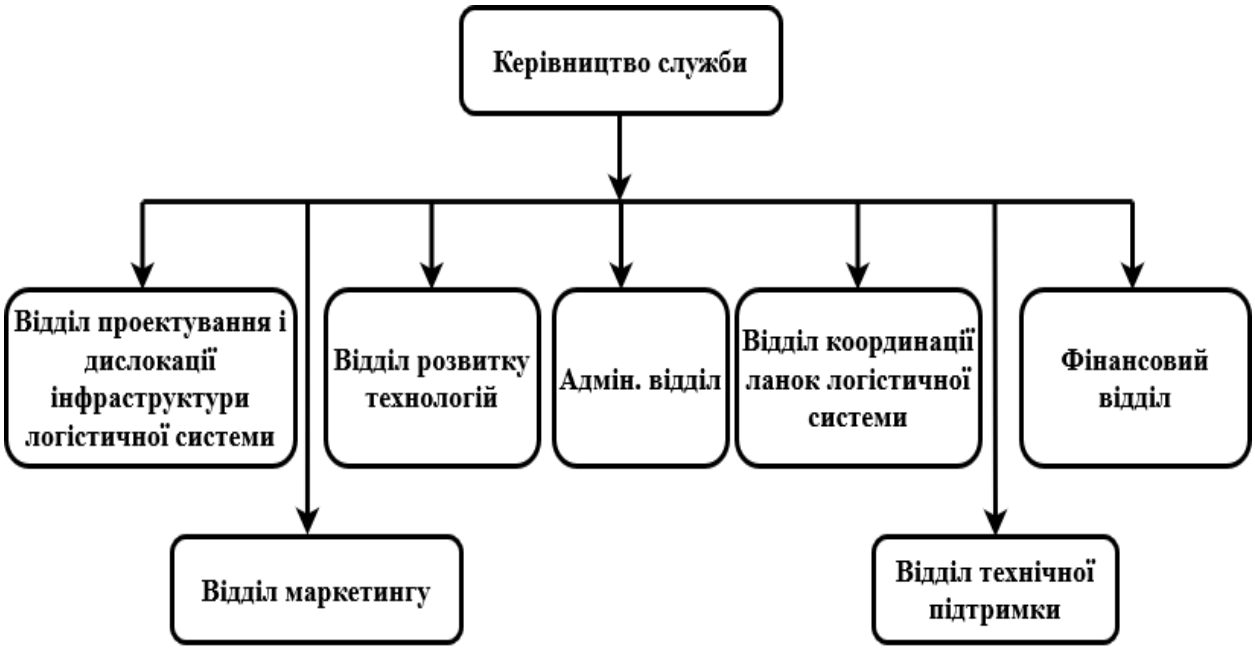

Рисунок 1.1 – Організаційна структура Компанія "Van Ommeren"

1.3 Стислі відомості про технології збору та передачі інформації компанії "Van Ommeren»

До складу приміщень компанії "Van Ommeren" входять: головний офіс за адресою: 49000, Україна, Дніпропетровська обл., м. Дніпро, вул. Набережна Заводська, 50 (приміщення на першому поверсі в 260 м2) та філія за адресою: Проспект Богдана Хмельницького, 4 (приміщення на першому поверсі в 75 м2).

В приміщені будівлі головного офісу розташовані наступні підрозділи зі своїми структурними групами: керівництво, адміністративний відділ (відділ кадрів), фінансовий відділ (входить відділ закупівель), відділ координації ланок логістичної системи, відділ проектування і дислокації інфраструктури логістичної системи, відділ маркетингу, технічний відділ. В приміщені будівлі філії розташований підрозділ маркетингу.

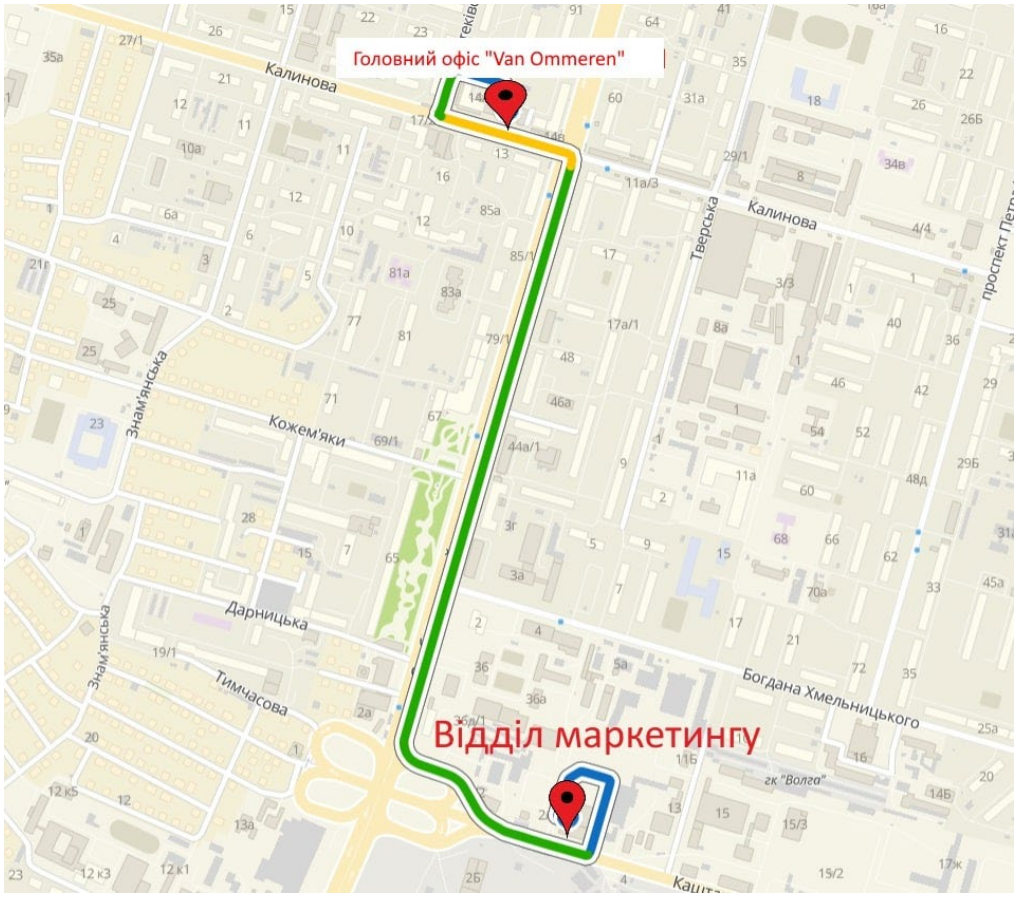

Рисунок 1.2 – Схема топологічного розташування компанії "Van Ommeren"

Розгортання корпоративної мережі дозволить підприємству краще задовольняти поставленим вимогам сучасного бізнесу.

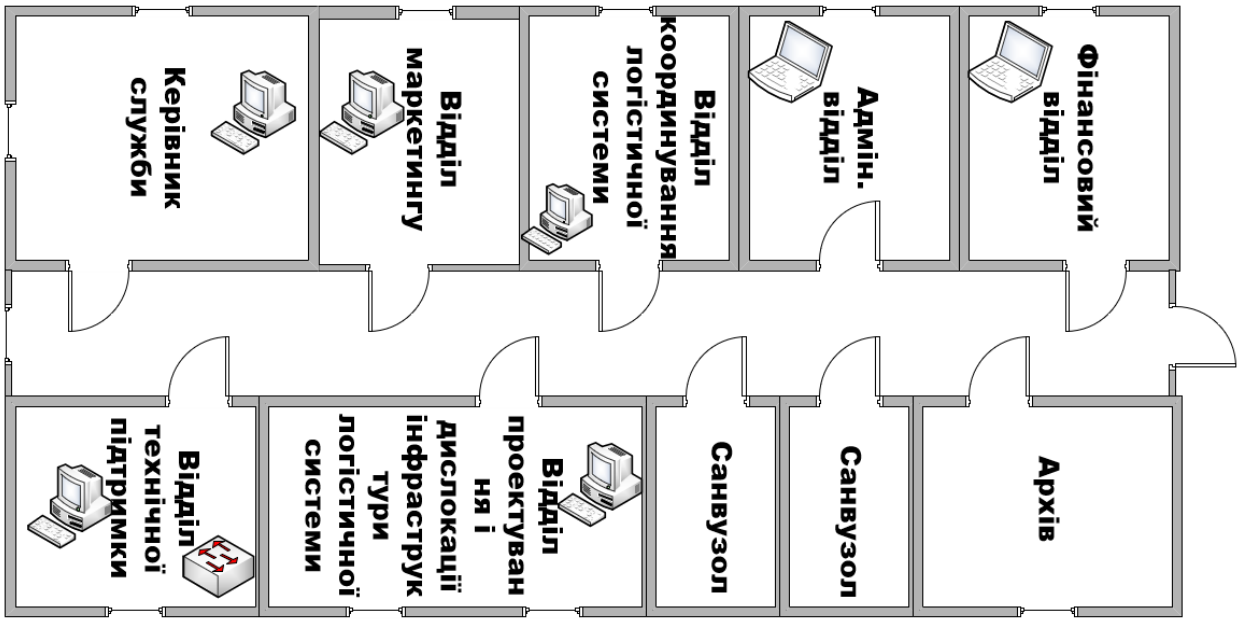

Рисунок 1.3 – План приміщення головного офісу

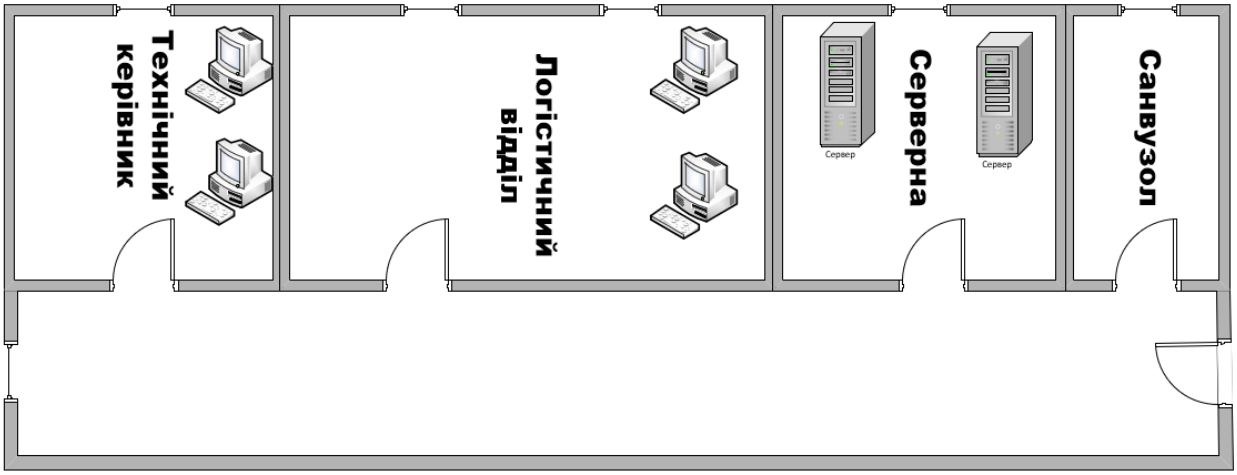

Рисунок 1.4 – План офісного приміщення філії

Ступінь комп'ютеризації робочих місць підприємства приведена в табл.1.1.

Таблиця 1.1 – Комп'ютеризовані робочі місця підприємства

Структурний підрозділ	Кількість		
	Працівники	ПК	Принтери
Керівництво	2	2	
Адміністративний відділ	3	3	1
Фінансовий відділ (в т.ч. закупівель)	4	4	1
Відділ проектування і дислокації інфраструкутури логістичної системи	4	4	1
Відділ координації ланок логістичної системи	4	4	1
Відділ маркетингу	3	3	1
Відділ розвитку технологій	3	3	1
Відділ технічної підтримки	2	2	1
Загалом ПК		25	7

1.4 Призначення та завдання систем моніторингу IT-інфраструктури

Системи моніторингу IT-інфраструктури призначені для збору, аналізу та відображення даних щодо стану різних компонентів інфраструктури компанії, таких як сервери, мережеві пристрої, бази даних, додатки та інші.

Основні завдання систем моніторингу IT-інфраструктури включають:

– забезпечення безперебійної роботи інфраструктури: система моніторингу допомагає виявляти проблеми в роботі компонентів інфраструктури до того, як вони призведуть до збоїв інфраструктури;

– виявлення проблем: система моніторингу дозволяє виявляти проблеми, такі як збої апаратного забезпечення, низька продуктивність, помилки програмного забезпечення та інші;

– аналіз продуктивності: система моніторингу допомагає виявляти проблеми з продуктивністю, такі як перевантаження, низька швидкість роботи та інші;

– відслідковування поточного стану: система моніторингу дозволяє відстежувати поточний стан інфраструктури компанії, зокрема, вільний обсяг дискового простору, поточну навантаженість системи та інші показники;

– попередження про проблеми: система моніторингу допомагає попередити про можливі проблеми та допомагає швидко вирішувати проблеми, що виникли;

– оптимізація інфраструктури: система моніторингу дозволяє виявляти можливості для оптимізації роботи інфраструктури, такі як підвищення продуктивності та ефективності;

– планування майбутнього розвитку: система моніторингу допомагає зрозуміти потреби інфраструктури компанії та планувати майбутні розвиткові заходи на основі отриманих даних щодо стану інфраструктури.

Усі системи управління і моніторингу (СУМ) слугують для того, щоби знизити роль людського чинника при експлуатації IT-інфраструктури. Вочевидь обсяг інформації, що має відстежувати системний адміністратор, просто величезний. Тому він повинен мати кваліфікацію, достатню для вирішення проблем. Слід розуміти, що система моніторингу не може замінити собою кваліфікацію користувачів і адміністраторів. Вона лише надає їм необхідні відомості для ухвалення правильного рішення [2].

Система моніторингу має давати чітку відповідь на питання, пов’язані з ефективністю функціонування підсистеми IT-інфраструктури. При цьому поряд зі зниженням наявних витрат об’єкта на інформаційні технології СУМ забезпечують подальший розвиток IT-інфраструктури у виваженому й стратегічно правильному напрямі. Насамперед тому, що керівництво об’єкта має розуміти, які процеси відбуваються на сучасному рівні розвитку IT-інфраструктури і яка модернізація необхідна в перспективі. Адже впровадження нових сервісів і технологій вимагає певного рівня розвитку IT-інфраструктури. Найчастіше впровадження нових програмно-апаратних засобів неможливе через «слабку» IT-інфраструктуру, коли система впроваджується, але не може потім повноцінно функціонувати [2].

Також, системи моніторингу IT-інфраструктури можуть бути інтегровані з іншими системами управління, такими як системи управління конфігураціями, системи управління сервісами, що дозволяє більш ефективно керувати інфраструктурою компанії.

Отже, системи моніторингу IT-інфраструктури є дуже важливим інструментом для підтримки роботи IT-інфраструктури компанії на оптимальному рівні продуктивності та ефективності, а також для зменшення ризиків виникнення проблем та збоїв.

Основними об’єктами для моніторингу IT є: сервери; мережеве обладнання; додатки; бази даних; користувальницькі станції; спеціальні системи [2].

1.5 Огляд існуючих рішень моніторингу IT-інфраструктури

Моніторинг IT-інфраструктури компанії "Van Ommeren" здійснюється для оцінки стану IT-середовища підприємства, включаючи доступність, продуктивність, використання та безпеку. Незалежно від розташування додатків, система моніторингу повинна відстежувати показники та надсилати сповіщення про стан інфраструктури, а також генерувати звіти зі статистикою.

При виборі програмного забезпечення для моніторингу IT важливим фактором є складність налаштування. Деякі інструменти вже мають готову інтеграцію з компонентами нашої інфраструктури, тоді як інші потребують додаткових зусиль для налаштування.

Давайте розглянемо особливості таких систем моніторингу як Zabbix, SNMP і Nagios.

Zabbix є вільною системою моніторингу, яка дозволяє відстежувати статус різних сервісів комп'ютерної мережі, серверів та мережевого обладнання. Для зберігання даних Zabbix використовує MySQL, PostgreSQL, SQLite або Oracle Database, а веб-інтерфейс реалізований на PHP. Система підтримує різні методи моніторингу.

1.5.1 Zabbix

Zabbix [3] є вільною системою моніторингу, яка дозволяє відстежувати статус різних сервісів комп'ютерної мережі, серверів та мережевого обладнання. Для зберігання даних Zabbix використовує MySQL, PostgreSQL, SQLite або Oracle Database, а веб-інтерфейс реалізований на PHP. Система підтримує різні методи моніторингу [4]:

– Simple checks – може перевіряти доступність і реакцію стандартних сервісів, таких як SMTP або HTTP, без встановлення будь-якого програмного забезпечення на хості, що спостерігається;

– Zabbix agent – може бути встановлений на UNIX-подібних або Windows-хостах для отримання даних про навантаження процесора, використання мережі, дисковому просторі тощо;

– External check – виконання зовнішніх програм, також підтримується моніторинг через SNMP.

Zabbix – безкоштовне програмне забезпечення з відкритим кодом для мережі, сервера, хмари, додатків та послуг, яке було випущено під версією GNU (Загальнодоступна ліцензія), без обмежень та прихованих витрат, вільна система моніторингу служб і станів комп'ютерної мережі, що в свою чергу складається з трьох базових компонентів:

– сервера для координації виконання перевірок, формування перевірочних запитів та накопичення статистики;

– агентів для здійснення перевірок на стороні зовнішніх хостів;

– фронтенда для організації управління системою.

Він має функції розширеного виявлення проблем та інтелектуального попередження та усунення несправностей.

Щоб зняти навантаження з центрального сервера і сформувати розподілену мережу моніторингу можна використати розгорнуту серію проксі-серверів, що агрегують дані про перевірку групи хостів. Сирцевий код агентів і серверної частини написаний на мові Сі, для розробки вебінтерфейсу використано мову PHP, дані можуть зберігатися в СУБД MySQL, PostgreSQL,

SQLite, DB2 і Oracle. Код проекту поширюється під ліцензією GPLv2. Програмне забезпечення Zabbix:

– виявляє проблеми із визначеними інтелектуальними порогами з високою гнучкістю, безліччю рівнів безпеки та виявленням аномалій;

– проводить гнучкий та розширюваний збір даних;

– автоматично визначає мережеві пристрої та зміни конфігурації пристрою;

– надає різноманітні опції для сповіщень;

– візуалізує на одній панелі скла, включаючи інформаційні панелі, графіки, карти та багато іншого на основі віджетів;

– захищає дані на всіх рівня, а також, у разі виявлення будь-якої проблеми, повідомляє та виправляє чи проблеми, а також захист даних на всіх рівнях;

– розгортання без зусиль, заощаджуючи ваш час, використовуючи готові шаблони та масштабуючи без обмежень.

Zabbix має чудові можливості для технічного користувача, і найбільш вигідним є програмне забезпечення з відкритим кодом.

Використання різних протоколів, таких як SNMP, IPMI та інші розширені можливості програмного забезпечення Zabbix дозволяють будувати гнучкі сценарії ескалації.

Zabbix може інтегруватись з різними інструментами та розширювати свої можливості за допомогою різноманітних розширень та настроювань. Він володіє активною спільнотою користувачів та розробників, що забезпечує підтримку, розвиток та покращення системи.

Хотів би відмітити, що система IT моніторингу Zabbix призупинила свою діяльність у Росії.

1.5.2 SNMP

SNMP (Simple Network Management Protocol) – це стандартний інтернет-протокол для керування пристроями в IP-мережах на основі архітектур

TCP/UDP [5]. Він дає можливість відстежувати керовані мережеві пристрої, включно з маршрутизаторами, мережевими комутаторами, серверами, принтерами та іншими пристроями, що ввімкнені через IP, через єдину систему керування/програмне забезпечення. SNMP використовує центральний комп'ютер із програмним забезпеченням SNMP для керування мережевими комутаторами. На сьогодні більшість мережевих комутаторів на ринку, як гігабітні комутатори, так і комутатори 40G, підтримують SNMP.

SNMP надає уніфікований і простий спосіб керування цими комутаторами.

SNMP – один із поширених мережевих протоколів для керування та моніторингу мережевих елементів. Щоб SNMP працювало, необхідно враховувати компоненти SNMP і принцип його роботи. Більшість мережевих елементів професійного рівня постачаються з вбудованим агентом SNMP.

Важливо відмітити і те, що перша версія SNMP (SNMPv1) була розроблена, коли механізму захисту не було як такого і практично вона не має захисту від злому і вразлива для атаки. Але в наступному протоколі SNMPv2 було значно поліпшено продуктивність і безпеку. Цей протокол підходить для малих і середніх підприємств, а також великих мереж із низькими вимогами до безпеки. Зараз майже всі основні великі виробники платформ мережевого управління підтримують протокол SNMPv3 (включно з HP OpenView, IBM Tivoli, CA Unicenter), який вирішує всі проблеми.

SNMP використовує центральний комп'ютер із програмним забезпеченням SNMP для керування мережевими комутаторами. Сьогодні більшість мережевих комутаторів на ринку, чи то гігабітні комутатори, чи то комутатори 40G, підтримують SNMP. SNMP надає уніфікований і простий спосіб керування цими комутаторами.

SNMP дає змогу адміністратору керувати різними застосунками та хмарними сервісами. Протокол має функціонал для виконання таких операцій:

– переналаштувати IP-адреси;

– надсилати запити для контролю пристроїв;

– віддалено скидати паролі;

– відстежувати навантаження на GPU сервера й отримувати повідомлення про перевищення допустимого порогу;

– збирати інформацію про поточні помилки;

– отримувати повідомлення про несправності;

– додати зведення через сторонні OID та ін.

1.5.3 Nagios

Nagios - це програмне забезпечення з відкритим вихідним кодом, призначене для моніторингу комп'ютерних систем та мереж [6].

Nagios - це програмний додаток, який використовується для

– спостереження, контролю стану обчислювальних вузлів і сервісів;

– сповіщення адміністратора про те, що якийсь із сервісів перестав працювати.

Nagios з’явилася раніше за Zabbix. Вона має готові плагіни для моніторингів. Якщо порівняти важкість налаштування Nagios із Zabbix , то Nagios набагато легше встановити та налаштувати. Але останнім часом її безкоштовна версія майже не розвивається. Має форк – Icinga.

Nagios пропонує такі рішення, як програмне забезпечення мережевого моніторингу, моніторинг мережевого трафіку та мережевий аналізатор. Мережевий аналізатор Nagios оснащений такими функціями, як комплексна інформаційна панель, розширені візуалізації, спеціальний моніторинг додатків, автоматизовані сповіщення, спеціалізовані подання та вдосконалене управління користувачами [7].

Функції програми Nagios [7]**:**

– контролює наявність вузлів та час їх роботи;

– спостерігає за часом відгуку кожного вузла;

– надає звіти та візуальні зображення;

– здійснює моніторинг мережі щодо збоїв серверів.

Програмне забезпечення для моніторингу мережі підтримує Microsoft, VMWare та Linux.

1.6 Визначення можливих напрямків рішення поставлених завдань

Автоматизація мережі – це процес автоматизації конфігурації, керування, тестування, розгортання та експлуатації фізичного та віртуального мережевого обладнання. Доступність мережевих послуг покращується, коли щоденні операції та функції мережі автоматизовані, а повторювані процеси регулюються й ефективно обробляються. Автоматизація мережі також важлива в програмно-визначених мережах (SDN), мережевій віртуалізації та мережевій оркестрації, що дозволяє автоматично розгортати віртуальних мережевих орендарів і виконувати такі завдання, як віртуальне балансування навантаження [5].

Мережеві завдання можуть бути об'єднані програмними продуктами автоматизації в готові програми, які можна вибирати, планувати та запускати із зовнішнього інтерфейсу програми. Шляхом упаковки інтерфейсів прикладного програмування (API), модулів, інвентаризацій і модулів, що підключаються до збірників сценаріїв, які користувачі можуть переглядати, вибирати і запускати для автоматизації налаштування мережі, безпеки, оркестрування, виділення ресурсів і багато іншого у постачальників послуг, таких як AWS, Microsoft і Cisco, наприклад, Red Hat Ansible Automation Platform може використовуватися для автоматизації мережевих дозволів та мереж.

Мережевий контролер, стрижень керування програмно визначеною мережею (SDN) – це розширювана роль сервера, що забезпечує централізовану програмовану точку автоматизації, яка може керувати, налаштовувати, контролювати та усувати неполадки віртуальної мережевої інфраструктури замість простого ручного налаштування мережевих пристроїв і послуги. Ось деякі переваги впровадження Network Controller:

– централізоване керування: Network Controller дозволяє забезпечити централізоване керування всіма складовими мережі з одного місця. Це

зменшує час, необхідний для налаштування та управління мережею, та забезпечує більш ефективне використання ресурсів мережі;

– автоматизоване управління: Network Controller надає можливість автоматизованого управління мережею. Це означає, що системні адміністратори можуть налаштувати правила та процеси, які дозволяють автоматично виконувати рутинні задачі управління мережею;

– керування безпекою: Network Controller забезпечує керування безпекою мережі, зокрема контроль доступу до ресурсів мережі, захист від атак та вірусів та моніторинг використання мережевої пропускної здатності;

– аналітика мережі: Network Controller дозволяє здійснювати аналіз мережі та надає інформацію про її пропускну здатність, стан мережевих пристроїв та інші параметри, що допомагає забезпечити ефективне використання ресурсів мережі та планування її подальшого розвитку.

– швидке виявлення та усунення проблем: Network Controller дозволяє виявляти проблеми у мережі та автоматично сповіщати про них системних адміністраторів. Це дозволяє швидко виявляти та усувати проблеми, що зменшує час, необхідний для відновлення роботи.

Отже, додавання Network Controller до комп'ютерної мережі дозволяє забезпечити більш ефективне та автоматизоване керування та моніторінг за мережею, що полегшує життя системним адміністраторам та забезпечує більш високу надійність та безпеку мережевих пристроїв та ресурсів.

Існує багато програмних рішень для моніторингу хостів, серверів або мережевих пристроїв, багато з яких вимагають серверних ресурсів для розгортання та налаштування. Але якщо у нас невелика корпоративна мережа, є один або кілька серверів або маршрутизатор із білою IP-адресою, то немає потреби розгортати складну структуру моніторингу лише для того, щоб перевірити, чи доступний хост, сервер чи маршрутизатор. Достатньо створити простого телеграм-бота, який надсилатиме телеграм-повідомлення, якщо перевірений IP недоступний (вниз) і коли він знову стане доступним (вгору).

1.7 Завдання і мета роботи, що виконується

Сучасні комп'ютерні системи базуються на мережі Інтернет або входять до локальної мережі з підключенням до Інтернету. У зв'язку з актуальним станом інформаційно-комунікаційних технологій виникає проблема моніторингу комп'ютерних мереж для збирання інформації про них. Ефективним рішенням є розробка програмного забезпечення для мережевого моніторингу, яке забезпечує доступ до інформації про мережевий контент, топологію та обладнання як для фахівців, так і для користувачів. [6]

В епоху стрімкого науково-технічного прогресу, коли цінність інформації та соціальний статус інформаційних послуг різко зросли, виникає потреба в підтримці працездатності локальних мереж. Підтримка працездатності локальних мереж вимагає великого штату кваліфікованих фахівців, включаючи підтримку доступності мережевих сервісів, оптимізацію продуктивності мережі, конфігурацію та оновлення мережевого обладнання та програмного забезпечення, виявлення та усунення інцидентів та інші функції. IT-персонал не завжди доступний для надання необхідної підтримки та супроводу. Однак альтернативою великому штату IT-спеціалістів є використання спеціалізованого програмного забезпечення, яке називається системою моніторингу та управління мережею. Основною функцією цього типу програмного забезпечення є моніторинг мережі. Це означає, що воно постійно стежить за комп'ютерною мережею, шукаючи повільні або несправні системи та попереджаючи адміністратора, якщо виявлено несправність.

Моніторинг мережі - це можливість контролювати та убезпечити робочі процеси, пов'язані з Інтернетом та комп'ютерами.

При проектуванні кваліфікаційної роботи необхідно виконати вимоги, що описані нижче:

– виконати аналіз зібраної інформації за об’єктом впровадження;

– розробити технічні вимоги на КС;

– обґрунтувати структуру та розробити специфікацію апаратних засобів КС

– розробити макет робочої мережі усієї мережі в Packet Tracer;

– додати в макет Network Controller (NC);

– створити програму для моніторингу хостів, яка сповіщатиме в Telegram, коли мережний пристрій недоступний або приєднався до мережі;

– протестувати передачу та отримання даних з інтеграцією з Telegram.

2 РОЗРОБКА АПАРАТНОЇ ЧАСТИНИ КОМП'ЮТЕРНОЇ СИСТЕМИ КОМПАНІЇ "VAN OMMERED"

2.1 Технічні вимоги до комп'ютерної системи

2.1.1 Вимоги до системи в цілому

2.1.1.1 Вимоги до структури і функціонуванню системи

Комп'ютерна система моніторингу доступності хостів (далі Система) розробляється для комп'ютерної мережі логістичної компанії «Van Ommered» та повинна бути побудована на основі 3-рівневої архітектури мереж SDN (Додаток А, рис. ДА.1).

2.1.1.1.1 Перелік підсистем, їхнє призначення й основні характеристики, вимоги до числа рівнів ієрархії та ступені централізації Системи

– Application layer: загалом виконує завдання з управління мережевою політикою та її моніторінгу з інтеграцією з Telegram;

– Control layer: сервер, на якому розміщений Network Controller, і з якого здійснюється керування та моніторинг мережних пристроїв;

– Infrastructure layer: локальна комп'ютерної мережа (ЛКМ) та мережеві пристрої, відповідальні за пересилання трафіку.

Згідно організаційної структури проект Infrastructure layer ЛКМ компанії "Van Ommeren" повинен складатися з восьми робочих груп (далі Підсистем):

– робоча група №1 – керівництво;

– робоча група №2– адміністративний відділ;

– робоча група №3 - фінансовий відділ;

– робоча група №4 – відділ проектування і дислокації інфраструктури логістичної системи;

– робоча група №5 – відділ координації ланок логістичної системи;

– робоча група №6– відділ маркетингу;

– робоча група №7 – відділ розвитку технологій;

– робоча група №8 – відділ технічної підтримки.

Відповідно до табл.2.2 робочі групи відокремлені в оремі 5 локальні мережі та підтримувати кількість вузлів з майбутнім розширенням.

Таблиця 2.2 – Робочі групи та кількість вузлів в сегменті мережі

№	Умовне позначення	Робоча група	Кількість ПК (поточна/в майбутньому)	Принтери
LAN1	DIRECTION	РГ №1+2	5/5	1
LAN2	DLS	РГ №3+4+5	12/20	1
LAN3	DEVNET	РГ №7	3 / 6	1
LAN4	MARKETING	РГ №6	3 / 10	1
LAN5	ITSUPPORT	РГ №8	2 / 2	1
		Загалом	25 /43	5

В мережі DLS потрібно налаштувати віртуальні локальні мережі для відповідної робочої групи.

ЛКМ повинна мати виділені сервери: файл-сервер, web-сервер та сервер, на якому розміщений Network Controller, і з якого здійснюється керування та моніторинг мережі.

FTP-сервер знаходиться у відділі керівництва. Інформація про мережеве обладнання та всі налаштування зберігаються на цьому сервері.

Web-сервер знаходиться у відділі технічної підтримки, а сервер з Network Controller знаходиться у відділі розвитку технологій.

Загальна топологічна схема розміщення структурних підрозділів зображена на рисунку ДА.2 в Додатку А.

2.1.1.1.2 Вимоги до способів і засобів зв’язку між компонентами комп’ютерної системи

Кожна робоча станція в робочих групах повинна мати доступ до всіх інших робочих груп та серверів на основі технрології FastEthernet по витій парі.

Для звязку між мережами компанії використовувати маршрутизатори і Serial-інтерфейси. Також необхідно реалізувати зв'язок між підмережами на основі протоколу динамічної маршрутизації OSPF.

Разом з цим кожна з підмереж повинна мати можливість доступу до Інтернет. Необхідно організувати підключення до мережі Інтернет на граничному маршрутизаторі за допомогою протоколу NAT.

Підключення сервера, що містить Network Controller, до Інтернету для отримання оновлень та інших необхідних даних.

Наявність надійної мережевої інфраструктури зі стабільним з'єднанням між сервером та мережевими пристроями.

Система моніторингу повинна взаємодіяти між собою по каналах зв'язку Ethernet, по протоколу TCP/IP, звертаючись до REST API відповідних підсистем або їх компонентів та обмінюючись повідомленнями в форматі JSON.

В випадку, коли мережевий адаптер на комп'ютері недоступний, і коли він став знов доступним, надсилати повідомлення в Telegram для оперативного інформування системного адміністратора.

2.1.1.1.3 Вимоги до характеристик взаємозв'язків створюваної системи із суміжними системами

Network Controller повинен підтримувати інтеграцію з різними типами мережевих пристроїв, що дозволяє керувати та моніторити їх за допомогою централізованого інтерфейсу.

Система повинна надавати API для інтеграції з іншими системами, такими як системи моніторингу, системи безпеки, системи автоматизації тощо.

2.1.1.1.4 Вимоги до режимів функціонування системи

Система моніторингу повинна бути надійною і стабільною, здатною працювати без збоїв 24/7. Вона повинна бути здатна виявляти та відновлювати свою працездатність в разі виникнення помилок або відмов.

Мережеве обладнання повинно забезпечувати функціонування системи моніторингу 24/7.

2.1.1.1.5 Вимоги до діагностування системи

Рекомендується дотримуватися документації та рекомендацій виробника для правильного виконання діагностування.

2.1.1.1.6 Перспективи розвитку, модернізації системи

Модернізація системи повинна проводитися шляхом заміни старого обладнання на нове, яке відповідає вимогам програмного та апаратного комплексу системи.

Система повинна забезпечувати можливість розширення функціональності та потужності Network Controller для впровадження нових функцій та збільшення продуктивності.

2.1.1.2 Вимоги до показників призначення

Система повинна здійснювати постійний моніторинг стану мережі, включаючи доступність, пропускну здатність, використання ресурсів, аналіз трафіку тощо.

Корпоративна мережа компанії повинна бути розроблена переважно для передачі інформації між робочими групами, враховуючи вимоги щодо безпеки корпоративної мережі.

2.1.2 Вимоги функцій, виконуваним системою на основі Network Controller

Комп’ютерна система моніторингу доступності хостів повинна мати централізований сервер Network Controller, який забезпечує можливість керування налаштуваннями, конфігурацією та управлінням роботою мережевих пристроїв, таких як маршрутизатори, комутатори, брандмауери тощо. Також система повинна здійснювати постійний моніторинг стану

мережі, включаючи доступність, пропускну здатність, використання ресурсів, аналіз трафіку тощо та в режимі реального часу повідомляти адміністратора через соціальну мережу Telegram.

Комп’ютерна система на основі Network Controller повинна реалізувати наступні функції:

– керування мережевими пристроями: Network Controller повинен забезпечувати можливість керування налаштуваннями, конфігурацією та управлінням роботою мережевих пристроїв, таких як маршрутизатори, комутатори, брандмауери тощо;

– моніторинг мережі: система повинна здійснювати постійний моніторинг стану мережі, включаючи доступність, пропускну здатність, використання ресурсів, аналіз трафіку тощо та в режимі реального часу повідомляти адміністратора через соціальну мережу Telegram;

– аутентифікація та авторизація: система повинна підтримувати механізми аутентифікації та авторизації, щоб забезпечити доступ до функцій Network Controller тільки авторизованим користувачам.

– аналітика та звітність: система повинна надавати функції аналізу даних мережі, створення звітів, графіків та інших інструментів для моніторингу та аналізу продуктивності та ефективності мережі.

Програмна система моніторингу комп‘ютерної мережі має забезпечувати такий функціонал:

– надсилати повідомлення в Telegram, коли мережевий адаптер, доступні на комп'ютері недоступний, і коли він став знов доступним;

– надавати інформацію про мережні пристрої в мережі;

– надавати загальну інформацію про стан мережі;

– надавати інформацію про стан мережі за категорією пристрою.

2.1.3 Вимоги до задач (налаштувань), які виконуються у комп'ютерній системі

Локальна мережа компанії повинна мати приватну адресу з класу С 192.168.36.0/24, при цьому мережі, що сполучають роутери повинні бути з маскою /30, а адресний простір, що залишився треба рівномірно поділити між підмережами кожного маршрутизатора.

Основними видами трафіку в мережі будуть локальний, робота в Інтернет та з файл-сервером.

Встановлене обладнання повинно мати близько 10% запасних портів, на випадок відмови портів або розширення та вдосконалення мережі.

За для роботи Системи, необхідно здійснити наступні кроки з налаштування маршрутизаторів та комутаторів Системи [10]:

– призначити системні найменування кожному пристрою за правилом *тип пристрою_номер пристрою_КВВ* та розробити банер MOTD, а також доменне ім'я, що повторює ім'я цього пристрою;

– призначити пароль *cisco* до консольних та vty ліній на всіх пристроях та пароль *class* для доступу до привілейованого режиму. Ці паролі мають бути зберігатись у зашифрованому вигляді за допомогою ключа RSA завдовжки 1024 біт;

– призначити використання протоколу ssh на всіх vty лініях;

– створити користувача *123191_Kanibolotsky* з паролем *admincisco* на всіх пристроях;

– встановити значення тактової частоти на DCE-інтерфейсах маршрутизаторів 128000;

– на serial-інтерфейсах має бути призначено пропускна спроможність, що дорівнює 128 Кб/с;

– призначити адреси інтерфейсам маршрутизаторів;

– налаштувати роботу протоколу маршрутизації OSPF, оголосивши підключені до нього мережі;

– задати на граничних маршрутизаторах статичні маршрути для доступу у мережу Internet та мережі провайдеру, та поширити цій маршрути на інші маршрутизатори мережі підприємства;

– забезпечити захист інформації в Системі шляхом впровадження та налаштування VLAN на маршрутизаторі, що відповідає за роботу адміністративно-управлінського персоналу;

– налаштувати функції безпеки портів на комутаторах, під’єднаних до серверів так, щоб тільки двом унікальним пристроям було надано дозівл на доступ до порту, їх MAC-адреси динамічно розпізнавались, а під час порушення системи безпеки з’являлось повідомлення, а порт залишався включеним.

– моніторити стан мережних пристроїв та надсилати в Telegram.

– розробити макет робочої мережі в Packet Tracer з Network Controller;

– написати програму на Python для моніторингу хостів, яка сповіщатиме в Telegram, коли мережний пристрій недоступний або приєднався до мережі;

– в чат-боті Telegram розробити додаткові меню, які будуть розширювати функціонал застосунку відповідно до потреб користувача

2.1.4 Вимоги до видів забезпечення комп’ютерної сич'стеми

2.1.4.1 Вимоги до інформаційного забезпечення

Підсистеми, що входять в КС, повинні взаємодіяти між собою по каналах зв'язку Ethernet, по протоколу TCP/IP, звертаючись до REST API відповідних підсистем або їх компонентів та обмінюючись повідомленнями в форматі JSON.

2.1.4.2 Вимоги до технічного забезпечення

Сервер: потужний сервер з достатнім обсягом оперативної пам'яті та процесорною потужністю для підтримки Network Controller та обробки великої кількості даних мережі.

Мережеві пристрої: сумісні маршрутизатори, комутатори та брандмауери, які можуть бути інтегровані з Network Controller і забезпечувати необхідні функції моніторингу та управління мережею.

Монітори: монітори для відображення даних та статусу мережі з Network Controller.

Мережеві кабельні системи: достатня кількість мережевих кабелів для підключення мережевих пристроїв до Network Controller та сервера.

2.1.4.3 Вимоги до програмного забезпечення

Операційна система: підтримка операційної системи, яка є сумісною з Network Controller та іншими компонентами мережі.

Network Controller Software: установлення та налаштування програмного забезпечення Network Controller на сервері для централізованого керування та моніторингу мережі.

Клієнтське програмне забезпечення: якщо необхідно, налаштування клієнтського програмного забезпечення на робочих станціях для доступу до функцій Network Controller.

2.2 Розробка апаратної частини комп’ютерної системи

2.2.1 Вибір та характеристика постачальника рішень для Network Controller

Пристрої мережевого контролера (NC) забезпечують засоби керування, моніторингу та налаштування підтримуваних мережевих пристроїв за допомогою графічного інтерфейсу користувача та API.

В якості рішення було обрано програму Packet Tracer версії 8.2.1, в якій підтримується використання контролера SDN, аналогічного існуючим реальним контролерам SDN, таким як Cisco DNA Center та APIC-EM.

Packet Tracer – симулятор мережі передачі даних, що випускається фірмою Cisco Systems. Дозволяє робити працездатні моделі мережі,

налаштовувати (командами Cisco IOS) маршрутизатори і комутатори, взаємодіяти між декількома користувачами (через хмару).

На базовому рівні можна налаштувати глобальні параметри та параметри інтерфейсу NC так само, як це можна зробити на звичайному ПК. Крім того, доступ до контролерів можна отримати ззовні Packet Tracer через HTTP REST API. Ця функція називається реальним доступом до світу.

NC надає централізовану панель інструментів Dashboard для перегляду *стану мережі, швидкого виявлення та усунення проблем, а також одночасної* відправки змін конфігурації на всі керовані пристрої.

Щоб звертатися до Network Controller API, задля того, щоб полегшити використання деяких функції для адміністратора мережі і застосовувати їх до всієї мережі, є два варіанти рішення:

– графічний інтерфейс контролеру через WEB;

– написання скриптів мовами програмування.

Cisco Packet Tracer безкоштовна для студентів академії Cisco.

Cisco Packet Tracer 8.1 сумісна з такими платформами: Microsoft Windows 8.1, 10, 11 (32-розрядні та 64-розрядні), Ubuntu 20.04 LTS (64-розрядні) та macOS 10.14 або новішої (64-розрядні).

Системні вимоги для Cisco Packet Tracer 8.1:

– операційна система: Microsoft Windows 8.1, 10,11 (64-розрядна версія), Ubuntu 20.04 LTS (64-розрядна версія) або macOS 10.14 або новішої версії.

– amd64(x86-64) CPU;

– 4 GB вільної RAM;

– 1.4 GB вільного місця на диску.

Для написання скриптів обрано мову Python, так як найкраще має можливості для роботи з форматом JSON.

2.2.2 Вибір і обґрунтування структурної схеми комплексу технічних засобів комп’ютерної системи

2.2.3 Розробка специфікації апаратних засобів КС

Кількість використаних пристроїв при побудові комп’ютерної мережі компанії у таблиці 2.3.

Таблиця 2.3 – Пристрої, використані при побудові корпоративної мережі компанії «Van Ommered»

Позиція	Тип пристрію	Кількість	Загальна кількість
1	Комп’ютер (PC)	Основна мережа — 27 Віддалена мережа — 3	30
2	Комутатор	Основна мережа — 6 Віддалена мережа — 1	7
3	Маршрутизатор	Основна мережа — 4 Віддалена мережа — 1	5
4	Сервер	Основна мережа — 2 Основна мережа — 1	3

Замовником було прийнято рішення використовувати обладнання компанії DELL та Cisco. розробка технічних вимог до комп’ютерної або кіберфізичної системи Повні характеристики мережевого обладнання приведені нижче.

Рисунок 2.5 – Комп'ютер Dell Vostro 3681 v15

Таблиця 2.4 – ехнічні характеристики комп'ютера Dell Vostro 3681 v15

Технічні характеристики комп'ютера	Intel Core i5-10400 (2.9 - 4.3 ГГц)/RAM 16 ГБ/HDD 1 ТБ + SSD 480 ГБ/Intel UHD Graphics 630/без ОД/LAN/Wi-Fi/Bluetooth/кардридер/Windows 10 Pro/клавіатура + миша

Продовження табл. 2.4

Процесор	Шестиядерный Intel Core i5-10400 (2.9 - 4.3 ГГц)
Чипсет плати	Intel B460
Обсяг оперативної пам'яті	16 ГБ
Порти На передній панелі:	2 x USB 3.2 Gen 1 Type-A 2 x USB 2.0 Комбінований аудіороз'єм для навушників/мікрофона Кардридер
На задній панелі:	2 x USB 3.2 Gen 1 Type-A2 x USB 2.0 x HDMI 1 x VGA 1 x LAN (RJ45) 1 Гбіт/сек
Потужність БЖ	260 Вт
Відеокарта	Intel UHD Graphics 630
Охолодження	Охолодження ЦП: BOX
Додаткові можливості	Модуль бездротового зв'язку Wi-Fi 802.11ac
Обсяг SSD	480 ГБ
Тип відеокарти	Інтегрована
Обсяг HDD	1 ТБ
Частота ядра	2,9

Рисунок 2.6 – Сервер Dell PowerEdge R440 A13 (per440ceeM02)

Таблиця 2.5 – Технічні характеристики Сервера Dell PowerEdge R440 A13 (per440ceeM02)

Форм-фактор	1U Rackmount
Кількість ядер	8
Тип процесорів	Intel Xeon
Процесор	Восьмиядерний Intel Xeon Silver 4208 (2.1-3.2 ГГц)
Обсяг оперативної пам'яті	16Gb
Частота	2.1 ГГц
Швидкість DIMM:	До 2666 млн транзакцій у секунду

Продовження табл. 2.5

Тип пам'яті:	RDIMM LRDIMM
Слоти для модулів пам'яті:	16 слотів для модулів DDR4 DIMM Підтримуються тільки реєстрові модулі DDR4 DIMM з ECC
Кількість ЦП в комплекті	1
Контролери SAS/SATA	PERC H730P
Рівні RAID	0/1/5/6/10/50/60
Кількість БЖ в комплекті	1
Порти на передній панелі:	1 x виділений USB-порт iDRAC Direct 1 x порт USB 2.0 1 x відеопорт
Порти на задній панелі	1 x виділений мережевий порт iDRAC 1 x послідовний порт 2 x порти USB 3.0 1 x відеопор
Слоти розширення:	1 x PCIe 3.0 x16
Додаткові характеристики	
Передні відсіки дисководів:	До 10 дисків SAS/SATA 2.5" (жорстких дисків або твердотілих накопичувачів) і до 4 твердотілі накопичувачі NVMe макс. місткістю 48 Тбайт
	або:До 4 жорстких дисків SAS/SATA 3.5" макс. місткістю 56 Тбайт Кількість LAN (RJ-45)

Маршрутизатор Cisco 4321 Integrated Services призначений для надання розширених послуг у середовищі філій малого підприємства. За замовчуванням він забезпечує 50 Мбіт/с. Завдяки продуктивності за принципом «оплата за зростанням», можна збільшити пропускну здатність до 100 Мбіт/с, придбавши ліцензії FL-4320-PERF-K9 [12]

Рисунок 2.7 – Маршрутизатор Cisco ISR4321

Таблиця 2.6 – Технічні характеристики Cisco ISR4321/K9

Виробник	Cisco Systems, Inc
Модель	CISCO ISR4321
Form Factor	External - modular - 2U
Розміри (WxDxH)	43.8 cm x 30.5 cm x 8.9 cm
Вага	8.2 kg
DRAM	4 Гб (installed) / 8 GB (max)
Flash	4 Гб (installed) / 8 GB (max)
Загальна кількість WAN та LAN портів	2
Інтегровані порти WAN	1 GE / SFP 1 GE
Aggregate Througput	50 Мб до 100 Мб
Протоколи ма ршрутизації	OSPF, IS-IS, BGP, EIGRP, DVMRP, PIM-SM, IGMPv3, GRE, PIM-SSM, static IPv4 routing, static IPv6 routing

Рисунок 2.8 – Комутатор Cisco WS-C2960X-24PS-L

Таблиця 2.7 – Технічні характеристики комутатора Cisco WS-C2960X-24PS-L

Manufacturer:	Cisco
Product ID:	WS-C2960X-24PS-L
Product Description:	Cisco Catalyst 2960-X 24 GigE PoE 370W, 4 x 1G SFP, LAN Base
Product Type:	Ethernet Switch
Interfaces/Ports	
Total Number of Network Ports:	24
Number of PoE (RJ-45) Ports:	24
Port/Expansion Slot Details:	24 x Gigabit Ethernet Network
	4 x Gigabit Ethernet Expansion Slot
Media & Performance	
Media Type Supported:	Twisted Pair
Twisted Pair Cable Standard:	Category 5
Ethernet Technology:	Gigabit Ethernet

Продовження табл. 2.7

Network Technology:	10/100/1000Base-T
I/O Expansions	
Number of Total Expansion Slots:	4
Expansion Slot Type:	SFP
Shared SFP Slot:	No
Number of SFP Slots:	4
Network & Communication	
Layer Supported:	2
Management & Protocols	
Manageable:	Yes
Management:	Web-based Management;VLAN; QoS; Syslog; Telnet; RMON I, II; SNMP v1, v2c, v3; CLI; DHCP
Memory	
Standard Memory:	512 MB
Memory Technology:	DRAM
Flash Memory:	128 MB

2.2.4 Розрахунок інтенсивності вихідного трафіку найбільшої локальної мережі підприємства

В підмережі DLS встановлений комутатор Cisco2960, що об'єднує 20 ПК працівників. Вихідний трафік з комутатора SW1_DLS надсилається до роутера R1_KBB в лінію з пропускною здатністю, що становить 1000 Мбіт/с.

Для того, щоб комутатор не був перенасичений, швидкість надходження пакетів не повинна перевищувати швидкості їх відправлення. Вважаємо, що послугами одночасно користуються 100% користувачів. Середня інтенсивність трафіку μ=86 (кадрів/с), а середня довжина повідомлення – 1150 байт.

Теоретично припустимо, що всі користувачі найбільшої підмережі DLS одночасно використовують мережу. В такому разі, пропускна здатність на рівні доступу буде дорівнювати:

$$P_{р.р.} = \mu * L_{пов} * N * 8 = 86 * 1150 * 20 * 8 = 15.8 \text{ Мбіт/с} \quad (2.1)$$

де $L_{пов}$ – середня довжина повідомлення;

N – кількість вузлів в мережі.

Отриманий результат не перевищуватиме заданих параметрів мережі по вихідному каналу, отже перенавантажень не трапиться.

Комутатор SW1_DLS також передає трафік до маршрутизатора зі швидкістю 1000 Мбіт/с. Отже, загальне навантаження на комутатор не повинно перевищувати:

$$\mu_{вих} = 10^9 / (1150 * 8) = 108\ 696 \text{ пакетів/с} \quad (2.2)$$

Оскільки в середньому, кожне джерело виробляє 86 пакетів/с, то маршрутизатор обмежений кількістю приєднань, яку ми можемо дізнатись наступним чином:

$$N = \mu_{вих} / \mu = 108\ 696/ 86 \approx 1264 \text{ джерела} \quad (2.3)$$

Ця кількість задовольняє кількості вузлів у найбільшій нашій локальній мережі, до якої входить 20 ПК.

Кожен з 20 ПК посилає потік заявок з інтенсивністю у 86 кадрів/с. Звідси, можем розрахувати інтенсивність вихідного трафіку:

$$\lambda = N * \mu = 20*86 = 1720 \text{ пакетів/с.} \quad (2.4)$$

Коефіцієнт затримки:

$$\rho = \frac{\lambda}{\mu вих} = \frac{1720}{108696} = 0{,}016 \quad (2.5)$$

Коефіцієнт зайнятості маршрутизатора:

$$\frac{\rho}{1-\rho} = \frac{0{,}016}{1-0{,}016} = 0{,}016 \quad (2.6)$$

Середня затримка кадру, пов’язана з чергою M/M/1, дорівнює:

$$T = \frac{1}{(\mu-\lambda)} = \frac{1}{(108696 - 1720)} = 9{,}35 \text{ мкс} \quad (2.7)$$

Середня довжина черги:

$$L_{чер} = \frac{\rho^2}{1-\rho} = \frac{0{,}016^2}{1-0.016} = 0.0026 \quad (2.8)$$

Середній час перебування пакета у черзі:

$$T_{оч} = \frac{Lчер}{\lambda} = \frac{0{,}026}{1720} = 0{,}15 \text{ мкс} \quad (2.9)$$

Це значення задовольняє вимогам до затримки в ЛМ.

3 ПРОЕКТУВАННЯ КОРПОРАТИВНОЇ МЕРЕЖІ ТА ПЕРЕВІРКА РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ КОМПАНІЇ «VAN OMMEREN»

3.1 Моделювання мережі в Packet Tracer з Network Controller

В ході кваліфікаційної роботи було розроблено топологію мережі компанії "Van Ommered" у Packet Tracer на основі SDN контролеру, а також розроблено вивід стану усієї мережі та стану кожного пристрою у Dashboard, написано програму керуванням мережею за допомогою API у додатку Python.

При розробці топології мережевих пристроїв у Packet Tracer, було використано: NC, роутери ISR 4321, комутатори 2960, пристрої-хости. На рис.3.1 зображено топологію мережі.

Доступ до контролера Packet Tracer можна отримати з реальної мережі та запрограмувати через його веб-API за допомогою Python від Visual Studio Code (VS Code).

Зв'язок між Cisco Network Controller (NC) і телеграм-ботом може бути здійснений за допомогою програми, яка використовуватиме API Telegram та API Network Controller (NC).

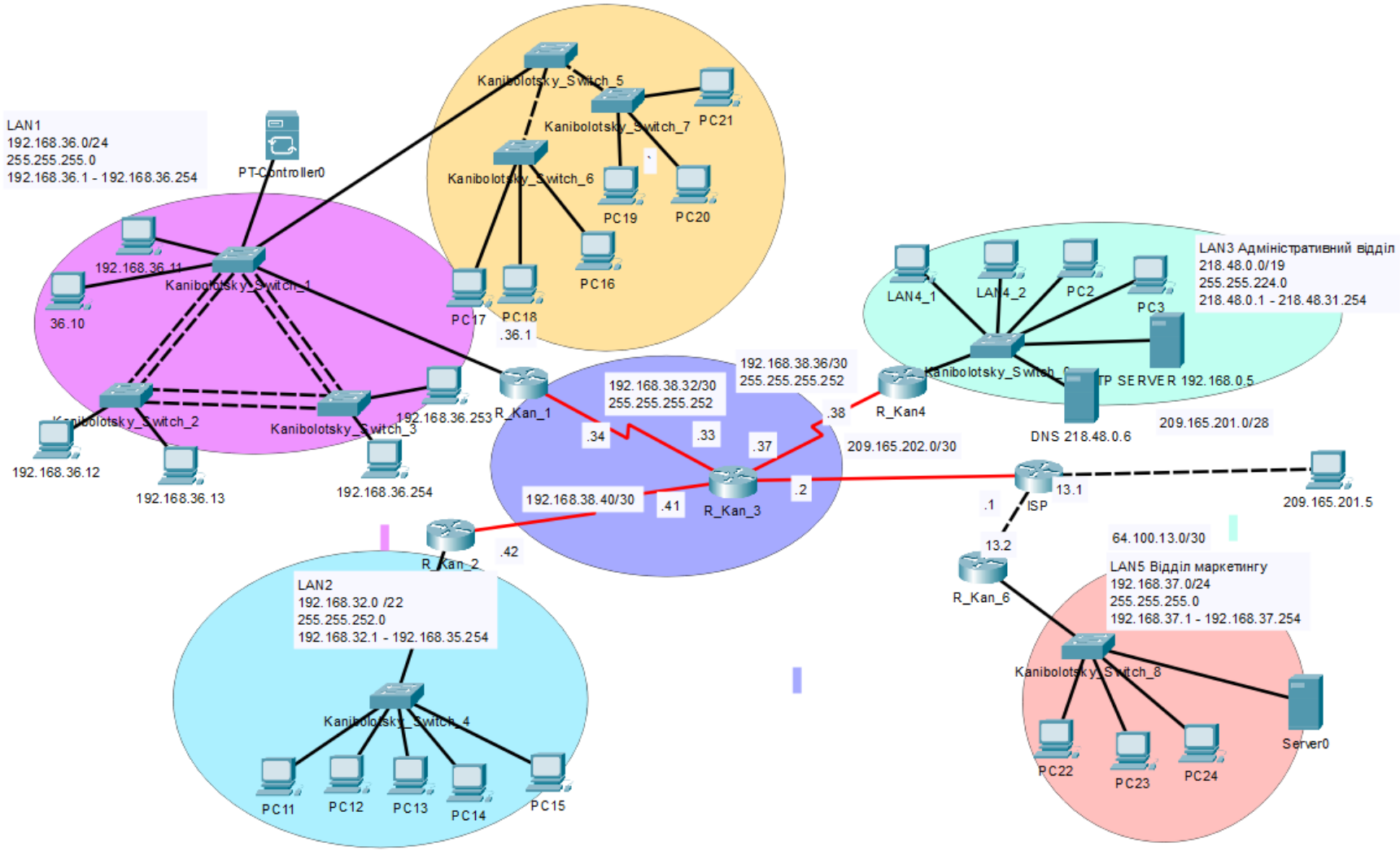

Рисунок 3.1 – Топологія мережі з використанням SDN

3.2 Розрахунок схеми адресації корпоративної мережі

Для виконання адресації мережі застосований метод VLSM. Метод VLSM – це технологія, яка дозволяє адміністратору розбивати адресний простір IP мережі на підмережі нерівних розмірів, на відміну від розбиття з маскою однакової довжини. Застосування такого методу масок надає можливість заощаджувати адреси

Для розрахунку IP-адресації спочатку необхідно визначити кількість підмереж у даній мережі. Згідно з завданням необхідно розділити вихідну мережу 192.168.36.0/24 на 5 локальних підмереж (LAN1 – LAN5) та 4 WAN мережі між маршрутизаторами.

Для правильного розрахунку методом необхідно мережі відсортувати від найбільшої за кількістю необхідних IP-адрес до найменшої.

Розрахунок схеми IP-адресації послідовних каналів між маршрутизаторами виконується аналогічно.

В таблиці 3.1 представлена розрахована схема 1P-адресації мережі за методом VLSM.

Таблиця 3.1 – Схема адресації мережі

Назва підмережі	Необхідна кіл-ть вузлів	Адреса підмережі	Маска Підмережі у Десятковому форматі	Діапазон допустимих IP-адрес вузлів
DIRECTION	5	192.168.36.56	255.255.255.224	192.168.36.1 - 192.168.36.30
DLS	60	192.168.36.128	255.255.255.192	192.168.36.129 - 192.168.36.190
DEVNET	6	192.168.36.48	255.255.255.248	192.168.36.49 - 192.168.36.54
MARKETING	10	192.168.36.32	255.255.255.240	192.168.36.33 - 192.168.36.46
ITSUPPORT	3	192.168.36.64	255.255.255.248	192.168.36.65 - 192.168.36.70
WAN1	2	192.168.36.76	255.255.255.252	192.168.36.73 - 192.168.36.74
WAN2	2	192.168.36.80	255.255.255.252	192.168.36.77 - 192.168.36.78
WAN3	2	192.168.36.84	255.255.255.252	192.168.36.81 - 192.168.36.82
WAN4	2	192.168.36.88	255.255.255.252	192.168.36.85 - 192.168.36.86

Топологія мережі була виконана в середовищі програмного забезпечення Cisco Packet Tracer під час її розробки були використані адреси пристроїв що наведені в таблиці 3.2.

Таблиця 3.2 – Схема адресації пристроїв

Пристрій	Інтерфейс	IP-адреса	Маска	Шлюз
KBB_Switch_0	—	192.168.38.2	255.255.255.224	192.168.38.1
KBB_Switch_1	—	192.168.36.2	255.255.255.0	192.168.36.1
KBB_Switch_2	—	192.168.36.3	255.255.255.0	192.168.36.1
KBB_Switch_3	—	192.168.36.4	255.255.255.0	192.168.36.1
KBB_Switch_4	—	192.168.32.2	255.255.252.0	192.168.32.1
KBB_Switch_5	—	192.168.38.2	255.255.255.224	192.168.38.1
KBB_Switch_6	—	192.168.38.3	255.255.255.224	192.168.38.1
KBB_Switch_7	—	192.168.38.4	255.255.255.224	192.168.38.1
KBB_Switch_8	—	192.168.37.2	255.255.255.0	192.168.37.1
PC-PT 209.165.201.5	G0/1	209.165.201.5	255.255.255.240	209.165.201.1
KBB_Router_1	G0/0	192.168.36.1	255.255.255.0	
	S0/2/0	192.168.38.34	255.255.255.252	
KBB_Router_3	G0/0	192.168.32.2	255.255.252.0	
	G0/1	192.168.38.1	255.255.255.224	
	S0/0/0	192.168.38.41	255.255.255.252	
	G0/1/0	209.165.202.2	255.255.255.252	
	S0/2/0	192.168.38.37	255.255.255.252	
	S0/2/1	192.168.38.33	255.255.255.252	
KBB_Router_2	G0/0	192.168.32.1	255.255.252.0	
	S0/2/0	192.168.38.42	255.255.255.252	
KBB_Router_4	G0/0	192.168.38.1	255.255.255.224	
	S0/2/0	192.168.38.38	255.255.255.252	
KBB_Router_5	G0/0	64.100.13.1	255.255.255.252	
	G0/1	209.165.201.1	255.255.255.240	
	G0/1/0	209.165.202.1	255.255.255.252	
KBB_Router_6	G0/0	64.100.13.2	255.255.255.252	
	G0/1	192.168.37.1	255.255.255.0	

Перед нами поставлене завдання: налаштувати маршрутизатори, комутатори і комп'ютери для підтримки з'єднання IPv4, виконати базовий захист прристроїв.

Також необхідно налаштувати маршрутизатори для використання OSPF, DHCP а також динамічного NAT.

Протокол OSPF використано у мережі тому, що він підтримує мережні маски змінної довжини, а це надає перевагу при створенні мережі, висока швидкість збіжності, відсутність обмежень досяжності, оптимальний вибір шляху маршрутизації.

3.3 Налаштування основних параметрів пристроїв

Перше, що необхідно при базовому налаштуванні мережі – це задання імені пристроям за правилом *тип пристрою_номер_Прізвище*, тобто R1_KBB:

```
Router>en
Router#configure terminal
Router#hostname R1_KBB
```

Для безпечного доступу до пристроїв необхідно задати на всіх пристроях пароль до консолі і vty cisco, а також задати пароль до привілейованого режиму class та зашифрувати всі паролі, що зберігаються у відкритому виді. Також налаштовуємо банер, що буде відображатись при підключенні до маршрутизатора:

```
R3_KBB(config)#line con 0
R3_KBB(config-line)#password cisco
R3_KBB(config-line)#login
R3_KBB(config-line)#line vty 0 4
R3_KBB(config-line)#password cisco
R3_KBB(config-line)#login
R3_KBB(config)#enable secret class
R3_KBB(config)#service password-encryption
R3_KBB(config)#banner motd
```

Також за допомогою протокола Telnet можна отримати доступ до консолі маршрутизатора. Дає змогу передавати дані у відкритому вигляді, або SSH захищеного з’єднання.

При налаштуванні створимо користувача 123_19z1Kanibolotsky з паролем cisco. Для шифрування даних створюємо ключ RSA довжиною 1024 біт.

```
R3_KBB(config)#usemame 123_19z1Kanibolotsky password cisco
R3_KBB(config)#ip domain-name Kanibolotsky.com
R3_KBB(config)#crypto key generate rsa
R3_KBB(config)# 1024
R3_KBB(config)#line vty 0 15
R3_KBB(config-line)#login local
R3_KBB(config-line)#transport input ssh
R3_KBB(config-line)#exit
```

Налаштуємо IPv4-адреси відповідно до таблиці 2, наприклад, на

```
R3_KBB(config)#int g0/0
R3_KBB(config-if)#ip address 192.168.36.2  255.255.252.0
R3_KBB(config-if)#no shutdown
R3_KBB(config-if)#exit
```

На DCE-інтерфейсах маршрутизаторів встановлюємо значення тактової частоти - 128000

```
R3_KBB(config)#int g0/1/0
R3_KBB(config-if)#ip add 209.165.202.2 255.255.255.252
R3_KBB(config-if)#clock rate 128000
```

3.4 Налаштування мереж VLAN, маршрутизації між VLAN

Найбільша є мережа DLS, в який працюють з відділів: фінансовий відділ, відділ проектування і дислокації інфраструктури логістичної системи і відділ координації ланок логістичної системи. В цій мережі, побудованій на комутаторах, необхідно реалізувати однакові за розміром мережі VLAN і маршрутизацію між VLAN.

Розрахунок IP-адресації цієї мережі методом VLSM представлено в табл.3.3:

Таблиця 3.3 – Адресація мереж VLAN

Name	Network Address	Slash	Mask	Usable Range
VLAN10	192.168.36.128	/28	255.255.255.240	192.168.36.129 - 192.168.36.142
VLAN20	192.168.36.144	/28	255.255.255.240	192.168.36.145 - 192.168.36.158
VLAN30	192.168.36.160	/28	255.255.255.240	192.168.36.161 - 192.168.36.174
VLAN40	192.168.36.176	/28	255.255.255.240	192.168.36.177 - 192.168.36.190

Для створення аналогічної мережі без технології VLAN було б потрібно використати відповідну кількість окремих комутаторів, а для зміни логічної структури мережі довелося б також міняти її фізичну структуру.

При побудові мережі на двох чи більше комутаторах, вони з’єднуються між собою кабелем. Порти, через які здійснюється таке з'єднання, мають належати до відповідних віртуальних локальних мереж.

3.5 Налаштування маршрутизації

Маршрутизація – процес визначення маршруту прямування інформації між мережами. Маршрутизатор приймає рішення, що базується на IP-адресі отримувача пакету. Для того, щоб переслати пакет даті, всі пристрої на шляху слідування використовують IP-адресу отримувача. Для прийняття правильного рішення маршрутизатор має знати напрямки і маршрути до віддалених мереж.

Є два типи маршрутизації:

– статична маршрутизація — маршрути задаються вручну адміністратором.

– динамічна маршрутизація — маршрути обчислюються автоматично за допомогою протоколів динамічної маршрутизації— RIP, OSPF, E1GRP, BGP, HSRP та ін, які отримують інформацію про топологію і стан каналів зв'язку від інших маршрутизаторів у мережі.

Відповідно до завдання у мережі використовується протокол динамічної маршрутизації OSPF.

Приклад налаштування OSPF на маршрутизаторі R3_KBB:b

Вмикаємо OSPF на маршрутизаторі, задаємо його унікальний ідентифікатор, змінюємо еталонну пропускну спроможність для обчислення вартості за замовчуванням для дозволу інтерфейсів GigabitEthemet на значення 1000 та призначаємо безпосередньо підключені локальні мережі:

```
R3_KBB(config)#router ospf 9
R3_KBB(config-router)#auto-cost reference-bandwidth 1000
R3_KBB(config-router)#network 192.168.36.0 0.0.0.255 area 0
R3_KBB(config-router) #default-information originate
R3_KBB(config-router) # ip route 0.0.0.0 0.0.0.0 g0/1/0
R3_KBB(config-router)#exit
```

Відключаємо поширення оновлень маршрутизації на інтерфейси в локальні мережі:

```
R3_KBB(config)#router ospf 9
R3_KBB(config-router)#passive-interface g0/0
R3_KBB(config-router)#passive-interface g0/1
 R3_KBB(config)#end
```

Задаємо пропускну спроможність на Serial-інтерфейсах, що дорівнює 128 Кб/с, а вартість метрики задаємо 7500:

```
R3_KBB(config)#int s0/0/0
R3_KBB(config-if)#bandwidth 128
R3_KBB(config-if)#ip ospf cost 7500
R3_KBB(config-if)#int s0/2/0
R3_KBB(config-if)#band width 128
R3_KBB(config-if)#ip ospf cost 7500
R3_KBB(config-if)#int s0/2/1
R3_KBB(config-if)#band width 128
R3_KBB(config-if)#ip ospf cost 7500
```

Налаштовуємо на ньому статичний маршрут та ручне підсумовування, щоб протокол маршрутизації підсумовував тільки підмережі організації:

```
R3_KBB(config-router)#ip route 0.0.0.0 0.0.0.0 209.165.200.3
R3_KBB(config-router)#default-information originate
```

Задаємо вказані у завданні значення пропускної спроможності на Serial-інтерфейсах та вартість метрики:

```
R3_KBB(config)#int s0/0/0
R3_KBB(config-if)#bandwidth 128
R3_KBB(config-if)#ip ospf cost 7500
R3_KBB(config)#router ospf 9
R3_KBB(config-router)#auto-cost reference-bandwidth 1000
R3_KBB(config-router)# network 192.168.38.32 0.0.0.3 area 0
```

Для перевірки налаштування OSPF переглянемо вміст таблиць маршрутизації на кожному маршрутизаторі. Кожен з них має відомості про свої під'єднанні до нього мережі (мітка C), відомості про віддалені мережі (мітка O), та маршрут за замовчуванням до R3_KBB, створений через повідомлення OSPF (мітка O*E2) (рис. 3.2-3.3).

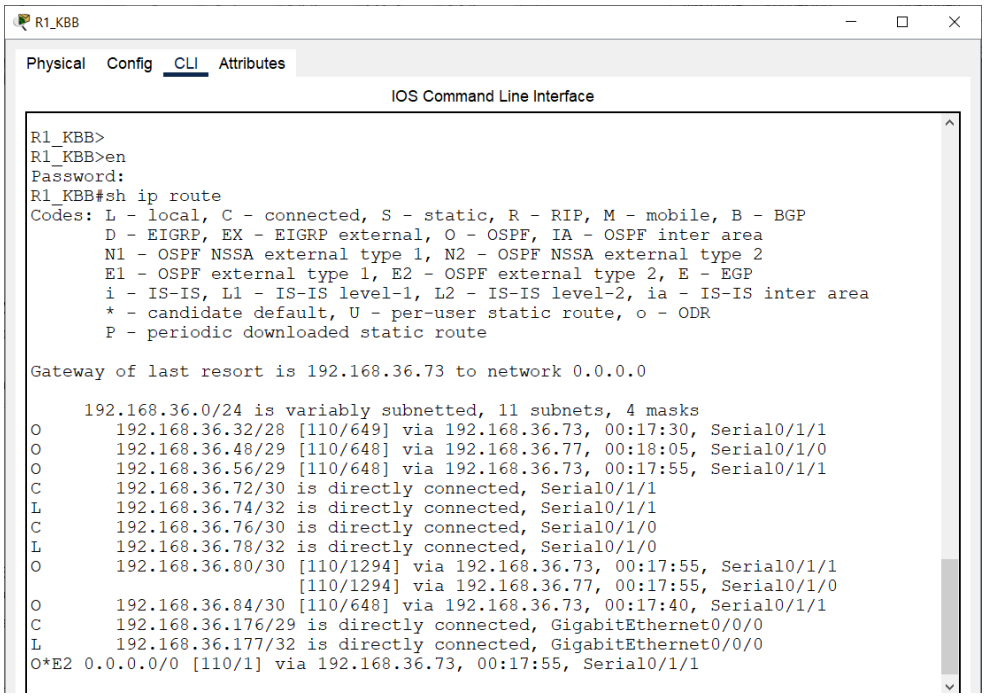

Рисунок 3.2 – Таблиця маршрутизації на R1_KBB

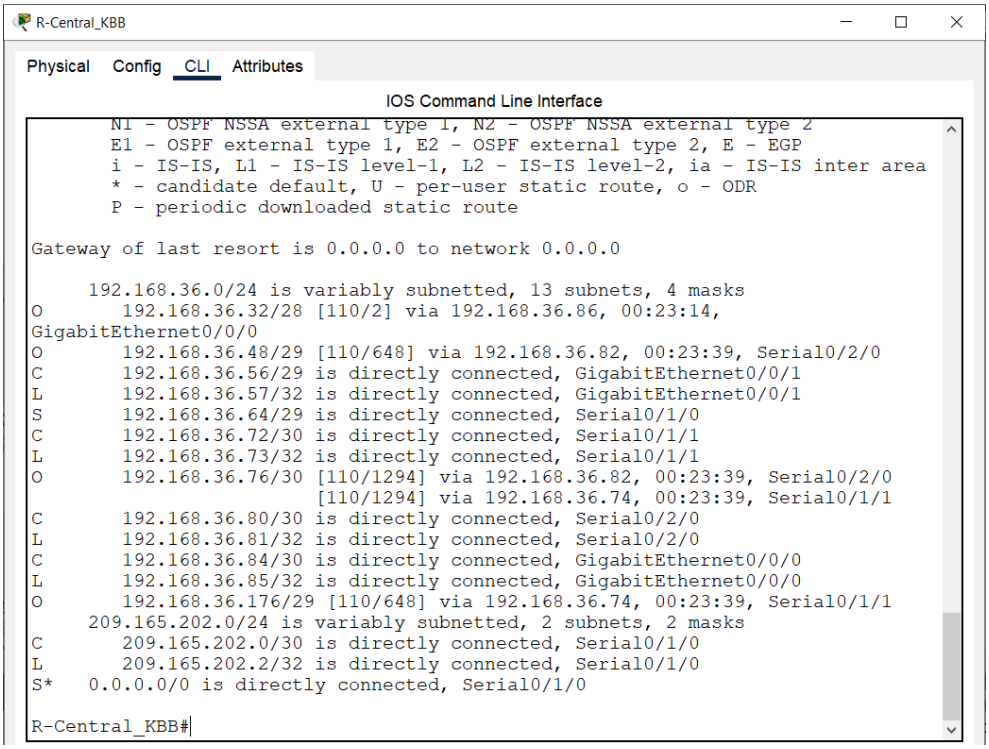

Рисунок 3.3 – Таблиця маршрутизації на R-Central_KBB

Перевіримо шлях командою tracert з вузла в мережі LAN2 в LAN3 (рис. 3.4).

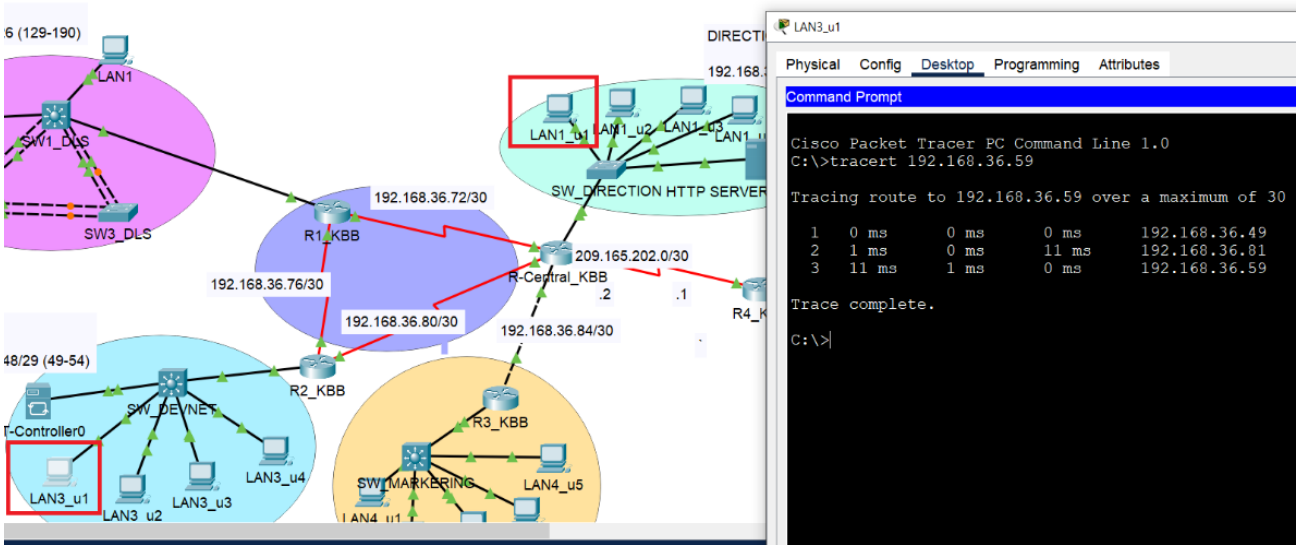

Рисунок 3.4 – Перевірка шляху між LAN3_u1 та LAN1_u1

3.6 Налаштування та перевірка DHCP та NAT

DHCP (Dynamic Host Configuration Protocol — протокол динамічної конфігурації вузла) — це мережний протокол, що дозволяє комп'ютерам автоматично одержувати IP-адресу й інші параметри, необхідні для роботи в комп'ютерній мережі TCP/IP. Для цього комп'ютер звертається до спеціального сервер), піл назвою сервер DHCP. Мережевий адміністратор може задати діапазон адрес, що розподіляють серед комп'ютерів. Це дозволяє уникнути ручного настроювання комп'ютерів мережі N зменшує кількість помилок. Протокол DHCP використовується в більшості великих мереж TCP/IP.

Було прийнято рішення в мережі LAN5 адресацію користувачів зробити по DHCP. В мережу був доданий сервер і налаштовано службу DHCP (рис.3.5).

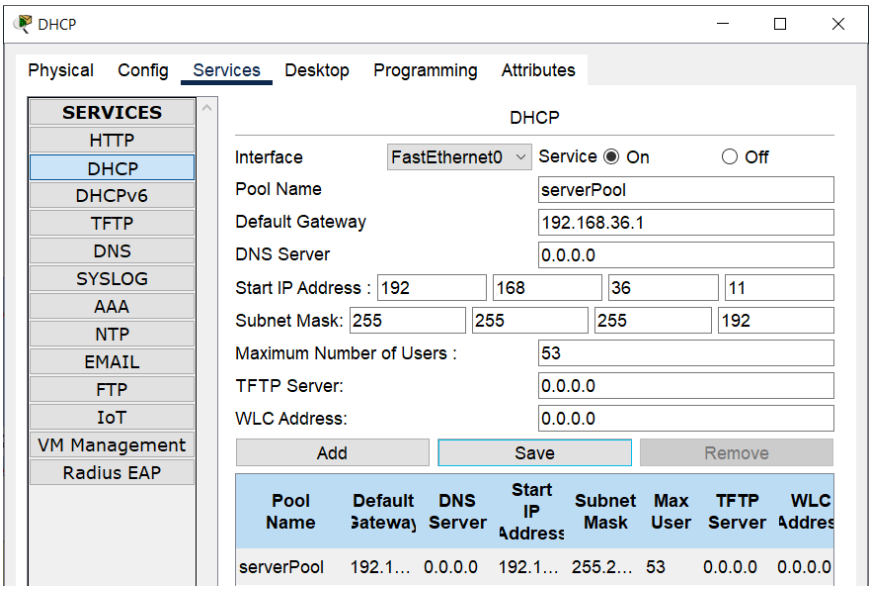

Рисунок 3.5 – Налаштування DHCP-сервера

Щоб перевірити його працездатність, на ПК в налаштуваннях IP вказуємо по DHCP. Вузли успішно отримують адреси з діапазону 192.168.36.1–192.168.36.190 (рис.3.6).

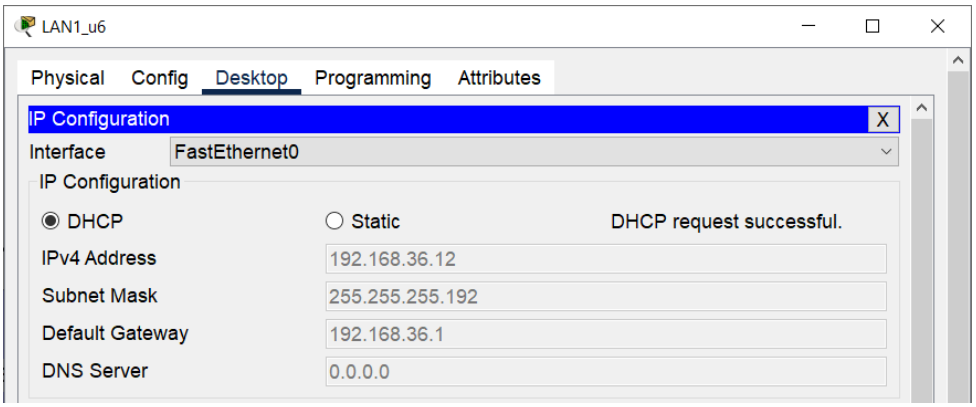

Рисунок 3.6 – Мережні налаштування ПК по DHCP

NAT - це механізм в мережах TCP/IP, що дозволяє перетворювати IP-адреси транзитних пакетів. Завдяки NAT можна, використовуючи одну або кілька зовнішніх IP-адрес, виданих провайдером, підключити до мережі практично будь-яку кількість комп'ютерів.

Більшість маршрутизаторів дозволяють виконувати трансляцію адрес, завдяки чому їх можна використовувати для підключення невеликих мереж до інтернету, використовуючи одну зовнішню IP-адресу.

Постачальник послуг Інтернету ISP виділив для організації діапазон публічних IP-адрес 209.165.202.128/29. Цей діапазон надає організації 6 публічних IP-адрес (209.165.202.129-134). IT-відділ надав наступну інформацію для перетворення наданих IP-адрес (табл. 3.4):

Таблиця 3.4 – Вихідні дані для NAT

Inside local	Inside global	Тип NAT
HTTP-server (192.168.0.5)	209.165.202.129	статичний
DNS – server (192.168.0.6)	209.165.202.130	статичний
LAN1-LAN5	209.165.202.134	PAT

Статичний маршрут використовується на ділянці від ISP до R-Central_KBB, тоді як маршрут за замовчуванням використовується на ділянці від R-Central_KBB до ISP. Створимо статичний маршрут на маршрутизаторі ISP до діапазону виділених публічних IP-адрес 209.165.202.128/29 організації.

ISP(config)#ip route 209.165.202.128 255.255.255.248 g0/1/0

Статичний NAT задає однозначну відповідність однієї адреси іншій. Іншими словами, при проходженні через маршрутизатор, адреса(и) змінюються на строго задану адресу, один-до-одного. Запис такої трансляції зберігається необмежено довго, поки є рядок в конфігурації.

Згідно табл. 3.4 виконаємо статичне перетворення, завдяки чому користувачі зможуть отримати доступ до серверів з Інтернету.

На граничному маршрутизаторі R-Central_KBB введено:

R-Central_KBB (config)#ip nat inside source static 192.168.36.10 209.165.202.129

Всі інтерфейси, підключені до підмереж організації, налаштовані як внутрішні інтерфейси NAT (рис. 3.7).

```
:
interface GigabitEthernet0/0/0
 ip address 192.168.36.85 255.255.255.252
 ip nat inside
 duplex auto
 speed auto
!
interface GigabitEthernet0/0/1
 ip address 192.168.36.57 255.255.255.248
 ip nat inside
 duplex auto
 speed auto
!
interface Serial0/1/0
 ip address 209.165.202.2 255.255.255.252
 ip nat outside
!
interface Serial0/1/1
 ip address 192.168.36.73 255.255.255.252
 ip nat inside
!
interface Serial0/2/0
 ip address 192.168.36.81 255.255.255.252
 ip nat inside
```

Рисунок 3.7 – Налаштування внутрішніх інтерфейсів NAT

Інтерфейс підключення до ISP як зовнішній інтерфейс NAT.

Динамічний NAT з перевантаженням (PAT) виконує трансляцію багато-в-один, задіюючи при цьому можливості транспортного рівня.

Створено іменований ACL-список назвою VAN-NAT, відповідний IP-адресам мереж LAN1-LAN5 (192.168.36.0/24)

ip access-list standard VAN-NAT

permit 192.168.36.0 0.0.0.255 any

Задано пул PUBLIC_PAT придатних до використання публічних IP-адрес.

ip nat pool NAT 209.165.202.1 209.165.202.1 netmask 255.255.255.252

Створено трансляцію NAT, зіставши ACL-список ACL_PAT з пулом зовнішніх адрес PUBLIC_PAT з параметром overload.

ip nat inside source list VAN-NAT pool NAT overload

Для перевірки з командного рядка ПК в кожній із мереж LAN1-LAN5 відправимо ехо-запити на PC-PT (209.165.201.5). Відобразимо перетворення NAT на маршрутизаторі R-Central_KBB за допомогою команди «show ip nat translations» (рис.3.8).

```
R-Central_KBB#sh ip nat translations
Pro  Inside global     Inside local       Outside local      Outside global
icmp 209.165.202.1:32541192.168.36.50:32541192.168.36.67:32541192.168.36.67:32541
icmp 209.165.202.1:32542192.168.36.50:32542192.168.36.68:32542192.168.36.68:32542
icmp 209.165.202.1:32566192.168.36.50:32566192.168.36.67:32566192.168.36.67:32566
icmp 209.165.202.1:32567192.168.36.50:32567192.168.36.68:32567192.168.36.68:32567
icmp 209.165.202.1:32568192.168.36.50:32568192.168.36.69:32568192.168.36.69:32568
icmp 209.165.202.1:32569192.168.36.50:32569192.168.36.70:32569192.168.36.70:32569
icmp 209.165.202.1:32574192.168.36.50:32574192.168.36.12:32574192.168.36.12:32574
icmp 209.165.202.1:32579192.168.36.50:32579192.168.36.67:32579192.168.36.67:32579
icmp 209.165.202.1:32580192.168.36.50:32580192.168.36.68:32580192.168.36.68:32580
icmp 209.165.202.1:32581192.168.36.50:32581192.168.36.69:32581192.168.36.69:32581
icmp 209.165.202.1:32582192.168.36.50:32582192.168.36.70:32582192.168.36.70:32582
icmp 209.165.202.1:32583192.168.36.50:32583192.168.36.12:32583192.168.36.12:32583
icmp 209.165.202.1:32593192.168.36.50:32593209.165.202.1:32593209.165.202.1:32593
icmp 209.165.202.1:32594192.168.36.50:32594192.168.36.66:32594192.168.36.66:32594
```

Рисунок 3.8 – Перевірка роботи PAT

4 РОЗРОБКА ПІДСИСТЕМИ МОНІТОРИНГУ ДОСТУПНОСТІ ХОСТІВ

4.1 Проектування мережі на основі Network Controller

З блоку *End Devises* пристрій *Network Controller*. додається в мережу DEVNET та з'єднується через кабель Straight-Throuhg порт Gig0 на NC з вільним портом на комутаторі (рис. 4.1).

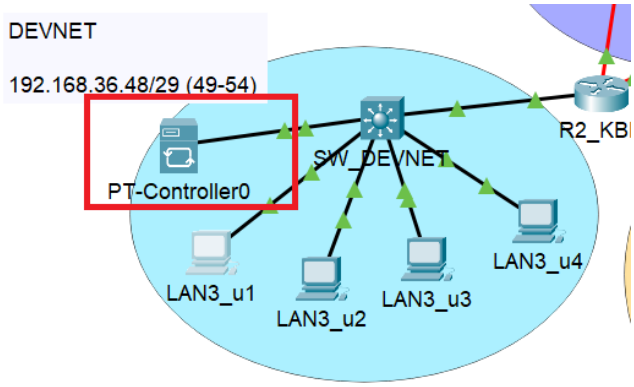

Рисунок 4.1 – Побудова мережі з Network Controller

Надається NC статична незайнята IP-адресу з діапазону допустимих адрес в цій мережі, Default Gateway та DNS Server.

4.1.2 Початкова конфігурація Network Controller

Щоб отримати доступ і керувати NC за допомогою веб-інтерфейсу, користувач повинен мати обліковий запис користувача рівня «administrator». Під час першого доступу до веб-інтерфейсу NC буде запропоновано виконати початкове налаштування, створивши такий обліковий запис, як показано нижче.

Доступ до веб-інтерфейсу можна отримати як з будь-якого ПК в Packet Tracer на вкладці *Desktop* обрати *Web Browser*. В полі URL вказати IP-адресу NC та натиснути Go (рис. 4.2).

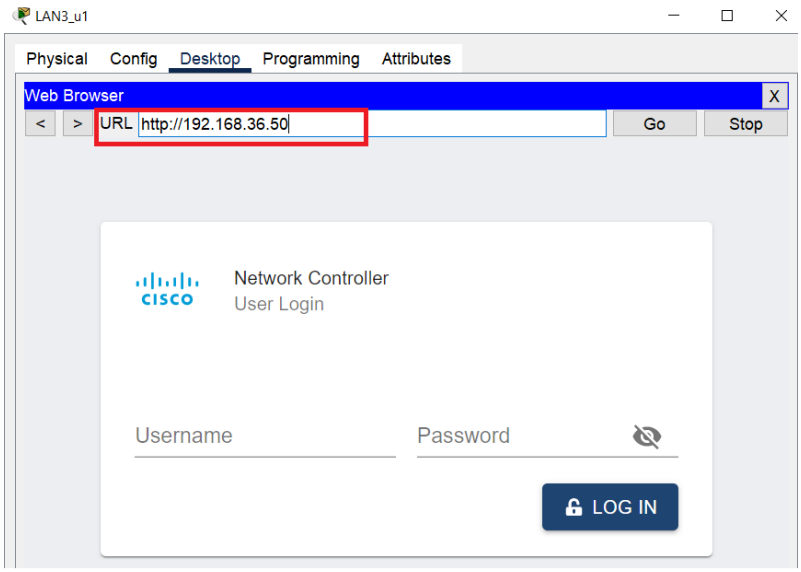

Рисунок 4.2 – Веб-сайт Network Controller з Packet Tracer

Або в веб-браузері на комп'ютері розробника вести loopback-адресу з налаштованим номер порта N (http://127.0.0.1:58000) (рис. 4.3).

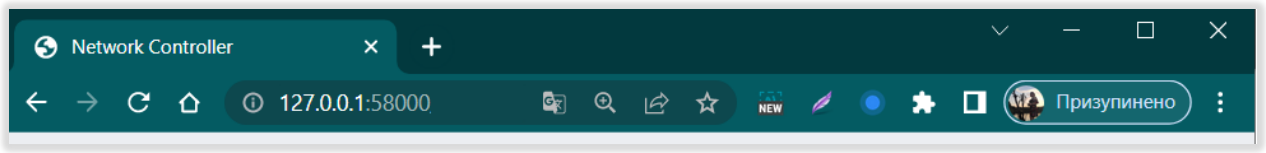

Рисунок 4.3 – Веб-сайт Network Controller з браузера

4.1.3 Додавання облікових даних для доступу до всіх мережевих пристроїв у топології

Після входу у NC тепер доступні усі дані про стан мережі, її топологія, адреси, проходження адрес, та здатність конфігурувати пристрої у мережі. Це так званий Dashboard.

На даному кроці відомості відсутні, тому налаштуємо NC для автоматичного виявлення пристроїв з використанням протоколу Cisco Discover Protocol (CDP).

Для віддаленого підключення до мережних пристроїв на них повинні бути виконані базові налаштування: назва пристрою, доступ до vty-ліній та пароль до привілейованого режиму, а на комутаторі ще IP-адреса та шлюз.

Важливо, щоб були відповідні налаштування на всіх маршрутизаторах та комутаторах:

```
enable
configure terminal
hostname SW_LAN4
username admin secret cisco123
ip domain-name labkm.com
crypto key generate rsa
1024
enable secret cisco
line vty 0 4
login local
transport input ssh
exit
```

// привласнення другої IP-адреси *на комутаторі* із відповідної мережі

```
interface vlan1
ip address 10.160.12.2 255.255.252.0
no shutdown
exit
```

В розділі *Provisioning* необхідно додати облікові дані користувача, які задавались в базовій конфігурації пристроїв, для використання в завданнях автоматизації (рис. 4.4).

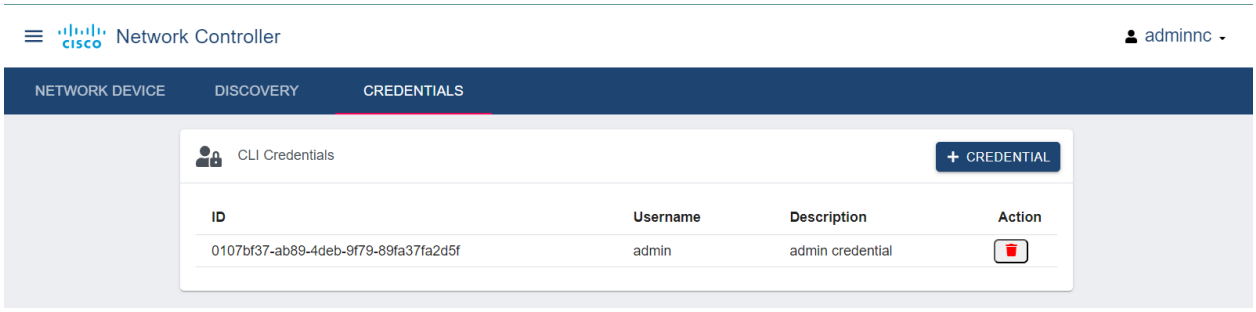

Рисунок 4.4 – Відомості про облікові дані CLI

4.1.4 Використання NC для збору інформації

В меню PROVISION на вкладці *DISCOVERY* виконується виявлення пристроїв в мережі (рис. 4.5).

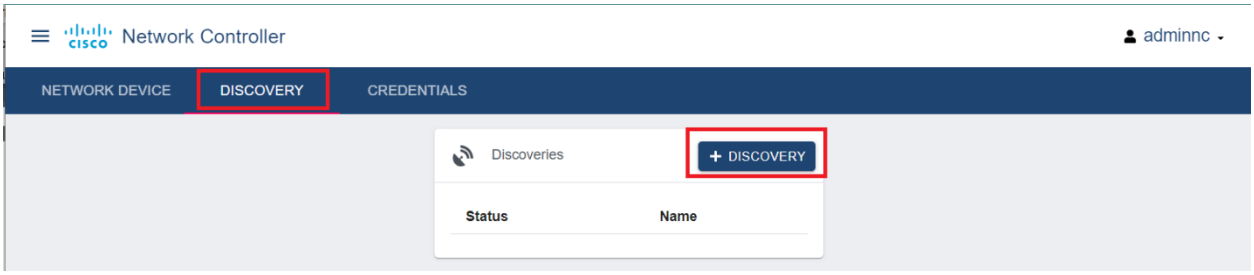

Рисунок 4.5 – Вкладка DISCOVERY

Необхідно надати назву пошуку, вказати IP-адресу шукаючого пристрою, обрати створений Credential, натиснути *ADD*.

Щоб переглянути список виявлених мережевих пристроїв натиснути NETWORK DEVICE.

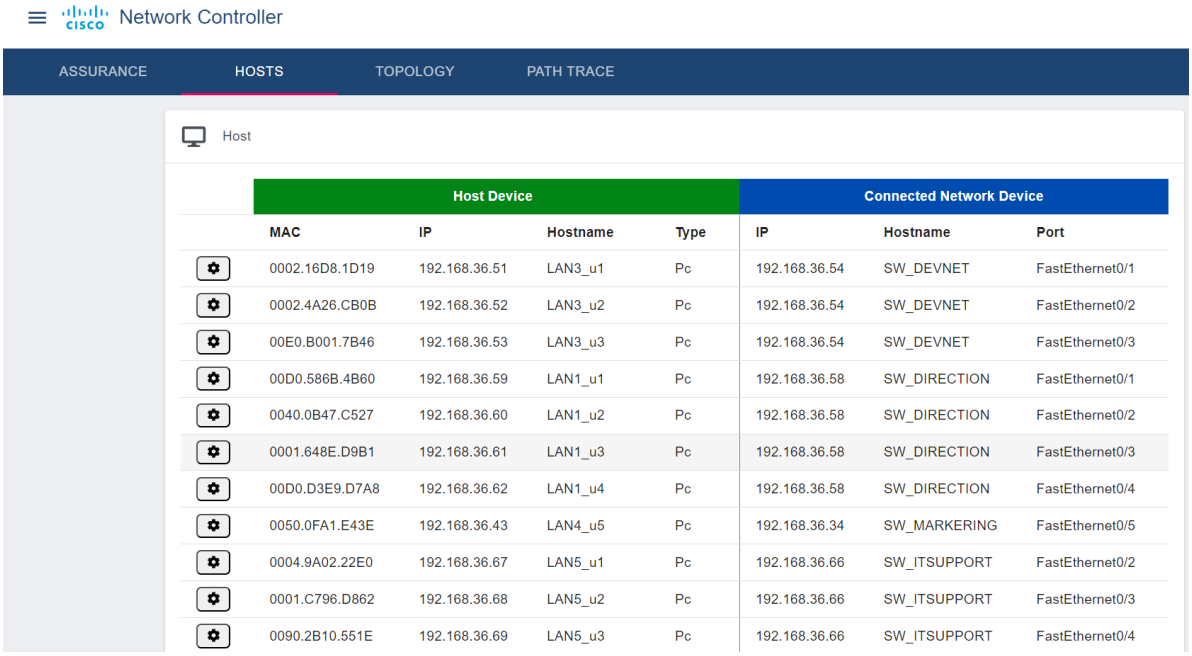

Рисунок 4.6 – Вікно зі списком мережевих пристроїв

Розділ Hosts містить інформацію про усі мережеві та хост пристрої, її повну інформацію (адресу, назву, Mac-адресу, порт). Також їх можна конфігурувати, або видаляти. На цій сторінці можна переглянути всю інформацію про під'єднання Рівня 2 і Рівня 3 для кожного вузла, а також пристрою мережі, до якого кожний з них приєднаний. Значок «Шестерні» поруч з будь-яким вузлом можна переглянути більш детальну інформацію.

4.2 Розробка програми моніторингу доступності хостів

4.2.1 Загальні відомості

Програма включає функціональність для взаємодії з API NC. Вона включає створення HTTP-запитів до NC API для отримання інформації про хости, кількості пристроїв, стан мережі, відомості про мережні присрої тощо.

Програма підключається до API Telegram, використовуючи токен доступу, який був отриманий при створенні телеграм-бота.

4.2.2 Взаємодія з NC

Після налаштування всіх апаратних конфігурацій слід перейти до написання програми, яка буде збирати дані про мережеве обладнання. Реалізувати таку програму можна на скриптовій мові Python, яка є однією з найпопулярніших при роботі з даними. Бібліотека requests надає змогу працювати з мережевою інформацією за допомогою запитів GET, POST, DELETE та інших.

Зв'язок між Cisco Network Controller (NC) і телеграм-ботом може бути здійснений за допомогою застосунку, який використовує API Telegram та API Network Controller (NC).

Програмний код реалізує наступну послідовність дій: запит для отримання сервісного квитка для доступу до мережевого контролера за його IP-адресою. Після цього формується запит на отримання інформації про мережеві прилади у вигляді JSON-файлу.

JSON – це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дає змогу описувати об'єкти та інші структури даних. Цей формат використовується переважно для передачі структурованої інформації через мережу завдяки процесу, що називають серіалізацією. Структура JSON-документу подібна до словника (*dictionary*) у мові Python, бо доступ до певного елемента можна отримати через відповідний ключ, або індекс.

4.2.3 Опис логічної структури програми

Застосунок має наступну логічну структуру роботи (рис. 4.7):

– користувач в додатку Telegram надсилає запити до чат-боту, який в свою чергу надсилає запит GET до NC для отримання відомостей про стан мережі та її компонентів;

– API дає відповідь, яку обробляє чат-бот та повертає її користувачу;

– NC в випадку додавання або видалення мережного пристрою надсилає в чат-бот повідомлення з відомостями про пристрої.

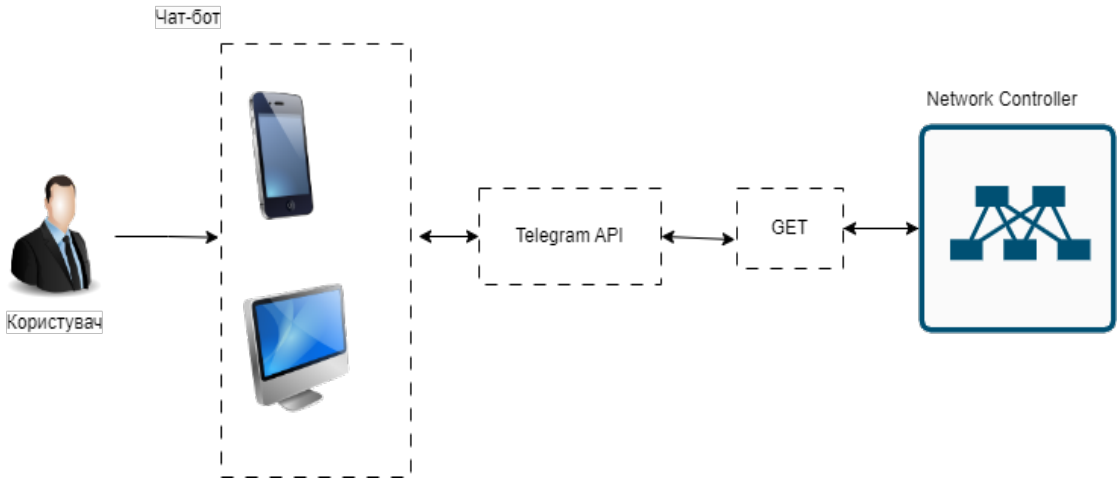

Рисунок 4.7 – Схема роботи застосунку

Програма складається з 24 функцій, та основною функцією def main() (Додаток Б):

– getTicket: функція для отримання токену (ticket) для авторизації в системі;

– Start: функція-обробник команди /start, яка надсилає інформаційне повідомлення;

– menu: функція-обробник команди /menu, яка надсилає користувачеві список доступних команд;

– getIOSDev: функція для отримання інформації про мережні пристрої (hostname, serialNumber, softwareVersion);

– iosversion: функція-обробник команди /iosversion, яка надсилає користувачеві інформацію про мережні пристрої;

– topology: функція-обробник команди /topology, яка відкриває топологію мережних пристроїв у веб-браузері;

– show_nt_info: функція-обробник команди /showntinfo, яка надсилає користувачеві інформацію про пристрої в мережі;

– unknown_command: функція-обробник невідомої команди, яка надсилає користувачеві повідомлення про невідому команду;

– check_new_hosts: функція для перевірки нових кінцевих пристроїв у мережі і надсилання сповіщень про їх додавання або видалення в телеграм;

– check_new_network_devices: функція для перевірки нових мережних пристроїв у мережі і надсилання сповіщень про їх додавання або видалення в телеграм;

– PostToTelegram: функція використовується для надсилання повідомлення в Telegram. Вона приймає параметри, такі як chat_id (ідентифікатор чату, куди потрібно надіслати повідомлення) та message (текст повідомлення). Функція виконує HTTP-запит до Telegram API, використовуючи chat_id та message, щоб надіслати повідомлення до зазначеного чату;

– getNewHosts: функція використовується для отримання списку нових хостів. Вона може використовувати різні методи аналізу журналів або моніторити мережу для виявлення нових хостів, які долучилися до мережі. Результатом функції є список нових хостів;

– getHostInfo: Ця функція використовується для отримання інформації про конкретний хост. Вона може використовувати різні методи, такі як опитування системи або мережеві запити, для отримання детальної інформації про хост. Результатом функції є об'єкт або структура даних, що містить інформацію про хост;

– getDevCount: Ця функція використовується для отримання кількості пристроїв, підключених до хоста. Вона може використовувати різні методи, такі як опитування системи або моніторинг мережі, для підрахунку кількості

підключених пристроїв. Результатом функції є кількість підключених пристроїв;

– getHostCount: Ця функція використовується для отримання загальної кількості хостів у мережі. Вона може використовувати різні методи, такі як опитування системи або моніторинг мережі, для підрахунку кількості хостів. Результатом функції є загальна кількість хостів.

4.2.4 Перевірка роботи застосунку

Після підключення до API Telegram програма встановлює спеціальні обробники (handlers) для реагування на різні типи повідомлень, такі як текстові повідомлення або команди.

Перевірка роботи застосунку буде виконувватися одночасно через відкритий веб-інтерфейс NC.

Після запуску чат-бота надсилається повідомлення про кількість хостів та мережних пристроїв (4.8). На рисунку кількість пристроїв на Dashboard та в повідомленні співпадають.

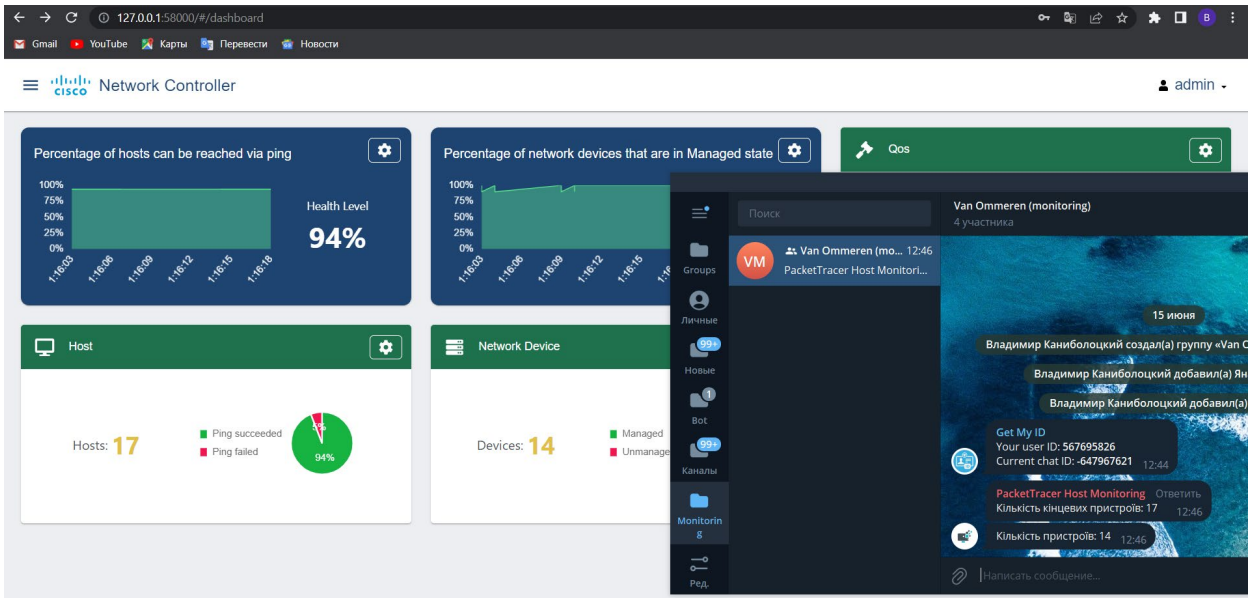

Рисунок 4.8 – Запуск застосунку

Для перевірки отримання повідомлень в випадках додавання або вилучання пристрою з топології в Telegram надсилається повідомлення з відповідним змістом та відомостями про пристрій (рис4.9).

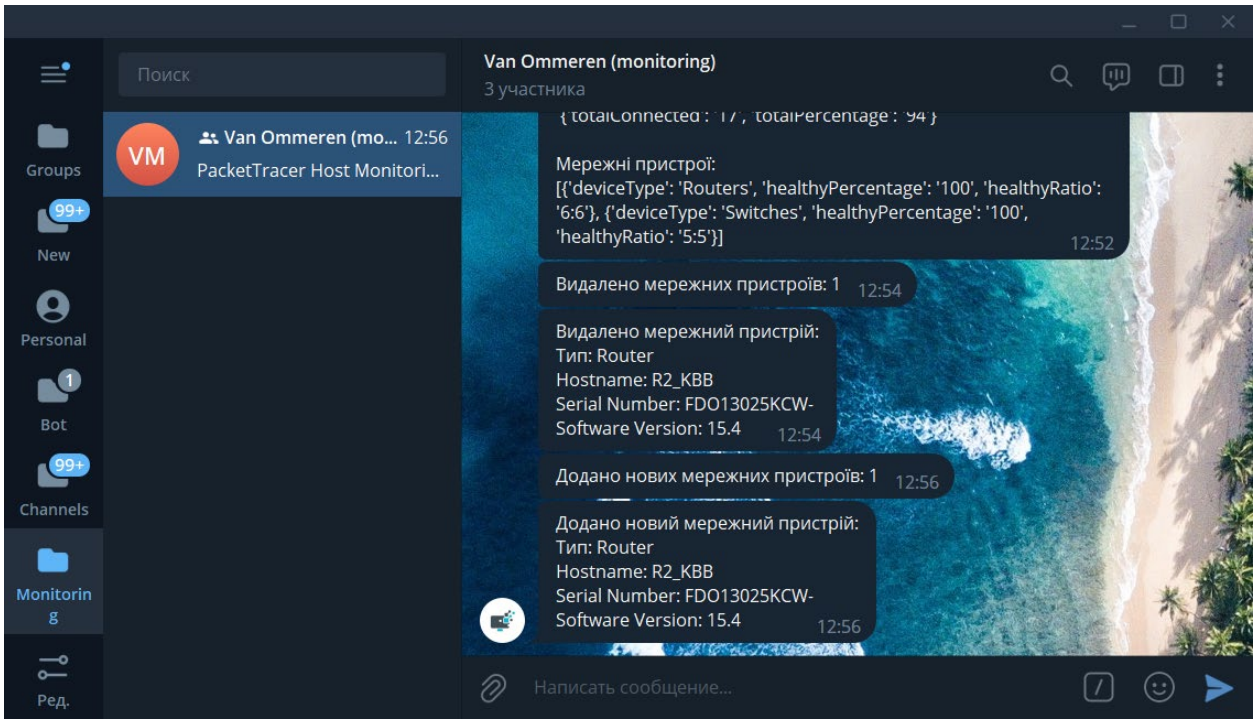

Рисунок 4.9 – Повідомлення про додавання або вилучення пристрою з мережі

Користувач може вести діалог із ботом за допомогою чотирьох команд:

– iosversion - Інформація про версію IOS мережних пристроїв;

– dashboardNC - Відкрити Dashbosrd Network Controlle;

– showndinfo - Показати інформацію про мережні пристрої в мережі;

– sitehealth - Показати загальну інформацію про стан мережі;

– networkhealth - Загальна інформація про стан мережі за категорією пристрою.

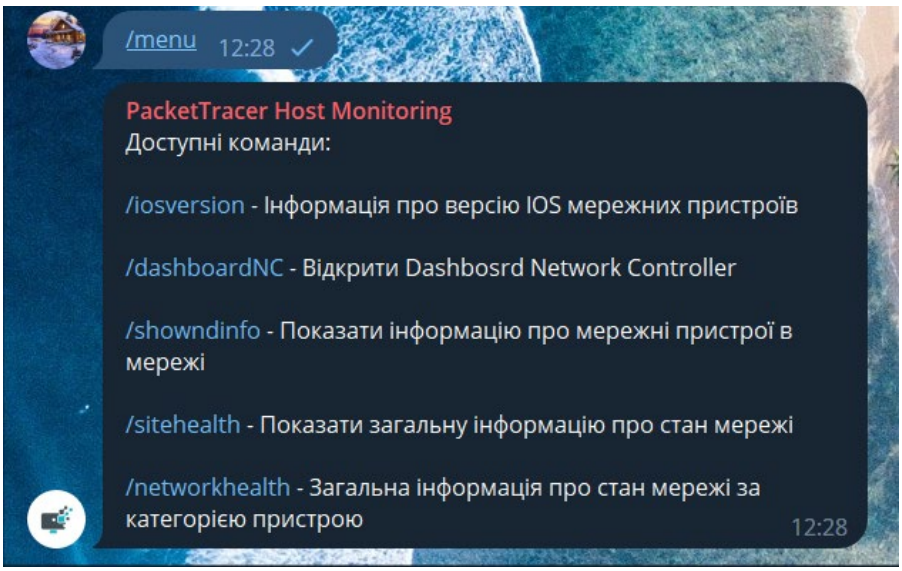

Рисунок 4.10 – Меню чат-бота

Меню /sitehealth надає дані про стан мережі. NC надає наступні дані в фоматі json:


```
{
   "healthyClient": "94",
   "healthyNetworkDevice": "100",
   "numApplicationPolicies": "0",
   "numLicensedRouters": "6",
   "numLicensedSwitches": "5",
   "numNetworkDevices": "14",
   "numUnreachable": "0"
}
```


Ці відомості відображаються на панелях на головній сторінці веб-сайту NC (рис. 411). За запитом ці дані надсилаються в Telegram (рис. 4.12)

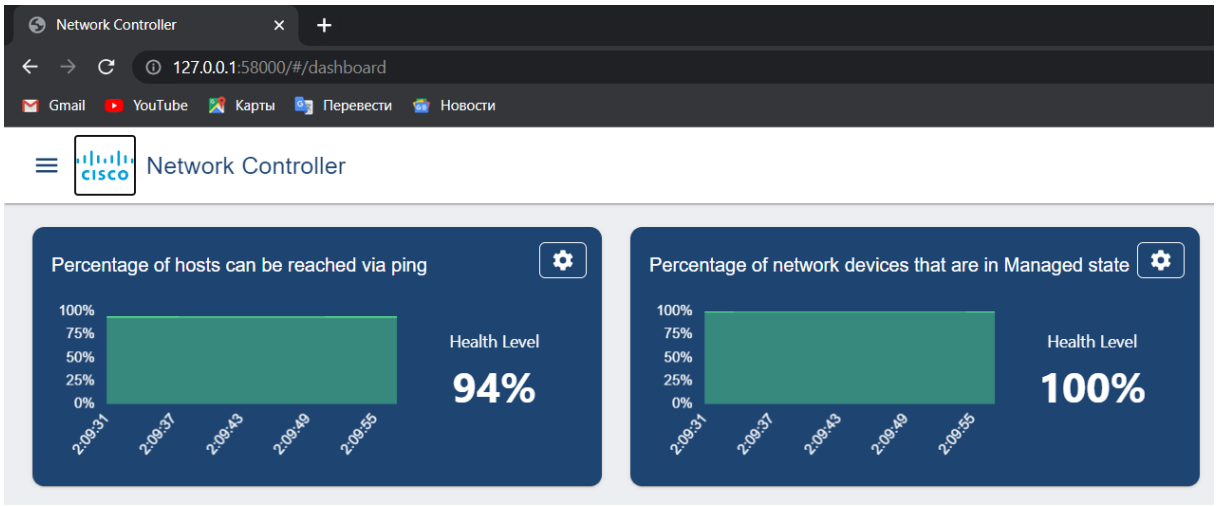

Рисунок 4.11 – Стан хостів та мережних пристроїв на Dashboard

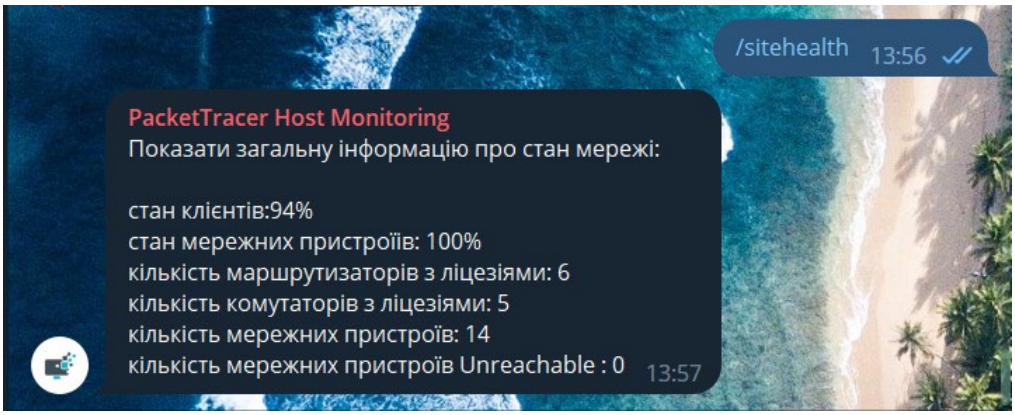

Рисунок 4.12 – Повідомлення в Telegram про ста хостів та мережних пристроїв

– Загальна інформація про стан мережі за категорією пристрою.

Додатково для отримання за необхідності версії IOS на мережних пристроях розроблено меню /iosversion (рис. 4.13)

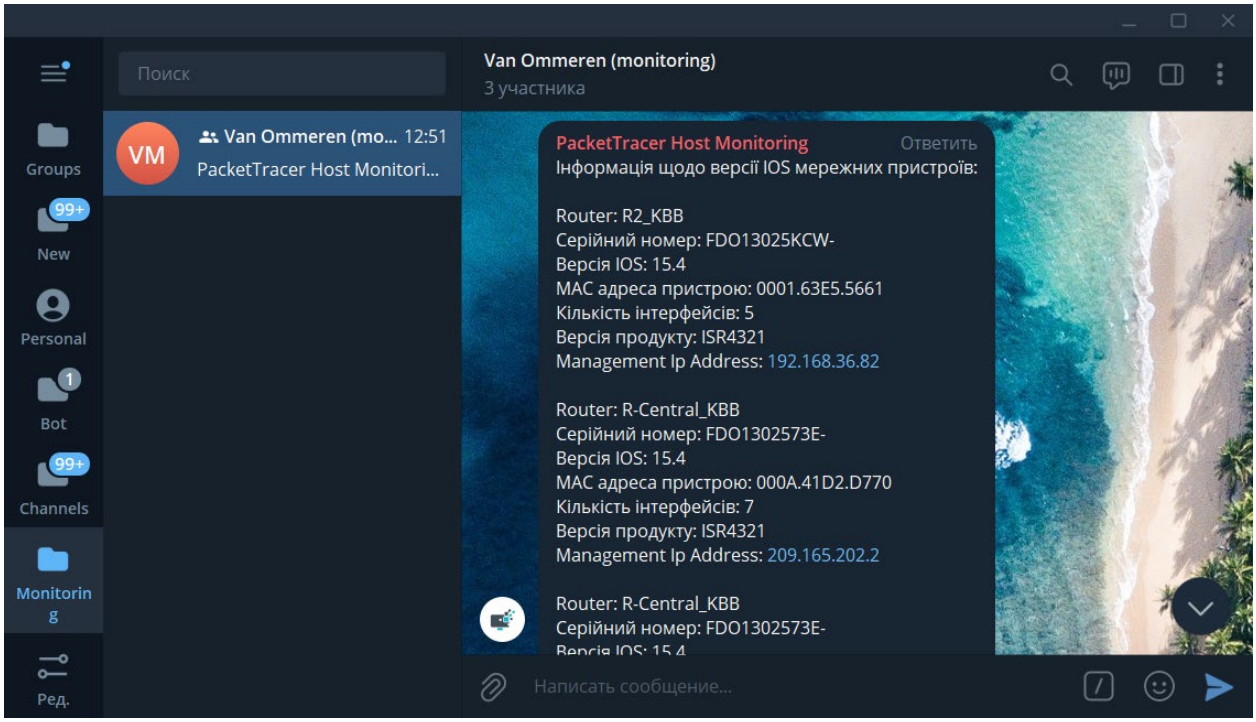

Рисунок 4.13 – Повідомлення про версії IOS мережних пристроїв

ВИСНОВОК

Метою кваліфікаційної роботи було розробка системи моніторингу стану мережних пристроїв для покращення надійності та безпеки комп’ютерної мережі компанії "Van Ommeren" та для підвищення ефективності процесу її адміністрування.

У першому розділі розглянпуто сферу дослідження, визначено , що являє собою моніторинг мережі, його значення у функціонуванні підприємства, проведений аналіз наявних систем моніторингу мережі.

У другому розділі встановлено вимоги до системи.

У третьому розділі розроблена апаратна частина комп’ютерної системи. Проведено моделювання мережі в Packet Tracer з Network Controller, налаштування основних параметрів пристроїв та налаштування мереж VLAN, маршрутизації між VLAN а також налаштування та перевірка DHCP та NAT.

У четвертому розділі розроблена підсистема моніторингу доступності хостів, проведено проектування мережі на основі Network Controller (NC), розроблена на Python програма моніторингу доступності хостів та проведена перевірка роботи застосунку**.** В випадках додавання або вилучання пристрою з топології в Telegram надсилається повідомлення з відповідним змістом та відомостями про пристрій.

В якості рішення задачі моніторингу доступності хостів було запропоновано додавання NC до комп'ютерної мережі. запропоноване рішення включає створення HTTP-запитів до NC API для отримання інформації про хости, кількості пристроїв, стан мережі, відомості про мережні пристрої тощо.

Створена програма мовою Python, яка автоматично та за потреби прямо в месенджері Telegram може показати інформацію про пристрої в мережі та показати загальну інформацію про її стан. У випадку коли потрібно встановити оновлення IOS, можна дізнатись поточну версію та пристрої, які потребують оновленн. Це рішення дозволить забезпечити більш ефективне та автоматизоване керування та моніторинг за мережею, що полегшує життя системним адміністраторам та забезпечує більш високу надійність та безпеку мережевих пристроїв та ресурсів.

Використовуючи модель схеми корпоративної мережі та застосовуючи програми Cisco Packet Tracer і месенджер Telegram, була перевірена функціональність розробленої системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Компаниец В.В. Мировые тренды современного транспортно-логистического сервиса // Вісник економіки транспорту і промисловості No 70-71, 2020 – с. 22-32. Режим доступу http://btie.kart.edu.ua/issue/download/13496/7147

2) ВИМОГИ ДО СИСТЕМ МОНІТОРИНГУ ІТ. Системи моніторингу. Українські підручники та статті – Бібліотека Posibniki.com.ua. Українські підручники, посібники та статті - Бібліотека Posibniki. Електронна бібліотека підручників онлайн. URL: https://posibniki.com.ua/post-vimogi-do-sistem-monitoringu-it (дата звернення: 05.06.2023).

3) Zabbix :: The Enterprise-Class Open Source Network Monitoring Solution. URL: https://www.zabbix.com/ru (date of access: 05.06.2023).

4) ТОП 20 систем моніторингу IT інфраструктури¸ - IT Education Center Blog. URL: https://blog.iteducenter.ua/ratings/monitoring_tools/ (date of access: 05.06.2023).

5) SNMP – Вікі ЦДУ. Вікі ЦДУ. URL: https://wiki.cuspu.edu.ua/index.php/SNMP (дата звернення: 05.06.2023).

6) Nagios - Network, Server and Log Monitoring Software. Nagios. URL: https://www.nagios.com/ (date of access: 05.06.2023).

7) 10 найкращих інструментів моніторингу мережі (2021 рейтинг) - Інший. Огляди, Ігри, Розваги, Червень 2023. URL: https://uk.myservername.com/top-10-best-network-monitoring-tools (дата звернення: 05.06.2023).

8) What is Network Automation? - GeeksforGeeks. GeeksforGeeks. URL: https://www.geeksforgeeks.org/what-is-network-automation/ (date of access: 05.06.2023).

9) Гузій М. М.. Аналіз технологій моніторингу комп’ютерних мереж [Електронний ресурс] / Станіславова О.В. Кадет М.В. //Наукові журнали Національного Авіаційного Університету. Режим доступу до журн.:

https://jrnl.nau.edu.ua/index.php/SBT/article/view/5091/5353 (дата звернення: 05.06.2023).

10) Методичні рекомендації до виконання кваліфікаційної роботи бакалавра студентами галузі знань 12 Інформаційні технології спеціальності 123 Комп'ютерна інженерія / Л.І. Цвіркун, С.М. Ткаченко, Я.В. Панферова, Д.О. Бешта, Л.В. Бешта. – Д.: НТУ «ДП», 2022. – 65 с.

11) ДСТУ 1.5:2015. Правила розроблення, викладання та оформлення національних нормативних документів – К.: Держстандарт, 2015. – 65 с.

12) Cisco 4321 Integrated Services Router. URL: https://www.cisco.com/c/en/us/support/routers/4321-integrated-services-router/model.html#~tab-documents (date of access: 05.06.2023).

13) Мережа на основі контролера | SDN & RESTAPI | Автоматизація [Електронний ресурс] // https://www.youtube.com Режим доступу: https://www.youtube.com/watch?v=a6VNcP-ieI8

14) Cisco APIC REST API Посібник з конфігурації [Електронний ресурс] // https://www.cisco.com Режим доступу: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/2-x/rest_cfg/2_1_x/b_Cisco_APIC_REST_API_Configuration_Guide/b_Cisco_APIC_REST_API_Configuration_Guide_chapter_01000.html

15) Network Controller with API [Електронний ресурс] // https://www.youtube.com Режим доступу: https://www.youtube.com/watch?v=5k5Y1-YO708

16) Що таке API? [Електронний ресурс] // https://seo24.kiev.ua Режим доступу: https://seo24.kiev.ua/razrabotka/chto-takoe-api/

Додаток А

Графічний супровід технічних вимог

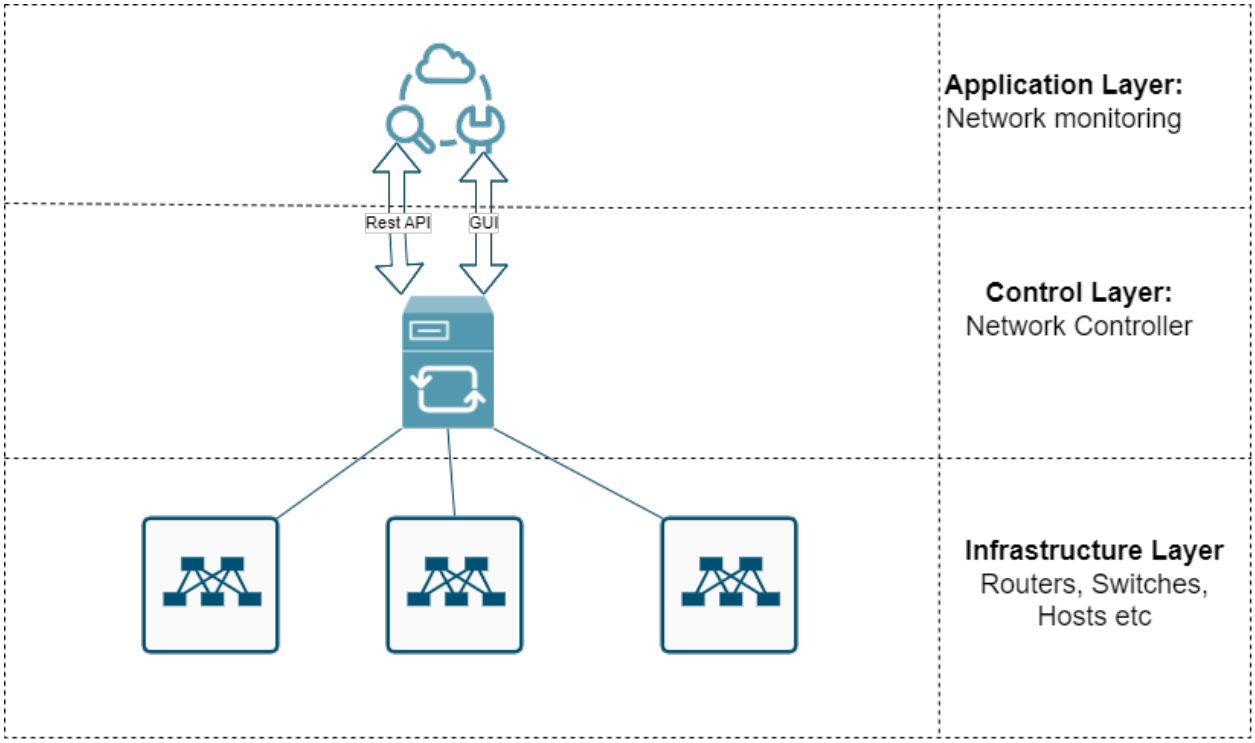

Рисунок ДА.1 – Архітектура системи

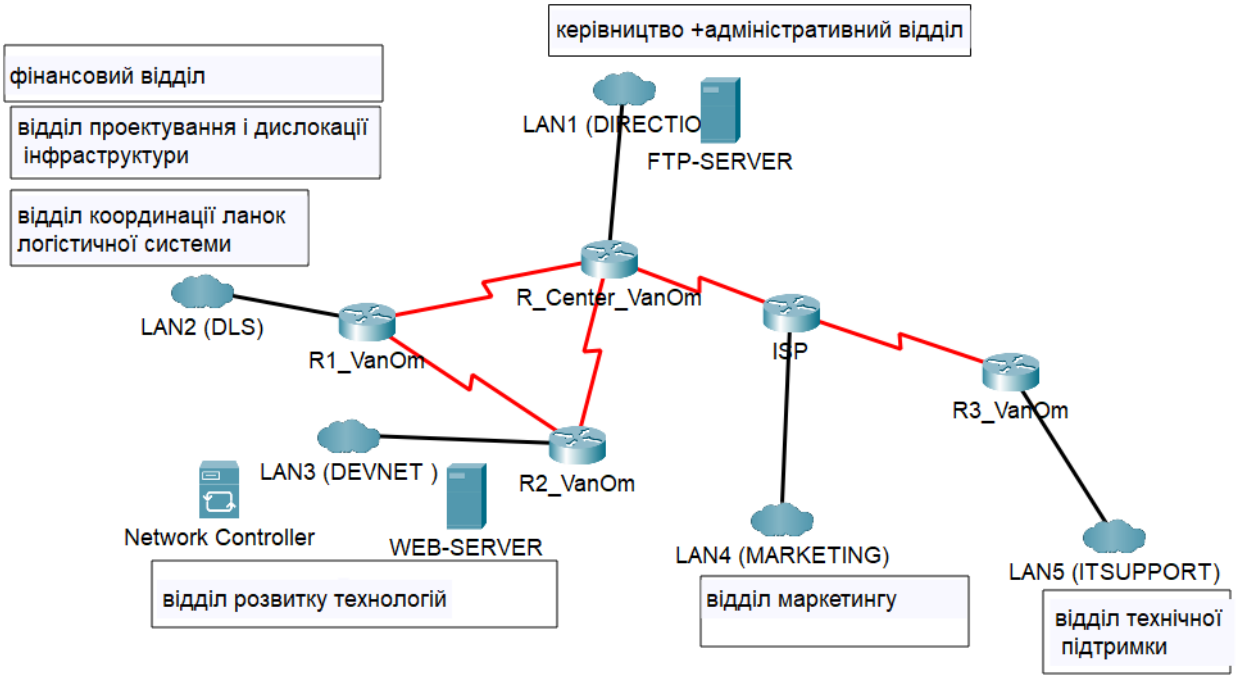

Рисунок ДА.2 – Загальна топологія мережі компанії «Van Ommered»

Додаток Б

Текст програми моніторингу доступності хостів

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
“ДНІПРОВСЬКА ПОЛІТЕХНІКА”

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ НАЛАШТУВАННЯ МЕРЕЖІ КОМП’ЮТЕРНОЇ СИСТЕМИ

Текст програми

804.02070743.23004-01 12 01

Листів 16

2023

АНОТАЦІЯ

Дана програма містить в собі програмний код для зв'язоку між Cisco Network Controller (NC) і телеграм-ботом. Програма написана на мові Python і використовує бібліотеки для роботи API Telegram та API Network Controller (NC).

Ця програма призначена для надсилання повідомлення в Telegram, коли комп'ютер або мережевий пристрій недоступні, і коли вони знов стали доступними, надає інформацію про мережні пристрої, загальну інформацію про стан мережі та інформацію про стан мережі за категорією пристрою.

Програма підключається до API Telegram, використовуючи токен доступу, який був отриманий при створенні телеграм-бота.

ЗМІСТ

```
import requests
import json
from time import sleep
import threading
import webbrowser
from telegram import Update
from telegram.ext import Filters
from telegram.ext import Updater, CommandHandler, MessageHandler, CallbackContext

NC = "http://127.0.0.1:58000/api/v1"
username = "adminnc"
password = "cisco123!"

baseUri = "https://api.telegram.org/"
botToken = "6089592268:AAFM6YDoNJu6gjGNu7nLS5Sp7lQDuge0Lq0"
chatID = "-925197936"
```

1. Ticket

```
def getTicket(baseUri, ad, pas):
        headers = {"Content-Type": "application/json"}
        data = json.dumps({"username": ad, "password": pas})

        try:
                resp = requests.post(baseUri + "/ticket", data=data, headers=headers)
                result = resp.json()

                if "response" in result:
                        ticket = result["response"]["serviceTicket"]
                        return ticket
        except:
                pass
```

2. start

```
def start(update: Update, context: CallbackContext) -> None:
        """Обробник команди /start."""
        user = update.effective_user
        context.bot.send_message(chat_id=update.effective_chat.id, text=f"Привет,
{user.first_name}!")
```

3. menu

```
def menu(update: Update, context: CallbackContext) -> None:
        """Обробник команди /menu."""
        menu_commands = [
                "/iosversion - Інформація про версію IOS мережних пристроїв"
                "\n\n/dashboardNC - Відкрити Dashbosrd Network Controller"
                "\n\n/showndinfo - Показати інформацію про мережні пристрої в мережі"
                "\n\n/sitehealth - Показати загальну інформацію про стан мережі"
```

```
            "\n\n/networkhealth - Загальна інформація про стан мережі за категорією
пристрою"
        ]
        menu_text = "Доступні команди:\n\n" + "\n".join(menu_commands)
        context.bot.send_message(chat_id=update.effective_chat.id, text=menu_text)
```

4. iosversion

```
def iosversion(update: Update, context: CallbackContext) -> None:
        """Обробник команди /iosversion."""
        ticket = getTicket(NC, username, password)
        ios_devices = getIOSDev(NC, ticket)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Інформація щодо
версії IOS мережних пристроїв:" + ios_devices)
```

5. sitehealth

```
def sitehealth(update: Update, context: CallbackContext) -> None:
        """Обробник команди /allassurance."""
        ticket = getTicket(NC, username, password)
        ios_devices = getSiteHealth(NC, ticket)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Показати загальну
інформацію про стан мережі:" + ios_devices)
```

6. networkhealth

```
def networkhealth(update: Update, context: CallbackContext) -> None:
        """Обробник команди /networkhealth."""
        ticket = getTicket(NC, username, password)
        ios_devices = getNetworkHealth(NC, ticket)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Загальна
інформація про стан мережі за категорією пристрою:" + ios_devices)
```

7. dashboardNC

```
def dashboardNC(update, context):
        url = 'http://127.0.0.1:58000/#/topology'
        webbrowser.open(url)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Відкриваю
вебсайт.")

def send_api_request(endpoint, NC, username, password):
        url = NC + endpoint
        response = requests.get(url, auth=(username, password))
        if response.status_code == 200:
                return response.json()
        else:
                print("Помилка при вконанні запиту:", response.status_code)
                return None
```

8. get_devices

```
def get_devices(NC, username, password):
        endpoint = "/devices"
        devices_data = send_api_request(endpoint, NC, username, password)
```

```
        return devices_data
```

9. show_nd_info

```
# Функция для обработки команды /showntinfo
def show_nd_info(update: Update, context: CallbackContext):
        """Обробник команди /showndinfo"""
        ticket = getTicket(NC, username, password)
        ios_devices = showNdInfo(NC, ticket)
        context.bot.send_message(chat_id=update.effective_chat.id, text="Інформація щодо
версії IOS мережних пристроїв:" + ios_devices)

def unknown_command(update: Update, context: CallbackContext) -> None:
        """Обробник невідомої команди."""
        context.bot.send_message(chat_id=update.effective_chat.id, text="Невідома
команда")
```

10. getIOSDev

```
def getIOSDev(baseUri, ticket):
        headers = {"X-Auth-Token": ticket}
        try:
                resp = requests.get(baseUri+"/network-device", headers=headers)

                #       print (resp.status_code)
                result = resp.json()
                print (json.dumps(result, indent=4))

                # Parsing raw response to list network devices
(hostname+serialNumn=ber+version IOS)
                print("\n---List of network devices and IOS version---")
                getIOS=""
                for device in result["response"]:
                        if(device["type"]=="Switch" or device["type"]=="Router"):
                                type=device["type"]
                                hostname = device["hostname"]
                                serial_number = device["serialNumber"]
                                ios_version = device["softwareVersion"]
                                message = "\n\n" + device["type"] + ": " + hostname +
"\nСерійний номер: " + serial_number+ "\nВерсія IOS: " + ios_version
                                print(message)
                                getIOS+=message
                                print(type + ": "+hostname + ", serialNumber: " +serial_number+",
softwareVersion: "+ios_version)
                        return getIOS
        except:
                return 0
```

11. getHostCount

```
def getHostCount(baseUri, ticket):
        headers = {"X-Auth-Token": ticket}
```

```
        try:
                resp = requests.get(baseUri + "/host", headers=headers)
                result = resp.json()
                count = 0
                if "response" in result:
                        for i in result["response"]:
                                if i["pingStatus"] == "SUCCESS":
                                        count += 1
                return count
        except:
                pass
        return 0
```

12. getHosts

```
def getHosts(baseUri, ticket):
        headers = {"X-Auth-Token": ticket}

        try:
                resp = requests.get(baseUri + "/host", headers=headers)
                result = resp.json()
                if "response" in result:
                        host_list=result["response"]
                return host_list
        except:
                pass

        return 0
```

13. getNetDevCount

```
def getNetDevCount(baseUri, ticket):
        headers = {"X-Auth-Token": ticket}
        try:
                resp = requests.get(baseUri + "/network-device", headers=headers)
                result = resp.json()
                count_man = 0
                count_un = 0
                count = 0
                if "response" in result:
                        for i in result["response"]:
                                count +=1
                                if i["collectionStatus"] == "Managed":
                                        count_man += 1
                                else:
                                        if i["collectionStatus"] == "Unreachable":
                                                count_un +=1
                        print("count dev = ", count, ": Manager =", count_man, "Unreachable = ",
count_un )
                return count
        except:
```

```
            pass

    return 0
```

14. getDiscovery

```
def getDiscovery(baseUri, ticket):
    headers = {"X-Auth-Token": ticket}

    try:
        resp = requests.get(f"{baseUri}/discovery/", headers=headers)
        result = resp.json()

        if "response" in result:
            discovery = result["response"]
            print("discovery = ", discovery)
            return discovery
    except:
        pass

    return None
```

15. getNetDev

```
def getNetDev(baseUri, ticket):
    headers = {"X-Auth-Token": ticket}

    try:
        resp = requests.get(f"{baseUri}/network-device/", headers=headers)
        result = resp.json()


        if "response" in result:
            device_list = result["response"]
            return device_list
    except:
        pass

    return None
```

16 getHostInfo

```
def getHostInfo(result, id):
    #headers = {"X-Auth-Token": ticket}

    try:
    #       resp = requests.get(f"{baseUri}/host/{device['id']}", headers=headers)
        for i in result:
            if str(i["id"]) == id:
                return {
                    "hostname": i["hostName"],
                    "MAC": i["hostMac"],
```

```
                                        "IP": i["hostIp"],
                                        "hosttype": i["hostType"]

                                }
        except:
                pass

#       return None
```

17. postToTelegram

```
def postToTelegram(text):

        sendMethod = "/sendMessage"

        url = baseUri + "bot" + botToken + sendMethod
        payload = {
                "chat_id": chatID,
                "text": text
        }

        response = requests.post(url, json=payload)

        if response.status_code == 200:
                print("\nПовідомлення відправлене до телеграм!")
        else:
                print("\nПомилка при відправленні повідомлення до телеграм")
```

18. check_new_hosts

```
def check_new_hosts(baseUri, ticket):
        host_list_before = []
        host_list_before = getHosts(baseUri, ticket)
        host_count_before = getHostCount(baseUri, ticket)

        while True:
                before = set()
                after = set()

                for i in host_list_before:
                        #print(host_list_before)
                        if "id" in i:
                                #print(i["id"])
                                before.add(i["id"])

        #       print("before host=", before)

                host_list_after = getHosts(baseUri, ticket)
                host_count_after = getHostCount(baseUri, ticket)

                if host_count_before != host_count_after:
```

```
                new_host_count = host_count_after - host_count_before

                for i in host_list_after:
                        if "id" in i:
                                after.add(i["id"])
                print("after host=", after)

                if new_host_count > 0:
                        text = f"Додано нових кінцевих пристроїв: {new_host_count}"
                        new_host_list = after - before
                        postToTelegram(text)

                        print("new host list", new_host_list)

                        if new_host_list:
                                for host in new_host_list:
                                        host_info = getHostInfo(host_list_after, host)
                                        print("new host = ", host_info)
                                        if host_info:
                                                print(host_info['hostname'])
                                                message = f"Додано новий кінцевий
пристрій:\nІм'я хоста: {host_info['hostname']}\nIP-адреса: {host_info['IP']}\nMAC:
{host_info['MAC']}\nТип: {host_info['hosttype']}"
                                                postToTelegram(message)
                                        else:
                                                        print("Помилка при читанні
інформації про новий пристрій")
                else:
                        text = f"Видалено кінцевих пристроїв: {abs(new_host_count)}"
                        print("Видалено кінцевих пристроїв", new_host_count)
                        postToTelegram(text)

                        removed_host_list = before - after

                        for host in removed_host_list:
                                host_info = getHostInfo(host_list_before, host)
                                if host_info:
                                        message = f"Видалено кінцевий пристрій:\nІм'я
хоста: {host_info['hostname']}\nIP-адреса: {host_info['IP']}\nMAC: {host_info['MAC']}\nТип:
{host_info['hosttype']}"
                                        postToTelegram(message)
                                else:
                                        print("Помилка при читанні інформації про
видалений host")
        else:
                print("Немає нових або видалених КІНЦЕВИХ пристроїв")

        host_list_before = host_list_after
        host_count_before = host_count_after
```

```
            sleep(3)
```

19. getDeviceInfo

```
def getDeviceInfo(result, id):
    #headers = {"X-Auth-Token": ticket}
    try:
    #       resp = requests.get(f"{baseUri}/network-device", headers=headers)
            for i in result:
                if str(i["id"]) == id:
                    return {
                        "type": i["type"],
                        "hostname": i["hostname"],
                        "serialNumber": i["serialNumber"],
                        "softwareVersion": i["softwareVersion"]
                    }
    except:
            pass

    return None
```

20. check_new_network_devices

```
def check_new_network_devices(baseUri, ticket):
    device_list_before = []  # Список пристроїв до перевірки
    device_list_before  = getNetDev(baseUri, ticket)
    device_count_before = getNetDevCount(baseUri, ticket)

    while True:     if "id" in i:

            before = set()
            after = set()

            for i in device_list_before:

            before.add(i["id"])

            print("before device=", before)

            device_list_after = getNetDev(baseUri, ticket)
            device_count_after = getNetDevCount(baseUri, ticket)

            if device_count_before != device_count_after:
                new_device_count = device_count_after - device_count_before

                for i in device_list_after:
                    if "id" in i:
                        after.add(i["id"])
                print("after devices=", after)
```

```
                    if new_device_count > 0:
                        text = f"Додано нових мережних пристроїв: {new_device_count}"
                        print("Додано нових мережних пристроїв:", new_device_count)
                        postToTelegram(text)
                    #   new_device_list = [i for i in device_list_after if i not in device_list_before]
                        new_device_list = after - before
                        print("new_device_list", new_device_list)

                        if new_device_list:
                            for device in new_device_list:
                                device_info = getDeviceInfo(device_list_after, device)
                                print(device_info)
                                if device_info:
                                    message = f"Додано новий мережний пристрій:\nТип: {device_info['type']}\nHostname: {device_info['hostname']}\nSerial Number: {device_info['serialNumber']}\nSoftware Version: {device_info['softwareVersion']}"
                                    postToTelegram(message)
                                else:
                                    print("Помилка при читанні інформації про новий devices")
                    else:
                        text = f"Видалено мережних пристроїв: {abs(new_device_count)}"
                        print("Видалено мережних пристроїв", new_device_count)
                        postToTelegram(text)

                        removed_device_list = before - after
                        for device in removed_device_list:
                            device_info = getDeviceInfo(device_list_before, device)
                            if device_info:
                                message = f"Видалено мережний пристрій:\nТип: {device_info['type']}\nHostname: {device_info['hostname']}\nSerial Number: {device_info['serialNumber']}\nSoftware Version: {device_info['softwareVersion']}"
                                postToTelegram(message)
                            else:
                                print("Помилка при читанні інформації про видалений пристрій")

                else:
                    print("Немає нових або видалених МЕРЕЖНИХ пристроїв")

                device_list_before = device_list_after
                device_count_before = device_count_after

                sleep(3)  # Очікування 10 секунд перед наступною перевірко
```

```
def getSiteHealth(baseUri, ticket):

        headers = {"X-Auth-Token": ticket}
        try:
                resp = requests.get(baseUri+"/assurance/health-issues", headers=headers)
                print (resp.status_code)
                result = resp.json()
                print(json.dumps(result, indent=4))

                # Parsing raw response to list network devices
(hostname+serialNumn=ber+version IOS)
                print("\n---Get all network health issues---")
                print(json.dumps(result, indent=4))
                getIOS=""

                for device in result["response"]:
                        if(device["type"]=="Switch" or device["type"]=="Router"):
                                type=device["type"]
                                hostname = device["hostname"]
                                serial_number = device["serialNumber"]
                                ios_version = device["softwareVersion"]
                                message = "\n\nІм'я кінцевого пристрою: " + hostname +
"\nСерійний номер: " + serial_number+ "\nВерсія IOS: " + ios_version
                                print(message)
                                getIOS+=message
                                print(type + ": "+hostname + ", serialNumber: " +serial_number+",
softwareVersion: "+ios_version)

                return getIOS

        except:
                return 0
```

21. getNetworkHealth

```
def getNetworkHealth(baseUri, ticket):
        headers = {"X-Auth-Token": ticket}

        try:
                resp = requests.get(f"{baseUri}/assurance/health", headers=headers)
                print("\n---Returns Overall Network Health information by Device category
(Router, Switch, Host) for any given point of time---")
                result = resp.json()
                print(json.dumps(result, indent=4))
                message = ""

                for device in result["response"]:
                        clients=device["clients"]

                        networkDevices = device["networkDevices"]["networkDevices"]
```

```
                print(networkDevices)
                message = "\n\nКлієти:\n\t"+ str(clients) + "\n\nМережні пристрої:\n" +
str(networkDevices)
                print(message)
                break
        #message = "\n\nІм'я кінцевого пристрою: " + hostname + "\nСерійний номер:
" + serial_number+ "\nВерсія IOS: " + ios_version
        #print(message)
        #getIOS+=message
        #print(type + ": "+hostname + ", serialNumber: " +serial_number+",
softwareVersion: "+ios_version)

        return message

    except:
        return 0
```

22. getSiteHealth

```
def getSiteHealth(baseUri, ticket):
    headers = {"X-Auth-Token": ticket}

    try:
        resp = requests.get(f"{baseUri}/network-health", headers=headers)
        print("\n---Returns Overall Health information for the network---")
        result = resp.json()
        print(json.dumps(result, indent=4))
        message = ""

        message = "\n\nстан клієнтів:"+ result["healthyClient"] + "%" + "\nстан
мережних пристроїів: " + result["healthyNetworkDevice"] + "%"
        message += "\nкількість маршрутизаторів з ліцезіями: " +
result["numLicensedRouters"]
        message += "\nкількість комутаторів з ліцезіями: " +
result["numLicensedSwitches"]
        message += "\nкількість мережних пристроїв: " + result["numNetworkDevices"]
        message += "\nкількість мережних пристроїв Unreachable : " +
result["numUnreachable"]

        print(message)

        #message = "\n\nІм'я кінцевого пристрою: " + hostname + "\nСерійний номер:
" + serial_number+ "\nВерсія IOS: " + ios_version
        #print(message)
        #getIOS+=message
        #print(type + ": "+hostname + ", serialNumber: " +serial_number+",
softwareVersion: "+ios_version)

        return message
```

```
        except:
                return 0
```

23. showNdInfo

```
def showNdInfo(baseUri, ticket):
                headers = {"X-Auth-Token": ticket}
                try:
                        resp = requests.get(baseUri+"/network-device", headers=headers)

                        #       print (resp.status_code)
                        result = resp.json()
                        print (json.dumps(result, indent=4))

                        # Parsing raw response to list network devices
(hostname+serialNumn=ber+version IOS)
                        message=""
                        for device in result["response"]:
                                if(device["type"]=="Switch" or device["type"]=="Router"):
                                        type=device["type"]
                                        hostname = device["hostname"]
                                        serial_number = device["serialNumber"]
                                        ios_version = device["softwareVersion"]
                                        message += "\n\n" + device["type"] + ": " + hostname +
"\nСерійний номер: " + serial_number+ "\nВерсія IOS: " + ios_version
                                        message+="\nMAC адреса пристрою: " +
device["macAddress"]
                                        message+="\nКількість інтерфейсів: " +
device["interfaceCount"]
                                        message+="\nВерсія продукту: " + device["productId"]
                                        message+="\nManagement Ip Address: " +
device["managementIpAddress"]

                        return message
                except:
                        return 0
```

24. main

```
def main():

        NC = "http://127.0.0.1:58000/api/v1"
        username = "adminnc"
        password = "cisco123!"

        updater = Updater(botToken, use_context=True)
        dispatcher = updater.dispatcher

        # Добавляем обработчики команд
        dispatcher.add_handler(CommandHandler("start", start))
        dispatcher.add_handler(CommandHandler("menu", menu))
        dispatcher.add_handler(CommandHandler("iosversion", iosversion))
```

```
        dispatcher.add_handler(CommandHandler('dashboardNC', dashboardNC))
        dispatcher.add_handler(CommandHandler('showndinfo', show_nd_info))
        dispatcher.add_handler(CommandHandler('sitehealth', sitehealth))
        dispatcher.add_handler(CommandHandler('networkhealth', networkhealth))
        dispatcher.add_handler(MessageHandler(Filters.text, unknown_command))


        # start your shiny new bot
        updater.start_polling()

        ticket = getTicket(NC, username, password)

        numberHost = getHostCount(NC, ticket)
        print("Кількість кінцевих пристроїв: ", numberHost)
        postToTelegram("Кількість кінцевих пристроїв: " + str(numberHost))

        numberDev = getNetDevCount(NC, ticket)
        print("Кількість пристроїв: ", numberDev)
        postToTelegram("Кількість пристроїв: " + str(numberDev))

        result= getNetDev(NC,ticket)

        # Створення окремих потоків для функцій check_new_network_devices та
check_new_hosts
        network_devices_thread = threading.Thread(target=check_new_network_devices,
args=(NC, ticket))
        hosts_thread = threading.Thread(target=check_new_hosts, args=(NC, ticket))

        # Запуск потоків
        network_devices_thread.start()
        hosts_thread.start()

        # Головний потік чекає на завершення потоків
        network_devices_thread.join()
        hosts_thread.join()

        sleep(10)

if __name__ == "__main__":
        main()
```