НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” _____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Система моніторингу та управління якістю обслуговування в комп’ютерній мережі.

Виконав (-ла): студент (-ка) IV курсу, групи ___КВ-53___
(шифр групи)

_____Болілий Дмитро Андрійович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _____доц., к.т.н., доц. Щербина О.А._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М. _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент професор каф. ОТ, д.т.н., проф. Кулаков Ю.О. _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

ЗАВДАННЯ
на дипломний проект студента
Болілого Дмитра Андрійовича

1. Тема проекту: Система моніторингу та управління якістю обслуговування в комп’ютерній мережі.
Керівник проекту Щербина Олександр Андрійович, доц. каф. СПіСКС, к.т.н., затверджені наказом по університету від «06» квітня 2018 р. №1108-С

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту: див. технічне завдання.

4. Зміст пояснювальної записки: аналіз існуючих рішень та обґрунтування теми диплому, обґрунтування вибору методів реалізації, розробка компонентів системи, аналіз роботи програмного продукту.

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо): загальна структура роботи системи, система моніторингу, розбір пакету на складові, синхронізація інтерфейсу та перехоплювача.

6. Консультанти розділів проекту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я.М., доц. каф. СПіСКС, к.т.н.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання
1.	Видача завдання на дипломне проектування	04.11.2018
2.	Вивчення літератури за тематикою роботи	16.11.2018
3.	Розроблення та узгодження технічного завдання	25.11.2018
4.	Розроблення структури додатку	17.01.2019
5.	Розроблення дизайну та графічних елементів	04.02.2019
6.	Програмна реалізація додатку	15.03.2019
7.	Тестування додатку	04.04.2019
8.	Підготовка матеріалів текстової частини проекту	24.04.2019
9.	Підготовка матеріалів графічної частини проекту	17.05.2019
10.	Оформлення технічної документації проекту	28.05.2019

Студент _____ Болілий Д.А.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Щербина О.А.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

АНОТАЦІЯ

Кваліфікаційна робота містить пояснювальну записку(50 сторінок, 19 рисунків, 1 таблиця)

Даний бакалаврський проект присвячений розробці системи моніторингу якості передачі даних комп’ютерною мережею. В роботі проаналізована доцільність використання різних протоколів моніторингу комп’ютерної мережі.

Основна увага присвячена розробці додатку (прикладного сервісу) з інтуітивно-зрозумілим інтерфейсом, який дозволяє забезпечити моніторинг мережі як в активному, так і в пасивному режимі.

Розроблена система моніторингу дозволяє:

- в реальному часі отримувати пакети, що проходять через комп'ютер;
- розбирати пакети на складові для зручного надання даних користувачеві
- збирати і відображати інформацію для моніторингу, таку як пропускна спроможність, кількість пакетів за одиницю часу, час життя пакетів відображений у вигляді графіку, кількість втрачених пакетів.

Ключові слова:

Моніторинг, обслуговування, Python3, PyQt5, Linux, qt5-designer, pyuic5

ABSTRACT

The qualifying work contains an explanatory note (50 pages, 19 images, 1 table)

This bachelor project is devoted to the development of a system for monitoring the quality of data transmission by computer network. The paper analyzes the feasibility of using different computer network monitoring protocols.

The main attention is devoted to the development of the application (application service) with an intuitive interface, which allows to monitor the network in both active and passive mode.

The developed monitoring system allows:

- In real time receive packets that pass through a computer;
- to disassemble packages for components for convenient data provision to the user;
- collect and display monitoring information such as bandwidth, packets per unit time, time to live displayed in graphics, number of lost packets.

Keywords:

Monitoring, Maintenance, Python3, PyQt5, Linux, qt5-designer, pyuic5

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4	ІАЛЦ.467100.002 ТЗ	Система моніторингу	4		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Технічне завдання			
2	А4	ІАЛЦ.467100.003 ТП	Система моніторингу	2		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Відомість технічного			
			проекту			
3	А4	ІАЛЦ.467100.004 ПЗ	Система моніторингу	49		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Пояснювальна записка			
4	А4	ІАЛЦ.467100.005 Е1	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Загальна структура роботи			
			системи. Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.001 ОА		
Розробив		Болілий Д.А.			Система моніторингу та управління якістю обслуговування в комп’ютерній мережі Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Щербина О.А.					
Н. контроль		Клятченко Я.М.				КПІ імені Ігоря Сікорського ФПМ КВ-53	
Затвердив		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
5	А4	ІАЛЦ.467100.006 Е2	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Система моніторингу.			
			Схема структурна			
6	А4	ІАЛЦ.467100.007 Д1	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Розбір пакету на складові.			
			Схема алгоритму			
7	А4	ІАЛЦ.467100.008 Д2	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Синхронізація інтерфейсу			
			та перехоплювача.			
			Схема алгоритму			
8		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічні матеріали			

ЗМІСТ

стор.

1. НАЙМЕНУВАННЯ І ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування роботи – «Система моніторингу та управління якістю обслуговування в комп’ютерній мережі».

Галузь застосування: інформаційні технології, аналіз трафіку комп’ютерних мереж.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання дипломного проекту першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. ЦІЛЬ І ПРИЗНАЧЕННЯ РОБОТИ

Розробка призначена для використання як тестування якості обслуговування мережі з метою виявлення та подальшого вирішення проблем використання будь-якої окремо взятої мережі.

4. ДЖЕРЕЛА РОБОТИ

Джерелами роботи є конспект лекцій з курсу «Комп’ютерні мережі», науково-технічна література по копм’ютерним мережам та моніторингу їх якості, статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до додатку, що розробляється:

- можливість перехоплювання пакетів, що проходять через мережу;
- можливість розбору пакетів на їх складові;
- відображення інформації основних метрик моніторингу мережі;
- можливість вибору протоколу, що цікавить користувача для отримання більш докладної інформації про нього;
- робота інтерфейсу та перехоплювача на різних потоках та їх синхронізація.

5.2 Вимоги до апаратного забезпечення:

- комп’ютер;
- маршрутизатор з доступом до мережі інтернет.

5.3 Вимоги до програмного забезпечення:

- операційна система Linux;
- Python 3.7.3;
- PyQt5;
- Qt5 Designer.

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання
1.	Видача завдання на дипломне проектування	04.11.2018
2.	Вивчення літератури за тематикою роботи	16.11.2018
3.	Розроблення та узгодження технічного завдання	25.11.2018
4.	Розроблення структури додатку	17.01.2019
5.	Розроблення дизайну та графічних елементів	04.02.2019
6.	Програмна реалізація додатку	15.03.2019
7.	Тестування додатку	04.04.2019
8.	Підготовка матеріалів текстової частини проекту	24.04.2019
9.	Підготовка матеріалів графічної частини проекту	17.05.2019
10.	Оформлення технічної документації проекту	28.05.2019

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
1	А4	ІАЛЦ.467100.004 ПЗ	Система моніторингу	49		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Пояснювальна записка			
2	А4	ІАЛЦ.467100.005 Е1	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Загальна структура роботи			
			системи.			
			Схема структурна			
3	А4	ІАЛЦ.467100.006 Е2	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Система моніторингу.			
			Схема структурна			

Змін.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467100.003 ТП		
Розробив		Болілий Д.А.			Система моніторингу та управління якістю обслуговування в комп’ютерній мережі Відомість технічного проекту	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Щербина О.А.					
Н. контроль		Клятченко Я.М.				КПІ імені Ігоря Сікорського ФПМ КВ-53	
Затвердив		Тарасенко В.П.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
4	А4	ІАЛЦ.467100.007 Д1	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Розбір пакету на складові.			
			Схема алгоритму			
5	А4	ІАЛЦ.467100.008 Д2	Система моніторингу	1		
			та управління якістю			
			обслуговування в			
			комп’ютерній мережі.			
			Синхронізація інтерфейсу			
			та перехоплювача.			
			Схема алгоритму			
6		Диск CD-ROM	Текст пояснювальної	1		
			записки.			
			Графічні матеріали			

					ІАЛЦ.467100.003 ТП	Арк.
Змін.	Арк.	№ докум.	Підпис	Дата		2

ЗМІСТ

стор

ДОДАТКИ

Додаток 1. Копії графічного матеріалу.

- ІАЛЦ.467100.005 Е1. Загальна структура роботи системи. Схема структурна;
- ІАЛЦ.467100.006 Е2. Система моніторингу. Схема структурна;
- ІАЛЦ.467100.007 Д1. Розбір пакету на складові. Схема алгоритму;
- ІАЛЦ.467100.008 Д2. Синхронізація інтерфейсу та перехоплювача. Схема алгоритму;

Додаток 2. Лістинг програми.

Додаток 3. Презентація.

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ДП	Диференційовані послуги
ІП	Інтегровані послуги
КЯО	Контроль якості обслуговування
СММ	Система моніторингу мережі
ЦП	Центральний процесор
DSCP	Differentiated Services Code Point. Кодова точка диференційованипх послуг
IP	Internet Protocol. Міжмережевий протокол
LAN	Local Area Network. Локальна мережа
MIB	Management Information Base. База управляючої інформації
MOS	Mean opinion score. Середній бал
MPLS	Multiprotocol Label Switching. Мультипротокова мережа комутації міток
NAT	Network Address Translation
NMS	Network Monitoring Systems. Система моніторингу мережі
PHB	Per-Hop Behaviour. Політика покрокового обслуговування
PPS	Packets Per Second
QoS	Quality of service. Якість обслуговування
RSVP	Resource Reservation Protocol. Протокол резервування ресурсів
SD-WAN	Software-Defined networking in a Wide Area Network
SLA	Service Level Agreement. Угода про рівень обслуговування
SNMP	Simple Network Management Protocol
TTL	Time To Live. Час життя пакету
URI	Uniform Resource Identifier
WMI	Windows Management Instrumentation

ВСТУП

Моніторинг та управління якістю передачі в комп’ютерній мережі завжди були не простим викликом навіть для професіоналів цієї справи. Для правильної роботи таких систем потрібні платформи з визначеним набором інструментів для коректної роботи. На сьогоднішній день з’являється все більше ресурсів, що наближаються до універсальних і дозволяють їх використовувати в самих різних мережевих системах.

Такі платформи мають автоматизовані алгоритми для виявлення більшості мережевих компонентів та автоматично відображають інформацію їх корисного використання у текстовому, графічному та інших видах, для зручності користування.

Розуміючи всі аспекти проблеми якості обслуговування мережі необхідно розробити рішення для моніторингу пакетів даних, які передаються через робочу станцію, що дозволить забезпечити збір необхідної інформації для надання її користувачу.

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ, ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМУ

1.1 Моніторинг мережі

Моніторинг мережі включає в себе декілька методів, які використовуються для підтримки безпеки та цілісності внутрішньої мережі. Внутрішня мережа також називається локальною мережею LAN (Local Area Network), а моніторинг охоплює контроль апаратного забезпечення, програмного забезпечення, наявність вірусів, шпигунських програм, вразливостей, такі як бекдор і отвори безпеки, а також інші аспекти, які можуть порушити цілісність мережі.

1.1.1 Методи моніторингу мережі

Існує широкий спектр методів моніторингу мережі, які реалізуються IT-фахівцями. Методи розгортаються за допомогою мережевих моніторингових рішень, які автоматично виявляють і реагують на проблеми безпеки і продуктивності. Розглянемо найбільш розповсюджені з них, до яких відносяться: виявлення вторгнень, перехоплення пакетів, сканування вразливостей, брандмауер моніторингу та тестування проникнення.

• Виявлення вторгнень: виявлення вторгнень контролює локальні мережі для несанкціонованого доступу хакерів. Цей метод може бути реалізований вручну, проте більшість IT-фахівців вважають за краще використовувати програму виявлення вторгнень, яка автоматично виявляє віруси та шкідливе програмне забезпечення, вразливості мережі, такі як бекдори, логічні бомби та інші загрози безпеці, окремі комп'ютери, які підключені до мережі та налаштування файлів. Програми виявлення вторгнень генерують звіти після перевірки системи, тому будь-які проблеми можна вирішити.

• Перехоплення пакетів: сніфер пакетів - програма, яка перевіряє кожен пакет даних, який проходить через мережу. Метою перегляду пакетів є виявлення неавторизованого програмного забезпечення для моніторингу мережі, яке може бути встановлено хакерами для шпигунства за діловою активністю та інформаційним процесом.

• Сканування вразливостей: сканер вразливостей періодично сканує комп'ютерну мережу та перевіряє на наявність вразливостей і слабких сторін, які відкривають потенціал для експлуатації. Цей метод відрізняється від виявлення вторгнення, оскільки він виявляє слабкість до того, як атака відбулася. Виявлення вторгнень виявляє несанкціонований доступ після того, як хакер порушив роботу мережі.

• Брандмауер моніторингу: брандмауери контролюють трафік, що надходить в мережу і виходить з неї. Моніторинг брандмауера відстежує діяльність брандмауера, для забезпечення належного і безпечного процесу перевірки вхідних і вихідних з'єднань.

• Тестування проникнення: тестування проникнення здійснюється IT-спеціалістами за допомогою методів, які хакери використовують для порушення цілісності мережі. Метою цього процесу є прийняття мережевої безпеки на інший рівень шляхом виявлення вразливостей, про які можуть знати хакери, але які ще не були виявлені за допомогою інших методів моніторингу.

1.1.2 Модель OSI

Модель OSI стандартизує ключові функції мережі, використовуючи мережеві протоколи. Це дозволяє різним типам пристроїв від різних постачальників спілкуватися один з одним через мережу.

У моделі OSI мережеві комунікації згруповані в сім логічних рівнів. Два пристрої використовують стандартизовані протоколи OSI на кожному рівні. В таблиці 1.1 наведені ці рівні та визначено функції, які вони реалізують [1].

Таблиця 1.1 – Рівні моделі OSI

Рівень	Функція
Рівень 7: Прикладні сервіси.	Взаємодіє з програмними додатками, які реалізують комунікаційний компонент.
Рівень 6: Представлення даних.	Перетворює вхідні та вихідні дані з одного формату презентації в інший (шифрування даних, стиснення тексту).
Рівень 5: Сесійний.	Контролює з'єднання між комп'ютерами. Встановлює і припиняє з'єднання, підтримує активним канал протягом сесії.
Рівень 4: Транспортний.	Забезпечує передачу даних від джерела до хоста призначення через одну або кілька мереж.
Рівень 3: Мережний.	Маршрутизує пакети даних між двома вузлами в мережі, використовуючи IP-адресу.
Рівень 2: Передачі даних.	Забезпечує надійне з'єднання між двома зв'язаними вузлами шляхом виявлення помилок на фізичному рівні, та передачу даних.
Рівень 1: Фізичний.	Передає бітовий потік через фізичне середовище, наприклад, через коаксіальний або волоконний кабель.

1.1.3 Функції системи моніторингу мережі

Системи моніторингу мережі забезпечують п'ять основних функцій:

- управління конфігурацією мережі;
- обробка помилок;
- аналіз продуктивності;
- управління безпекою;
- облік роботи мережі.

Моніторинг мережі починається з процесу виявлення її складових. Простіше кажучи, якщо не відомо, які модулі підключено до комунікаційного середовища та топологія зв’язків, неможливо

контролювати мережу. Системи моніторингу мережі (СММ) виявляють пристрої в мережі - маршрутизатори, комутатори, брандмауери, сервери, принтери та багато іншого.

СММ включають бібліотеку шаблонів моніторингу, яка визначає, як слід контролювати пристрій. У програмі WhatsUp Gold ці шаблони називаються ролями пристроїв.

Ролі пристроїв є типовими та специфічними для постачальника. Наприклад, те, що контролюється на маршрутизаторі Cisco, відрізнятиметься від того, що контролюється на сервері Dell.

Коли система моніторингу мережі конкурує з процесом виявлення, вона автоматично призначає відповідну роль кожному виявленому пристрою.

СММ відрізняються можливостями виявлення. Всі СММ виявляють пристрої в мережі. Однак не всі виявляють, як пристрої підключені до мережі. Наприклад, СММ може ідентифікувати сервер у мережі, але він не матиме інформації про те, до якого вузла він підключений.

Системи моніторингу мережі генерують карти мережі. Мережеві карти є потужним інструментом першої відповіді, який дозволяє адміністраторам мережі візуалізувати свої мережі. Вони забезпечують чисте і впорядковане подання шафи електропроводки. Мережа відображає пристрої відображення та оновлення статусу.

Багато СММ вимагають значного обсягу ручної обробки для створення карти мережі. Деякі лише надають інструмент для її проектування і використовують знання адміністратора мережі для відображення мережі [2].

Системи моніторингу мережі сповіщають адміністраторів мережі про некоректність функціонування. Вони надсилають сповіщення у текстовому повідомленні електронною поштою.

СММ налаштована на оповіщення, коли використання процесора на маршрутизаторі перевищує 80%. Це дає змогу адміністратору мережі

активно досліджувати та реагувати, перш ніж маршрутизатор повністю вийде з ладу.

Для збору, та дослідження даних, потрібних для моніторингу комп’ютерної мережі існують окремі протоколи, такі як Simple Network Management Protocol (SNMP) та Windows Management Instrumentation (WMI). Вони мають різні реалізації, але обидва мають місце для застосування.

1.1.4 Протокол SNMP

SNMP - це стандартний протокол, який збирає дані практично з будь-якого пристрою, підключеного до мережі, зокрема: маршрутизаторів, комутаторів, контролерів бездротової локальної мережі, бездротові точки доступу, сервери, принтери та багато іншого.

SNMP працює, запитуючи "Об'єкти", під якими розуміють модулі, про які СММ збирає інформацію. Наприклад, використання центрального процесору (ЦП) є об'єктом SNMP. Запити на об'єкт використання ЦП повертають значення, яке система моніторингу мережі (Network Monitoring System, NMS) використовує для оповіщення та звітування.

Об'єкти, запитувані протоколом SNMP, зберігаються в базі управляючої інформації (Management Information Base, MIB), що визначає всю інформацію, яка піддається впливу за допомогою керованого пристрою. Наприклад, MIB для маршрутизатора Cisco буде містити всі об'єкти, визначені Cisco, які можуть бути використані для моніторингу такого маршрутизатора, як використання процесора, використання пам'яті та стан інтерфейсу [3].

Об'єкти в MIB каталогізовані з використанням стандартизованої системи числення. Кожен об'єкт має свій унікальний ідентифікатор об'єкта або OID.

Деякі NMS надають браузер MIB, який дозволяє мережевим адміністраторам здійснювати навігацію через MIB, для знаходження

додаткових об'єктів, які вони хочуть контролювати на мережному пристрої.

1.1.5 Протокол WMI

WMI - це протокол для моніторингу серверів і додатків на основі Microsoft Windows. WMI є специфічним для Windows і не контролює мережні пристрої або сервери, що не належать до Microsoft.

WMI має велику бібліотеку з тисячами лічильників продуктивності. WMI може використовуватися для моніторингу майже всього на сервері Windows, який можна контролювати за допомогою SNMP.

Недоліком WMI, є більша ресурсомісткість для NMS, що споживає більше процесора і пам'яті для обробки, ніж SNMP.

Підприємства використовують різні бізнес-додатки, які встановлюються на серверах корпоративної мережі або центрі обробки даних для надання послуг хостам всередині організації. Є також додаткове керування мережею та користувачами, наприклад, DNS, Active Directory, DHCP тощо, які надаються з серверів. Крім того, користувачам або клієнтам в організації також потрібна операційна система. Серед численних варіантів, доступних для операційної системи, найчастіше використовуються ОС на базі Windows, як для серверних, так і для клієнтських хост-вимог на підприємстві.

Наявність бізнес-додатків на серверах вимагає постійного контролю використання ресурсів, таких як пам'ять, дисковий простір, кеш, процесор і багато іншого. Моніторинг також допомагає визначити можливі проблеми, які впливають на продуктивність сервера. Окрім серверів, клієнтські пристрої також потребують моніторингу для забезпечення безпроблемного користування мережею кінцевому користувачу

Системи на базі Windows можуть надавати дані системам моніторингу, які потім обробляють і використовують дані для звітування про продуктивність і корисну роботу серверів і робочих станцій. Дані, які

використовуються для моніторингу, можуть бути зібрані в операційній системі Windows за допомогою будь-яких доступних методів, описаних нижче.

1.2 Управління якістю передачі інформації

Якість обслуговування QoS (Quality of service) відноситься до будь-якої технології, яка управляє трафіком даних для зменшення втрати пакетів та затримки в мережі. Якість обслуговування контролює та управляє мережевими ресурсами, встановлюючи пріоритети для конкретних типів даних у мережі.

Корпоративні мережі повинні забезпечувати передбачувані та вимірювані послуги програмам для передачі голосу, відео та даних, що чутливі до затримки. Організації використовують контроль якості обслуговування (КЯО) для задоволення вимог до трафіку чутливих додатків, таких як передача голосових та відео потоків в реальному часі, а також для запобігання погіршенню якості, викликаної втратою пакетів, затримкою та помилками.

Організації для забезпечення КЯО, використовують певні інструменти та методи, такі як формування буфера різниці затримки пакетів. Для багатьох організацій КЯО включено до угоди про рівень обслуговування SLA (Service Level Agreement) з постачальником послуг мережі, щоб гарантувати певний рівень продуктивності.

1.2.1 Параметри якості обслуговування

Організації можуть кількісно вимірювати QoS, використовуючи декілька параметрів, включаючи наступні.

• Втрата пакетів відбувається, коли мережні пристрої стають перевантаженими, а маршрутизатори та комутатори починають відкидати пакети. Коли пакети втрачаються під час спілкування в реальному часі,

наприклад, у голосових або відеодзвінках, ці сеанси можуть відчувати різницю в затримці пакетів і прогалини в мові.

• Різниця в затримці є результатом перевантаження мережі, дрейфу часу та змін маршруту. Занадто багато різниці в затримці може погіршити якість голосового та відеозв'язку.

• Час затримки - це час, протягом якого пакет повинен переміщуватися від джерела до місця призначення. Затримка має бути максимально близькою до нуля. Якщо голосовий виклик через IP має високу затримку, він може відчувати відлуння та перекривання звуку.

• Пропускна спроможність - це здатність мережевої комунікаційної лінії передавати максимальну кількість даних від однієї точки до іншої за певний проміжок часу. КЯО оптимізує мережу, керуючи пропускною спроможністю та встановлюючи пріоритети для додатків, які потребують більшої швидкості, ніж інші.

• Середній бал (Mean opinion score, MOS) - це показник якості голосу, який використовує п'ятибальну шкалу, де п'ять вказує на найвищу якість.

1.2.2 Реалізація контролю якості обслуговування

Існують три моделі для реалізації КЯО: негарантована доставка, гарантована доставка (інтегровані послуги, далі ІП) та диференційовані послуги (ДП).

Негарантована доставка (Best Effort) - це модель QoS, де всі пакети отримують однаковий пріоритет і немає гарантованої доставки пакетів. Best Effort застосовується, коли мережі не налаштовували політики КЯО або коли інфраструктура не підтримує КЯО.

Гарантована доставка (ІП) - це модель КЯО, яка резервує пропускну спроможність по певному маршруту в мережі. Програми запитують мережу на резервування ресурсів, а мережеві пристрої стежать за потоком

пакетів, щоб переконатися, що поточне використання мережевих ресурсів дозволяє приймати пакети.

Реалізація ІП вимагає маршрутизаторів, що працюють з ІП і використовує протокол резервування ресурсів RSVP (Resource Reservation Protocol) для резервування мережевих ресурсів. ІП має обмежену масштабованість і високе споживання мережевих ресурсів.

Диференційовані послуги (ДП) - це модель КЯО, де мережеві елементи, такі як маршрутизатори та комутатори, налаштовані на обслуговування декількох класів трафіку з різними пріоритетами. Мережевий трафік повинен бути розділений на класи на основі вимог компанії.

Наприклад, голосовий трафік може бути визначений більш високим пріоритетом, ніж інші типи трафіку. Пакетам призначаються пріоритети за допомогою кодової точки диференційованих послуг DSCP (Differentiated Services Code Point) для класифікації. ДП також використовують визначення політики покрокового обслуговування PHB (Per-Hop Behaviour) для пакета, щоб застосувати методи КЯО, такі як чергування і визначення пріоритетів, до пакетів.

Мережева архітектура також впливає на те, як організація реалізує QoS. Мультипротокова мережа комутації міток MPLS (Multiprotocol Label Switching) включає в себе приватне посилання, яке пропонує комплексний КЯО по одному шляху. SLA для MPLS визначають смугу пропускання, QoS, затримку та тривалість роботи. Однак, MPLS може бути дорогим для організацій.

Програмне забезпечення Software-Defined Wide Area Network (SD-WAN) використовує кілька типів підключень, включаючи MPLS і широкосмуговий. SD-WAN відстежує стан поточних мережевих з'єднань для вирішення проблем із продуктивністю. Наприклад, якщо втрата пакетів перевищує певний рівень на одному з'єднанні, інструменти SD-WAN будуть шукати альтернативне з'єднання.

1.2.3 Механізми контролю якості обслуговування

Деякі механізми КЯО можуть керувати якістю трафіку даних і підтримувати вимоги QoS, зазначені в угодах SLA. Механізми QoS підпадають під конкретні категорії залежно від ролей, які вони відіграють у керуванні мережею. Розглянемо головні інструменти QoS.

• Інструменти класифікації та маркування розрізняють програми та сортують пакети в різні типи трафіку. Маркування позначить кожен пакет як частину мережевого класу, що дозволяє пристроям у мережі розпізнавати клас пакетів. Класифікація та маркування реалізовані на мережевих пристроях, таких як маршрутизатори, комутатори та точки доступу.

• Інструменти керування перевантаженнями використовують класифікацію пакетів і маркування, щоб визначити, в яку чергу записувати пакети.

1.2.4 Засоби моніторингу

Область інструментів моніторингу мережі, програмного забезпечення та постачальників дуже велика. Нове програмне забезпечення, інструменти та утиліти запускаються майже щороку, щоб конкурувати на постійно мінливому ринку IT-моніторингу та моніторингу серверів.

Деякі з функцій, які присутні у інструментів моніторингу, - це індикатори Uptime / Downtime, а також надійні та ретельні системи оповіщення (через електронну пошту / SMS), спеціальні шаблони та пороги, інтеграція з SNMP, функціональність автоматичного виявлення, зіставлення топології мережі та багато іншого.

Монітор продуктивності мережі SolarWinds, - це простий у налаштуванні інструмент, що автоматично виявляє мережеві пристрої і розгортається протягом години. Простий підхід до нагляду за цілою

мережею робить його одним з найпростіших у використанні та найбільш інтуїтивним для користувача.

Продукт легко настроюється і інтерфейс легко змінювати під свої потреби. Можна налаштувати інформаційні панелі, діаграми та перегляди на веб-основі, розробити топологію з урахуванням всієї мережевої інфраструктури, а також створити індивідуальні інтелектуальні сповіщення.

Програмне забезпечення PRTG Network Monitor широко відоме своїми передовими можливостями управління інфраструктурою. Всі пристрої, системи, трафік і програми у мережі можуть бути легко відображені в ієрархічному вигляді, який узагальнює продуктивність і оповіщення. PRTG відстежує IT-інфраструктуру, використовуючи такі технології, як SNMP, WMI, SSH, Потоки / Перехоплення пакетів, HTTP-запити, REST API, Pings, SQL та багато іншого.

Це один з найкращих варіантів для організацій з низьким досвідом моніторингу мережі. Інтерфейс користувача дійсно потужний і дуже простий у використанні.

Особливістю PRTG є можливість контролювати пристрої у центрі обробки даних за допомогою мобільного додатка. QR-код, що відповідає датчику, роздрукований і приєднаний до фізичного обладнання. Для сканування коду використовується мобільний додаток, а на екрані мобільного телефону відображається коротка інформація про пристрій.

ManageEngine OpManager використовується для управління інфраструктурою, моніторингу мережі та управління продуктивністю додатків APM програмного забезпечення.

Продукт добре збалансований, коли мова йде про функції моніторингу та аналізу.

Рішення може керувати мережею, серверами, конфігурацією мережі, несправністю та продуктивністю; Він також може аналізувати мережевий

трафік. Для запуску Manage Engine OpManager він повинен бути встановлений на робочій станції.

Головною відмінністю цього продукту є те, що він поставляється з попередньо налаштованими шаблонами пристроїв моніторингу мережі. Вони містять попередньо визначені параметри моніторингу та інтервали для конкретних типів пристроїв.

WhatsUp Gold (WUG) - це програмне забезпечення для моніторингу мережі від Ipswitch. Це один з найпростіших у використанні і висококонфігурованих інструментів на ринку.

Для щоденного управління IT, WhatsUp Gold є функціональним, збалансованим інструментом моніторингу мережі. Він також повністю настроюється. Панелі інструментів можна налаштувати для відображення IT-інфраструктури та оповіщень відповідно до вимог [4].

Таким чином, існуючі засоби моніторингу комп'ютерної мережі дають широкий спектр можливостей адміністраторам та звичайним користувачам. Але для отримання максимального розуміння роботи певної мережі, механізмів, параметрів, потрібні знання протоколів передачі та моніторингу. Інакше все, що на дисплеї – просто числа.

2. ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДІВ РЕАЛІЗАЦІЇ

2.1 Показники моніторингу

Сирі метрики, коли мова йде про моніторинг, це лише цифри, які нічого не повідомляють. Деяке число може здатися високим, але не призводить до виникнення будь-яких проблем, які можуть бути видимі для кінцевих користувачів. Інше число може здатися низьким, але не настільки низьким, як вимагають SLA. Єдиний спосіб зробити значущі метрики - розмістити їх у контексті того, як використовується програма та що очікують користувачі.

Всі ці фактори означають, що не можна дивитися на статистику ефективності з чисто технічної точки зору. Єдиний спосіб зрозуміти продуктивність - подивитися показники продуктивності з точки зору користувача. Це вірно навіть тоді, коли у вас є видимість в мережі та програмі. Мета моніторингу не повинна полягати в підвищенні показників за метриками.

У цій роботі я вибрав наступні основні метрики моніторингу мережі з боку користувача:

- Пропускна спроможність (bandwidth) – метрика, що напряму показує швидкість з якою в даній мережі можуть відправлятися та отримуватися дані. Я взяв в мегабайтах за секунду;
- Пакетна пропускна спроможність (PPS) – метрика, що показує швидкість з якою пакети можуть проходити через мережу. Виміряється в пакетах за секунду;
- Показник втрати пакетів (frame loss) – якщо неможливо відновити фрейм, або він не відповідає контрольній сумі, то він може бути відкинутий. Показник показує кількість втрачених пакетів та відсоток їх кількості;

- Час життя пакету TTL (Time To Live) – через скільки маршрутизаторів пакет може бути доставлений з однієї точки до іншої;
- Заголовки пакетів – потрібні для ручного контролю та дозволяють досвідченим користувачам отримувати повну інформації про пакети, що проходять через комп’ютер.

2.1.1 Обґрунтування пропускної спроможності

Пропускна спроможність – це максимальна швидкість, з якою можна отримувати, та відправляти дані в Інтернеті. Чим вона більша, тим більше даних можна отримати за деякий проміжок часу. В повсякденності вона виміряється в мегабітах за секунду. Слідує звернути увагу на те, що один мегабайт це вісім мегабіт. Якщо у вас швидкість один мегабіт в секунду – то для загрузки файлу розміром 1 мегабайт вам треба вісім секунд.

2.1.2 Обґрунтування пакетної пропускної спроможності

У доповнення до пропускної спроможності метрика, така як пакетна пропускна спроможність, також вельми важлива для повного розуміння продуктивності мережі. Інтерфейси мережевого пристрою працюють з лінійною швидкістю, коли пристрій здатний пересилати пакети незалежно від розміру. Таким чином, навіть для самих маленьких пакетів (найвища швидкість передачі пакетів) мережевий пристрій буде виконувати свої функції. Наприклад, оскільки один з новітніх маршрутизаторів Cisco ASR, здатний пересилати пакети зі швидкістю до 16 Мбіт / с з включеними послугами, він може підтримувати обробку еквівалента трафіку 10 Гбіт / с з лінійною швидкістю, навіть для невеликих пакетів [5].

Для маршрутизаторів пакетна пропускна спроможність є одним з найбільш важливих параметрів, але для інших пристроїв необхідно враховувати і інші метрики, такі як брандмауери, балансування

навантаження, системи запобігання вторгнень, пристрої трансляції NAT (Network Address Translation).

2.1.3 Обґрунтування показника втрати пакетів

Втрата пакету - це збій передачі одного або декількох пакетів в місці призначення. Ця подія може викликати помітні зміни у всіх типах цифрових комунікацій.

Наслідки втрати пакетів:

- Часткова втрата даних, що призводить до помилок у завантажених файлах;
- У відеоконференції, це може створити тремтіння картинки, через те, що для відтворення відео використовуються дані про зміни картинки, відносно попереднього кадру. Приклад на рисунку 2.1;
- У аудіозв'язку, це може викликати тремтіння і часті пропуски в прийнятій мові;
- У гірших випадках втрата пакета може привести до серйозного пошкодження отриманих даних, втрати більшої частини зображень, нерозбірливої мови або навіть повної відсутності прийнятого сигналу.

Рисунок 2.1 – Наслідок втрати пакету з даними для відтворення відео

Причини втрати пакетів включають в себе недостатню потужність сигналу в пункті призначення, природні або штучні перешкоди, надмірний шум системи, апаратний збій, пошкодження програмного забезпечення або перевантажені вузли мережі. Часто більш ніж один з цих факторів впливає на появу втрати пакетів.

У разі, коли причина не може бути усунена, може використовуватися маскування втрати пакета, щоб мінімізувати вплив втрачених пакетів.

2.1.4 Обґрунтування часу життя пакету (TTL).

Час життя (TTL) - це, в основному, кількість маршрутизаторів, які проходить пакет, перш ніж він буде відкинутий. Певні номера TTL вказують максимальний діапазон який може пройти пакет.

Початкове значення TTL задається при відправці хостом як поле з восьми двійкових цифр заголовка пакета. Поле TTL встановлюється відправником дейтаграми і зменшується кожним маршрутизатором на маршруті аж до місця призначення. При пересиланні IP-пакетів маршрутизатор зменшує значення TTL мінімум на 1. Коли значення TTL пакета досягає 0, маршрутизатор відкидає його і відправляє повідомлення з використанням протоколу ICMP назад на хост.

2.2 Мова програмування

Правильно вибрати мову програмування для конкретної задачі – це важлива частина роботи. Для реалізації даних можливостей була обрана мова Python. Вона дозволяє досить легко та елегантно створювати об’єктно орієнтовані програми та має ряд корисних внутрішніх модулів, що підходять для виконання поставленої задачі.

2.2.1 Модуль socket

Сокетне програмування – це спосіб з’єднання двох вузлів в мережі один з одним. Один сокет (вузол) слухає конкретний порт з IP-адреси, а

інший сокет звертається до іншого, щоб сформувати з'єднання. Сервер формує сокет слухача, коли клієнт звертається до сервера. Вони являють собою реальні основи для перегляду веб-сторінок. Говорячи коротко є сервер і клієнт.

Програмування сокета починається шляхом імпортування бібліотеки сокетів і створення простого сокета:

```
s = socket.socket (socket.AF_INET, socket.SOCK_STREAM)
```

Тут створюється екземпляр сокету та в нього передаються два параметри. Перший - AF_INET, відноситься до сімейства адрес. Другий - SOCK_STREAM значить орієнтований на з'єднання протокол TCP.

Тепер для прикладу можна підключитися до серверу за допомогою цього сокету. Треба звернути увагу, що якщо при створенні сокету виникає яка-небудь помилка – то генерується помилка socket.error та нам не вдасться підключитися до серверу. Для виправлення помилки нам потрібно буде знайти IP:

```
ip = socket.gethostbyname ('www.google.com')
```

Приклад підключення до Google:

```
import socket
import sys
try:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    print "Сокет успішно створений"
except socket.error as err:
    print "Створення сокету завершилося з помилкою %s" %(err)
port = 80
try:
    host_ip = socket.gethostbyname('www.google.com')
except socket.gaierror:
    print "Помилка хоста"
```

```
    sys.exit()
# підключення до серверу
s.connect((host_ip, port))
print "сокет успішно приєднано до google"
on port == %s" %(host_ip)
```

Також модуль socket дозволяє створювати програми типу клієнт-сервер. Сервер має метод bind(), що зв'язує його з IP та портом, щоб він міг прослуховувати вхідні запроси. Також метод listen(), що переводить сервер в режим прослуховування. Також методи accept() для з'єднання з клієнтом та close() для закриття з'єднання.

Модуль socket в даному проекті потрібен для:

- Передачі даних між процесами;
- Отримання потоку даних, що проходять через мережу.

2.2.2 Модуль PyQt5

PyQt5 це бібліотека, що дозволяє використовувати каркаси Qt GUI у мові програмування Python. Так як Qt написаний на мові C++, то швидкість створеного додатку буде достатньо високою, а написання мовою Python легше та швидше ніж на C++.

Модуль дозволяє створювати графічний інтерфейс з певними віджетами. Усі елементи, що бачить користувач у вікні є віджетами. Вони можуть бути вкладеними.

Самі популярні віджети:

- QLabel – мітка з текстом;
- QComboBox – для вибору одного з випадаючого списку, рис. 2.2;

- QCheckBox – для встановлення “галочки” на пункті, рис. 2.2;
- QRadioButton – вибір одного варіанту з декількох, рис 2.4;
- QPushButton – кнопка для натискання, рис 2.5;
- QTableWidget – таблиця зі значеннями, рис 2.6;
- QLineEdit – для вводу значень з клавіатури користувачем, рис 2.7;
- QSlider – для вибору приблизного значення переміщенням повзунка, рис 2.8;
- QProgressBar – для виводу прогресу виконання, рис. 2.9.

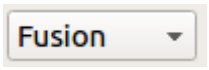

Рисунок 2.2 – QcomboBox.

Рисунок 2.3 – QCheckBox.

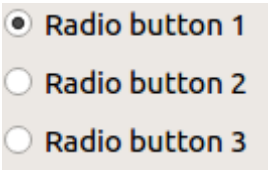

Рисунок 2.4 – QRadioButton.

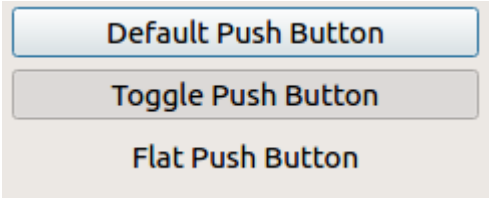

Рисунок 2.5 – QPushButton.

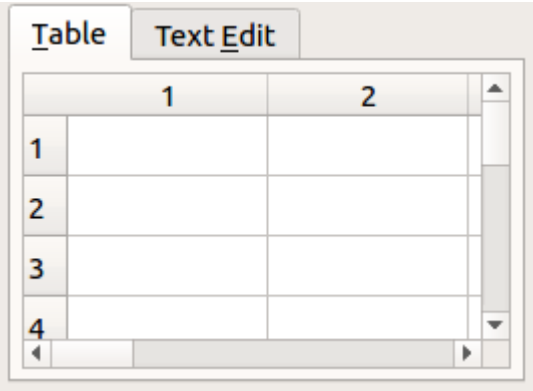

Рисунок 2.6 – QTableWidget.

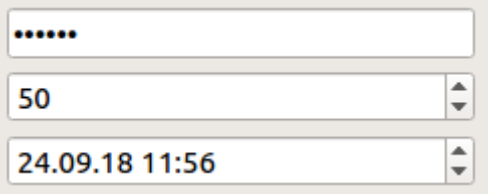

Рисунок 2.7 – QLineEdit.

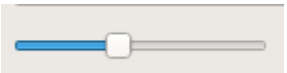

Рисунок 2.8 – QSlider.

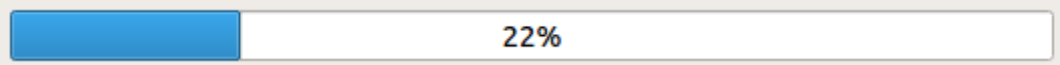

Рисунок 2.9 – QProgressBar.

Одна з сильних сторін Qt це підтримка стилів написаних творцем програми. Для використання специфічного стилю, треба використовувати setStyle(...). Доступні стилі залежать від платформи, найпопулярніші це 'Fusion', 'Windows', 'WindowsVista'(тільки для Windows) і 'Macintosh'(тільки для Mac).

2.2.2.1 Сигнали та слоти

Qt використовує механізми, що називаються сигналами, щоб дати можливість реагувати на події, такі як натискання кнопки, вибір пункту меню, встановлення “галочки” та інші.

Для прив’язки віджету до функції використовується метод connect(), що викликається для подій, що зарезервовані для кожного елементу.

Приклад прив’язки натискання кнопки до функції:

```
app = QApplication ([])
button = QPushButton ('Click')
def on_button_clicked ():
    alert = QMessageBox ()
    alert.setText («Ви натиснули на кнопку!»)
    alert.exec_ ()

button.clicked.connect (on_button_clicked)
button.show ()
app.exec_ ()
```

Button.clicked це так званий сигнал, а .connect() дозволяє встановити слот, що є простою функцією, що викликається при появі сигналу. Результат можна бачити на рис. 2.10.

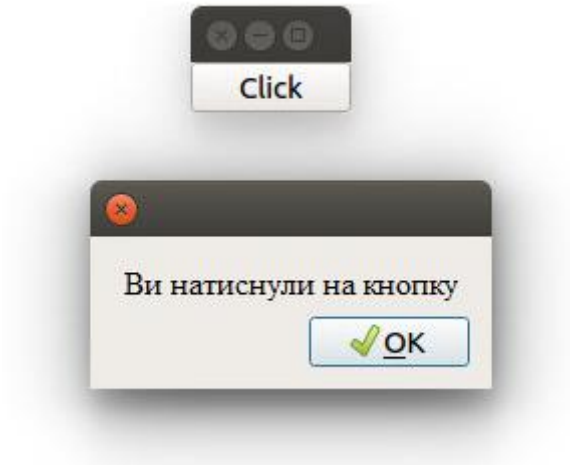

Рисунок 2.10 – результат прив’язки функції до кнопки.

3. РОЗРОБКА КОМПОНЕНТІВ СИСТЕМИ

3.1 Загальна структура системи

Розглянемо загальну структуру та функції розробленої системи.

Програмний продукт є системою, що складається з:

1) Модуля графічного інтерфейсу (вікно меню, вікно загальних параметрів, вікно TCP, вікно UDP, вікно HTTP, вікно іншої інформації);
2) Перехоплювача пакетів;
3) Модуля розбору пакетів на їх складові;
4) Модуля збору інформації для моніторингу якості мережі.

Структура зображена на рис. 3.1.

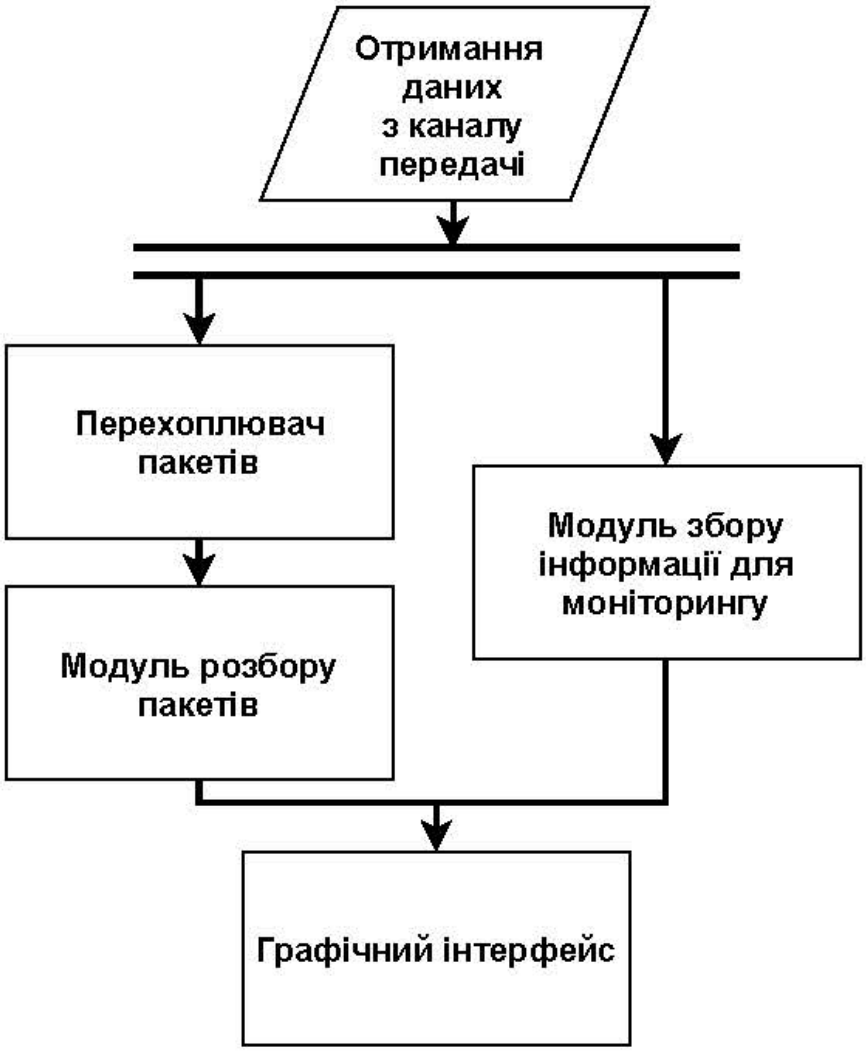

Рисунок 3.1 - Загальна структура системи

Розглянемо основні функції, що реалізує розробка.

- Отримання даних з каналу передачі даних.
- Отримання заголовків фрейму, IP, TCP, UDP, ICMP.
- Декодування даних, у тому випадку, коли вони не зашифровані.
- Отримання інформації для моніторингу, такої як пропускна спроможність, Packets Per Second (PPS), втрата пакетів, TTL та інших.
- Вивід інформації у графічні вікна програми.
- Представлення деяких видів інформації у вигляді графіку.

3.1.1 Модуль графічного інтерфейсу. Фреймворк PyQt5

PyQt5 – це набір прив’язок графічного фреймворку Qt для мови програмування Python.

PyQt5 включає в себе:

- Набір віджетів графічного інтерфейсу;
- Стилі віджетів;
- Доступ до баз даних SQL;
- Парсер XML;
- Підтримка SVG;
- Підтримка аудіо та відео;
- Qt Designer .

Qt Designer – крос-платформна програма для розробки дизайну вікон та віджетів.

Модулі PyQt5, що будут використовуватися:

- QtCore;
- QtGui.

QtCore – модуль, що містить в собі основні не графічні класи, систему сигналів та слотів, абстракції потоків, роздільної пам’яті, регулярних виразів та інші.

QtGui – модуль, що містить в собі компоненти графічного інтерфейсу та елементи управління.

Модуль графічного інтерфейсу – це модуль призначений для відображення даних, що надходять з інших модулів у інтерфейсі продукту. Він складається з імплементацій вікон та їх віджетів.

Імплементація вікон:

1) mainWindow.py;
2) generalInfo.py;
3) TCP.py;
4) UDP.py;
5) HTTP.py;
6) OTHER.py.

Імплементації представляють собою тільки оболонку з блоками у вікні створені за допомогою фреймворку PyQt5.

Віджети вікон:

1) myWindow.py;
2) generalWidget.py;
3) tcpWidget.py;
4) udpWidget.py;
5) httpWidget.py;
6) otherWidget.py.

Віджети вікон це реалізація отримання даних від інших модулів з їх виводом у вікно у класах. Використовуючи інтервальне переривання для періодичного виводу та мьютекси для попередження критичних секцій де дані можуть змінюватися іншим модулем.

Призначення класів:

1) mywindow(QtWidgets.QMainWindow) – клас основного вікна, що має екземпляри інших класів віджетів для їх відкриття при натисненні відповідних кнопок;

2) generalWin(QtWidgets.QMainWindow) – клас вікна головної інформації. Призначений для виводу такої інформації як пропускна спроможність, пакетів за секунду, кількість пакетів що не досягли призначення, графік часу життя пакетів та розбита інформація з пакетів у текстовому блоці;

3) tcpWin(QtWidgets.QMainWindow) – клас вікна інформації тільки про TCP пакети. Призначений для виводу розбитої інформації з TCP пакетів та графіку TTL для TCP пакетів;

4) udpWin(QtWidgets.QMainWindow) – клас вікна інформації тільки про UDP пакети. Призначений для виводу розбитої інформації з UDP пакетів та графіку TTL для UDP пакетів;

5) httpWin(QtWidgets.QMainWindow) – клас вікна інформації тільки про HTTP пакети. Призначений для виводу розбитої інформації з HTTP пакетів та графіку TTL для HTTP пакетів;

6) otherWin(QtWidgets.QMainWindow) – клас вікна інформації про пакети, що не розпізнані програмою як IPv4 пакети. Призначений для виводу короткої розбитої інформації про ці пакети.

3.1.2 Перехоплювач пакетів. Модуль socket

Модуль socket – модуль мови програмування python, що забезпечує використання сокетів для обміну даними між процесами, як на одній ЕОМ, так і на різних, що з’єднані мережею.

Методи модуля socket:

- bind()

  зв’язує адресу з сокетом;

- listen()

встановлює і запускає прослуховувач TCP;

- accept()
  пасивно приймає TCP-клієнтське з’єднання, очікуючи, поки не прийде з’єднання (блокує код);
- connect()
  ініціює підключення до серверу TCP;
- recv()
  отримує TCP повідомлення;
- send()
  відправляє TCP повідомлення;
- recvfrom()
  отримує UDP повідомлення;
- sendto()
  відправляє UDP повідомлення;
- gethostname()
  повертає ім’я хоста.

Перехоплювач пакетів за допомогою модуля python-socket дозволяє отримувати дані що проходять через мережу Та зберігати їх у PCAP файли для подальшого вивчення. Модуль буде працювати тільки з правами адміністратора.

Перехоплення відбувається у декілька етапів:

1) Відкриття сокету для отримання пакетів;
2) Створення екземпляра класу PCAP, що описаний у файлі pcap.py та призначений для запису даних у файл;
3) Запуск циклу для постійного отримання пакетів;
4) Спроба отримати пакет;
5) Запис пакету у файл;
6) Перехід у модуль розбиття пакетів на складові.

Реалізація даного модулю переплітається з модулем розбиття пакетів та описаний у пакеті sniffer.py в функціях, що починаються зі sniff.

3.1.3 Модуль розбору пакетів на їх складові

Модуль розбору пакетів на їх складові складається з класів описаних в файлах, що знаходяться в папці sniffer/networking та реалізації в файлі sniffer/sniffer.py та sniffer/general.py.

Короткий опис класів та методів.

1) class Ethernet: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, розбирає та записує з них в свої атрибути MAC адреси відправника та отримувача, версію протоколу IP та дані що передає пакет.

2) class IPv4: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, у випадку, якщо версія протоколу IPv4, розбирає та записує з них в свої атрибути версію, довжину заголовку, час життя пакету, протокол передачі та IP адреси відправника та отримувача.

3) class TCP: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, у випадку, якщо протокол передачі TCP, розбирає та записує з них в свої атрибути порти відправника та отримувача, запит, підтвердження, флаги URG, ACK, PSH, RST, SYN, FIN та дані, що передаються в пакеті.

4) class UDP: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, у випадку, якщо протокол передачі UDP, розбирає та записує з них в свої атрибути порти відправника та отримувача, розмір даних та самі дані, що передаються в пакеті.

5) class HTTP: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, у випадку, якщо протокол HTTP, розбирає та записує з них в свої атрибути декодовані дані, що передаються в пакеті.

6) class ICMP: def __init__(self, raw_data): - приймає пакет у вигляді послідовних байтів, у випадку, якщо отриманий ICMP

пакет, розбирає та записує з них в свої атрибути тип, код помилки, checksum та дані.

7) class GENThread(QtCore.QThread): def __init__(self, app): - клас, що характеризує собою окремо виконуваний процес, що збирає дані за допомогою функції def sniffGEN(app). Описаний у файлі widgets/generalWidget.py.

8) class TCPThread(QtCore.QThread): def __init__(self, app): - клас, що характеризує собою окремо виконуваний процес, що збирає дані за допомогою функції def sniffTCP(app). Описаний у файлі widgets/tcpWidget.py.

9) class UDPThread(QtCore.QThread): def __init__(self, app): - клас, що характеризує собою окремо виконуваний процес, що збирає дані за допомогою функції def sniffUDP(app). Описаний у файлі widgets/udpWidget.py.

10) class HTTPThread(QtCore.QThread): def __init__(self, app): - клас, що характеризує собою окремо виконуваний процес, що збирає дані за допомогою функції def sniffHTTP(app). Описаний у файлі widgets/httpWidget.py.

11) class OTHERThread(QtCore.QThread): def __init__(self, app): - клас, що характеризує собою окремо виконуваний процес, що збирає дані за допомогою функції def sniffOTHER(app). Описаний у файлі widgets/otherWidget.py.

12) def sniffGEN(app): - функція для отримання і розбору усіх пакетів, що проходять через мережу. Використовується вікном ui/generalInfo.py.

13) def sniffTCP(app): - функція для отримання і розбору TCP пакетів, що проходять через мережу. Використовується вікном ui/TCP.py.

14) def sniffUDP(app): - функція для отримання і розбору UDP пакетів, що проходять через мережу. Використовується вікном ui/UDP.py.

15) def sniffHTTP(app): - функція для отримання і розбору HTTP пакетів, що проходять через мережу. Використовується вікном ui/HTTP.py.

16) def sniffOTHER(app): - функція для отримання і розбору пакетів не розпізнаних як IPv4. Використовується вікном ui/OTHER.py.

3.1.4 Модуль збору інформації для моніторингу якості мережі

Модуль збору інформації для моніторингу якості мережі збирає дані про пропускну спроможність, кількість пакетів що пройшли через мережу за секунду, втрачені пакети та інформацію про TTL (графік та середнє значення).

Кожний віджет має свою константу MAX_TTL_ARR_COUNT, що вказує на те скільки максимально значень TTL можуть бути збережені та відображені на графіку.

Модуль реалізований окремо для класу кожного віджета та використовує наступні атрибути:

1) self.ttlArr – масив значень TTL, для послідовних пакетів;
2) self.ttlCount – кількість значень TTL в масиві self.ttlArr;
3) self.ttlSum – сума значень TTL в масиві self.ttlArr;
4) self.bandwidthOut – пропускна спроможність відправки за останню секунду;
5) self.bandwidthIn – пропускна спроможність отримання за останню секунду;
6) self.maxBandwidthOut – максимальна пропускна спроможність відправки за секунду;

7) self.maxBandwidthIn – максимальна пропускна спроможність отримання за секунду;

8) self.bandwidthTimer – таймер, що кожну секунду викликає метод getBandWidth;

9) def getBandWidth(self): - метод, що заміряє пропускну спроможність, кількість пакетів за останню секунду та втрачені пакети.

3.2 Розробка графічного інтерфейсу. Використання фреймворку PyQt5

Графічний інтерфейс – це важлива складова програмного продукту, так як він повинен буди простий та доступний для користувача.

3.2.1 Розробка дизайну

Дизайн розроблявся за допомогою додатку PyQt5 Designer. За допомогою цього додатка можна відносно легко створити дизайн інтерфейсу свого продукту та зберегти у файлі *.ui.

За допомогою пакету pyuic5 файли *.ui переформовувались у Python файли, створюючи в ньому клас UI_*, який надалі можна використовувати як облік вікна програми.

3.2.2 Розробка класів вікон

Класи вікон – це віджети, що вже мають не тільки UI а й логіку програми при взаємодії з інтерфейсом.

Для правильної роботи з віджетом в його конструкторі потрібно викликати конструктор класа QtWidgets.QMainWindow, створити прототип UI класу та викликати в ньому метод setupUi:

```
class generalWin(QtWidgets.QMainWindow):
    def __init__(self):
        super(generalWin, self).__init__()
        self.ui = Ui_generalObj()
        self.ui.setupUi(self)
```

Клік по кнопкам прив’язується до функції класа за допомогою метода connect:

```
self.ui.startButton.clicked.connect(self.startSniff)
```

Для представлення графіків використовувався блок графічного виводу QGraficView. Для його використання потрібно встановити вирівнювання по лівому-верхьому краю за допомогою методу setAlignment, створити екземпляр класу QtWidgets.QGraphicsScene, прив’язати його до QGraficView та створити екземпляр класу QtGui.QPen:

```
self.ui.ttlGraphic.setAlignment(Qt.AlignLeft|Qt.AlignTop)
self.ttlGraficScene = QtWidgets.QGraphicsScene()
self.ui.ttlGraphic.setScene(self.ttlGraficScene)
self.pen = QtGui.QPen(QtCore.Qt.green)
```

Після цього малювати лінії на даному блоці можна за допомогою методу addLine класу QGraphicsScene:

```
self.ttlGraficScene.addLine(QtCore.QLineF(x1, y1, x2, y2), self.pen)
```

Для виведення тексту на екран використовувався метод періодичного переривання інтерфейсу для виконання функції виводу накопиченої потоком перехоплювача пакетів набору інформації QtCore.QTimer. Для його використання потрібно було створити екземпляр класу QtCore.QTimer, за допомогою його методу setInterval встановити інтервал переривання в мілісекундах, за допомогою методу timeout.connect приєднати до функції виводу та запустити методом start:

```
self.outputTimer = QtCore.QTimer(self)
self.outputTimer.setInterval(200)
self.outputTimer.timeout.connect(self.updateAll)
self.outputTimer.start()
```

3.3 Розробка перехоплювача та розбору пакетів на їх складові. Використання модуля socket

Модулі перехоплення та розбору пакетів працюють в процесі, окремо від інтерфейсу для того, щоб запобігти зависанням програмного продукту.

Перехоплювач отримує дані в «сирому» вигляді та передає в модуль розбору пакетів, що читає заголовки та видає готові до читання дані про пакет.

3.3.1 Розробка модуля перехоплювача пакетів

Перехоплювач пакетів використовує модуль socket тому для отримання пакетів потрібно відкрити з'єднання з сокетом AF_PACKET, що дозволяє отримувати пакети через ядро операційної системи. Також потрібно використовувати параметр «сирого» сокету для контролю кожного біту заголовка пакету. Також параметр ntohs(3) для використання мережевих інтерфейсів низького рівня. Наступна стрічка коду дозволяє відкрити такий тип сокету:

```
conn = socket.socket(socket.AF_PACKET, socket.SOCK_RAW, socket.ntohs(3))
```

Після з'єднання для отримання перехоплення пакету потрібно використати метод recvfrom:

```
raw_data, addr = conn.recvfrom(65535)
```

3.3.2 Розробка модуля розбору пакетів

Розбір пакетів відбувається в декілька етапів та з використанням класів описаних у папці sniffer/networking.

Алгоритм модулю розбору пакетів зображений на рис. 3.2

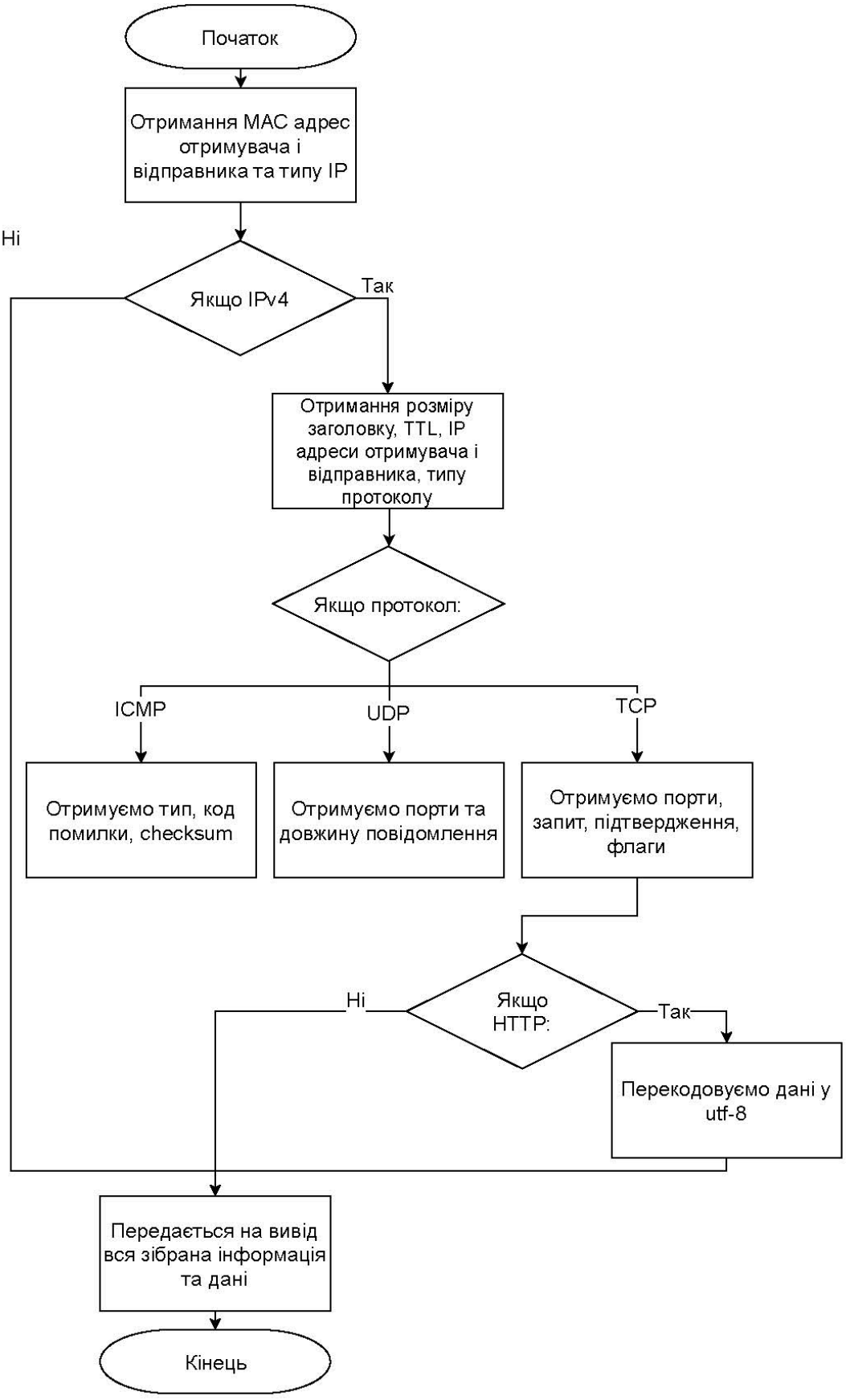

Рисунок 3.2 - Алгоритм модуля розбору пакетів

Опис реалізації структури.

1) На початку маємо Ethernet фрейм у якому перші 8 байт синхронізації, наступні 6 MAC адреса отримувача, наступні 6 MAC адреса відправника, наступні 2 поле типу, що вказує на версію IP.

2) Створюємо екземпляр класу sniffer.networking.ethernet. Ethernet, що отримує MAC адреси отримувача і відправника та поле prototype(з двох байтів поля TYPE, Ethernet фрейму).

3) Якщо отриманий prototype дорівнює 8 – то отриманий пакет версії IPv4. Тоді маємо IPv4 заголовок зображений на рис. 3.3.

4) Створюємо екземпляр класу sniffer.networking.ipv4.IPv4, що отримує розмір заголовку, TTL, IP адреси отримувача і відправника та поле протоколу.

5.1) Якщо поле протоколу дорівнює 6 то пакет переданий за протоколом TCP та маємо його структуру зображену на рис. 3.4. Тоді створюємо екземпляр класу sniffer.networking. tcp.TCP, що отримує порти отримувача та відправника, порядковий номер, номер підтвердження та флаги URG, ACK, PSH, RST, SYN, FIN.

5.1.1) Якщо порт відправника, або порт отримувача дорівнює 80 то пакет HTTP прикладного рівня. Тоді декодуємо дані у формати виводу utf-8 для перегляду в виводі детальних даних про пакет.

5.2) Якщо поле протоколу дорівнює 17 то пакет переданий за протоколом UDP та маємо структуру зображену на рис. 3.5. Тоді створюємо екземпляр класу sniffer.networking.udp. UDP, що отримує порти отримувача та відправника та довжину датаграми.

5.3) Якщо поле протоколу дорівнює 1 то пакет переданий за протоколом ICMP та маємо структуру зображену на рис. 3.6. Тоді створюємо екземпляр класу

sniffer.networking.udp.ICMP, що отримує тип, код та контрольну суму.

6) Після всього розбору на вивід передається уся зібрана інформація.

Версія	Довжина заголовку	Тип сервісу	Загальна довжина	
Ідентифікатор пакету			IP флаги x D M	Зміщення фрагменту
Час життя пакету		Протокол верхнього рівня	Контрольна сума	
IP адреса відправника 32 біти				
IP адреса отримувача 32 біти				
Опції і вирівнювання (не обов’язково)				

Рисунок 3.3 - Структура IPv4 заголовку

Порт відправника			Порт отримувача
Порядковий номер			
Номер підтвердження			
Довжина Заголовку	Резерв	Флаги	Розмір вікна
Контрольна сума			Вказівник на термінові дані
TCP параметри			

Рисунок 3.4 - Структура TCP

Порт відправника	Порт отримувача
Довжина датаграми	Контрольна сума

Рисунок 3.5 - Структура UDP

Тип (8 або 0)	Код(0)	Контрольна сума
Ідентифікатор		Номер по порядку

Рисунок 3.6 - Структура ICMP

3.4 Розробка модуля збору інформації для моніторингу якості мережі

В даному модулі для відображення необхідної інформації, а це пропускна спроможність за одиницю часу, максимальна пропускна спроможність, кількість пакетів за одиницю часу, максимальна кількість пакетів, кількість втрачених фреймів та відсоток втрачених фреймів, потрібно в класі головного вікна створити такі змінні:

- bandwidthOut - пропускна спроможність відправки;
- bandwidthIn- пропускна спроможність отримання;
- maxBandwidthOut – максимальна пропускна спроможність відправки;
- maxBandwidthIn – максимальна пропускна спроможність отримання;
- tx1, rx1 – кількість байт відправки та отримання в попередню секунду;
- tx2, rx2 – кількість байт відправки та отримання в дану секунду;
- packetsNow – спільна змінна для модулю перехоплювача та модулю збору інформації. Модуль перехоплювача збільшує змінну на одиницю, коли отримує пакет;
- packetsPerSecond – кількість пакетів, що були зібрані за останню секунду;
- maxPacketsPerSecond – максимальна кількість пакетів, що були зібрані за останню секунду;
- bandwidthTimer – таймер, що кожну секунду отримує значення показників;

- allPacketsCount – спільна змінна для модулю перехоплювача та модулю збору інформації. Модуль перехоплювача збільшує змінну на одиницю, коли отримує пакет;
- lossPacketsCount – спільна змінна для модулю перехоплювача та модулю збору інформації. Модуль перехоплювача збільшує змінну на одиницю, коли пакет було втрачено.

Значення для змінних tx1 та tx2 будуть отримуватися із системного файлу операційної системи Linux /sys/class/net/your_adapter/statistics/ tx_bytes/. Для змінних rx1 та rx2 з файлу /sys/class/net/your_adapter/statistics/ rx_bytes/.

Завдяки таймеру кожну секунду з файлів буде зчитуватися кількість байт у змінні tx2 та rx2, після чого різниця між tx2 та tx1 буде записуватись у змінну bandwidthOut, а різниця між rx2 та rx1 у змінну bandwidthIn, а змінні tx2 та rx2 будуть приймати значення tx1 та rx1 відповідно. Якщо змінна bandwidthOut стане більшою ніж maxBandwidthOut, то друга візьме значення першої. Аналогічно для bandwidthIn та maxBandwidthIn.

Також кожну секунду змінна packetsPerSecond приймає значення змінної packetsNow, а друга присвоюється нулю. Виводиться у вікно змінна lossPacketsCount та lossPacketsCount / allPacketsCount * 100 + ‘%’, для відображення відсотку втрачених пакетів.

4. АНАЛІЗ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

4.1 Запуск програми

Для запуску програми необхідно встановити python 3.7.3, завантаживши його з офіційного сайту. Також встановити такі пакети:

```
pip3 install --user pyqt5
sudo apt-get install python3-pyqt5
sudo apt-get install pyqt5-dev-tools
sudo apt-get install qttools5-dev-tools
```

Файл для запуску – getPackets.py

4.2 Тестування роботи

Просте тестування та моніторинг мережі. Запускаємо файл getPackets.py та бачимо головне меню зображене на рис. 4.1.

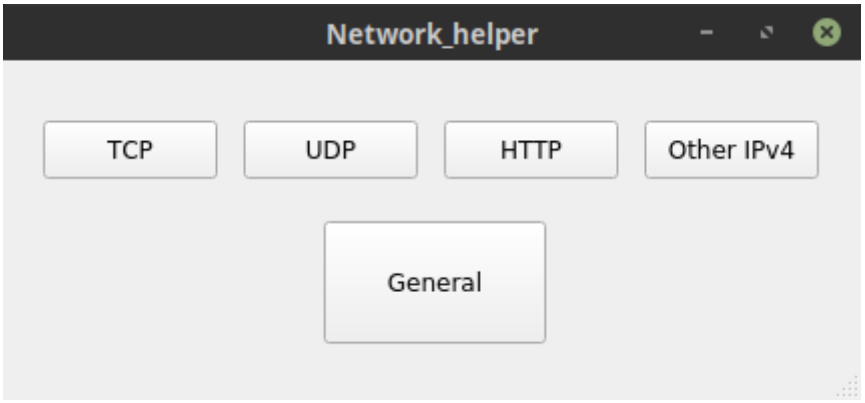

Рисунок 4.1 – Головне меню додатку.

Якщо натиснути на кнопку General відкриється вікно рис. 4.2 та процес розпочнеться. На кожній сторінці присутні кнопки управління для початку або зупинки роботи модулів перехоплювача та розбору, а також кнопка Clear, що видаляє всі збережені в програмі до цього дані.

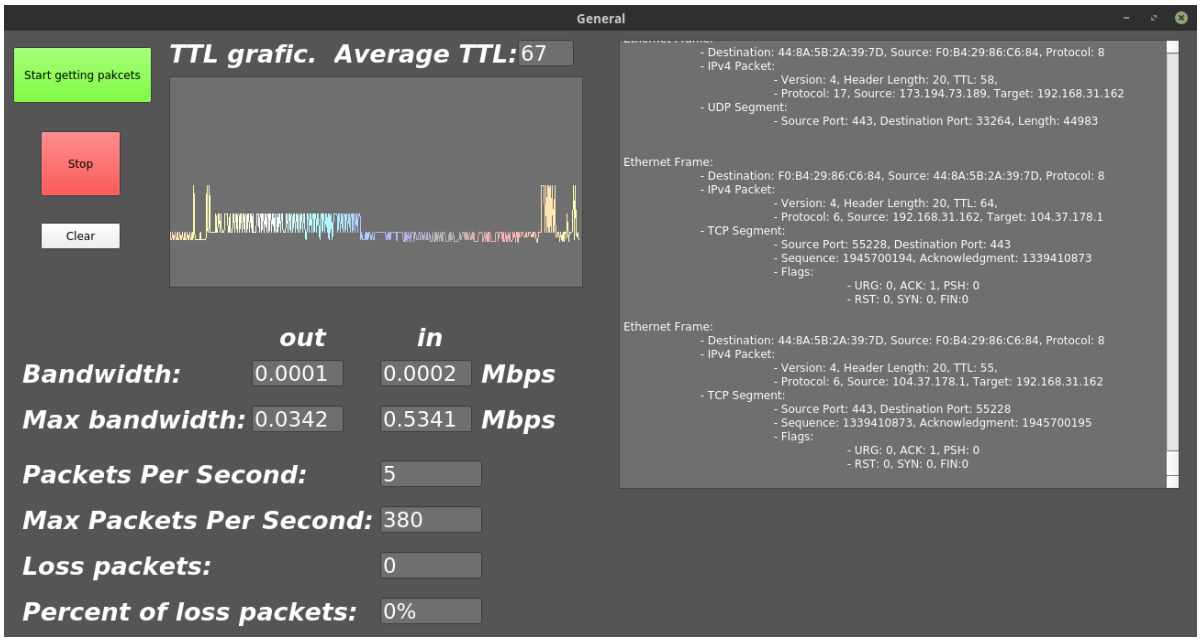

Рисунок 4.2 – Вікно General

Спробуємо загрузити мережу по максимуму рис. 4.3.

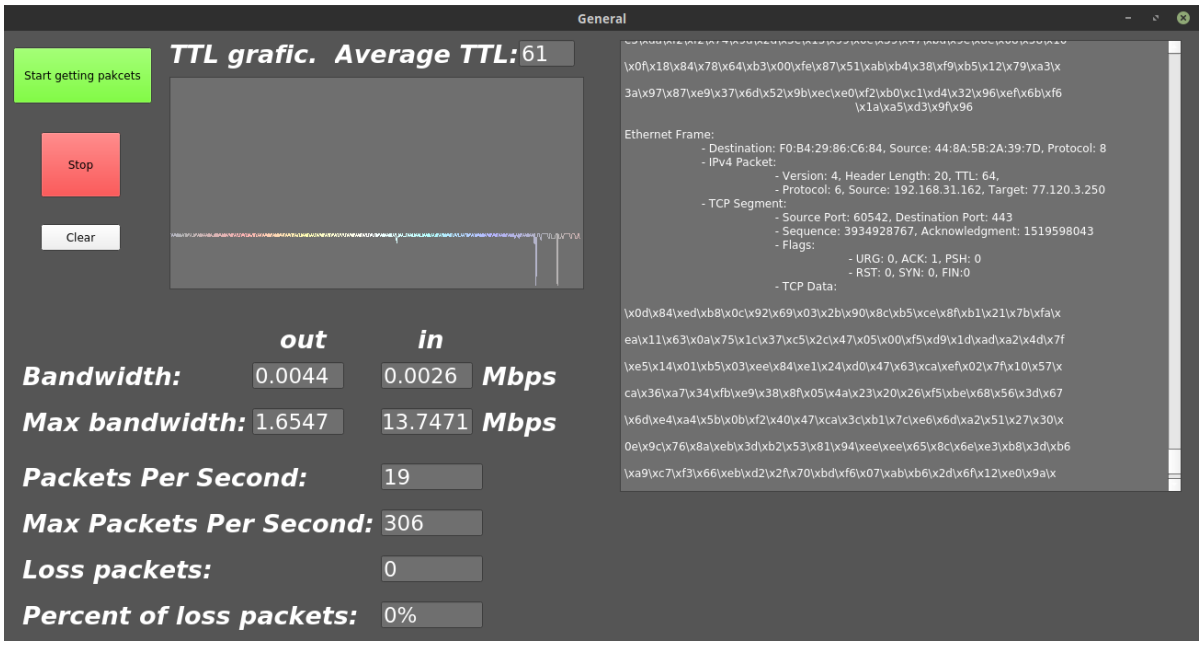

Рисунок 4.3 – Тест максимальної загрузки. Вікно General

Мережа, яка тестувалася, видала пікову швидкість отримання 13.74 мегабайти в секунду, віддачі 1.65 мегабайтів в секунду, 306 пакетів в секунду та ніякі пакети не були втрачені. TTL 61, нормальний для моєї мережі. Передача з серверу до клієнту 2-4 маршрутизатори. Пікова

пропускна спроможність досягається передачею TCP, або UDP пакетів з великою кількістю даних. Дані не відображаються у нормальному вигляді, через те, що вони зашифровані.

Не зашифровані дані можуть передаватися у http пакетах, які можна відстежувати у вікні HTTP рис. 4.4. Серед цих даних можна побачити такі поля:

- Host – доменне ім’я сторінки, до якої запитується доступ, наприклад “s.somehost.to”;
- Connection – відомості про проведення з'єднання, наприклад “keep-alive”;
- User-Agent – список назв і версій клієнта і його компонентів з коментарями, наприклад “Mozilla/5.0 (X11; Linux x86-64)”;
- Accept – список допустимих форматів сторінки, наприклад “text/css, */*;q=0.1”;
- Referer – Uniform Resource Identifier (URI) ресурсу, після якого клієнт зробив поточний запит, наприклад “http://uri.com.ua/”;
- Accept-Encoding – перелік підтримуваних способів кодування вмісту при передачі, наприклад “gzip, deflate”;
- Accept-Language – список підтримуваних мов, наприклад “en-US,en;q=0.9,ru;q=0.8”.

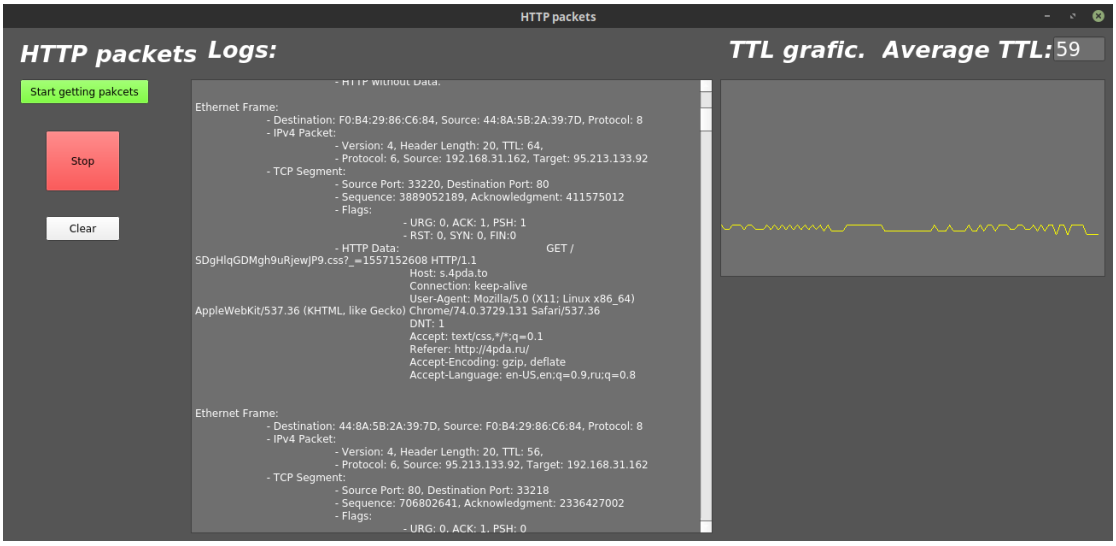

Рисунок 4.4 – Вікно HTTP

Для окремого відображення TCP пакетів потрібно відкрити вікно tcp рис 4.5.

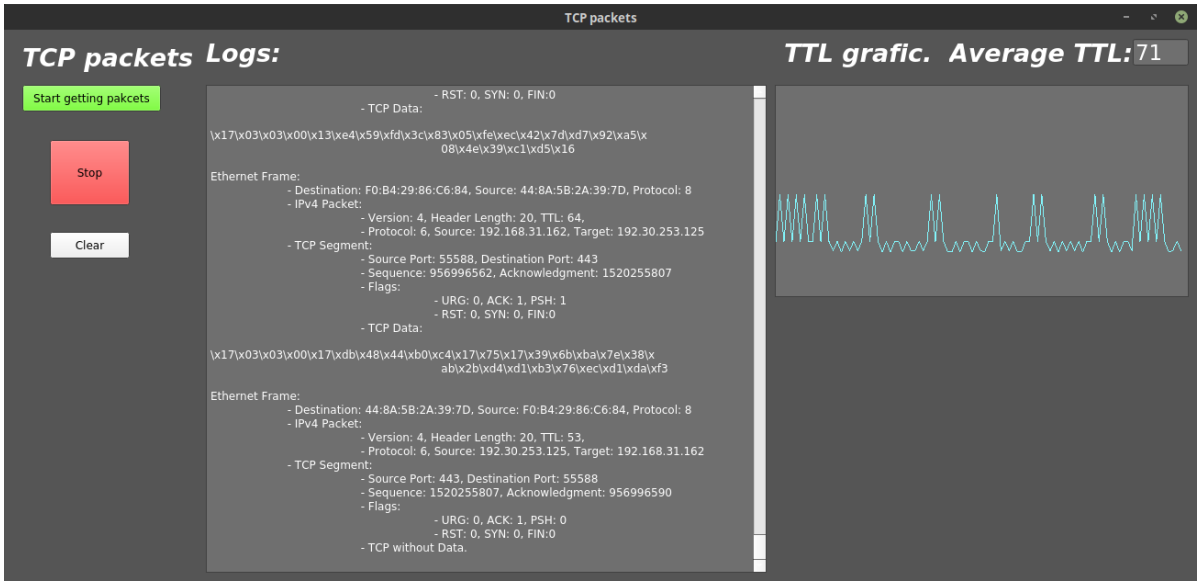

Рисунок 4.5 – Вікно TCP

Для окремого відображення UDP пакетів потрібно відкрити вікно udp рис 4.6.

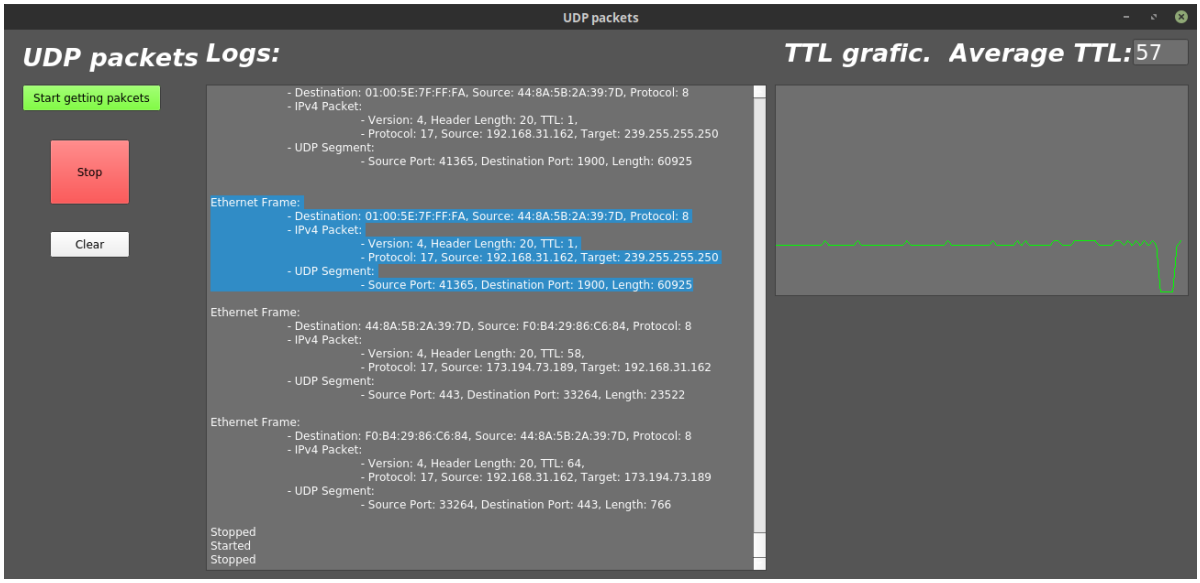

Рисунок 4.6 – Вікно UDP

Для окремого відображення інших фреймів, що проходять через робочу станцію потрібно відкрити вікно other рис 4.7.

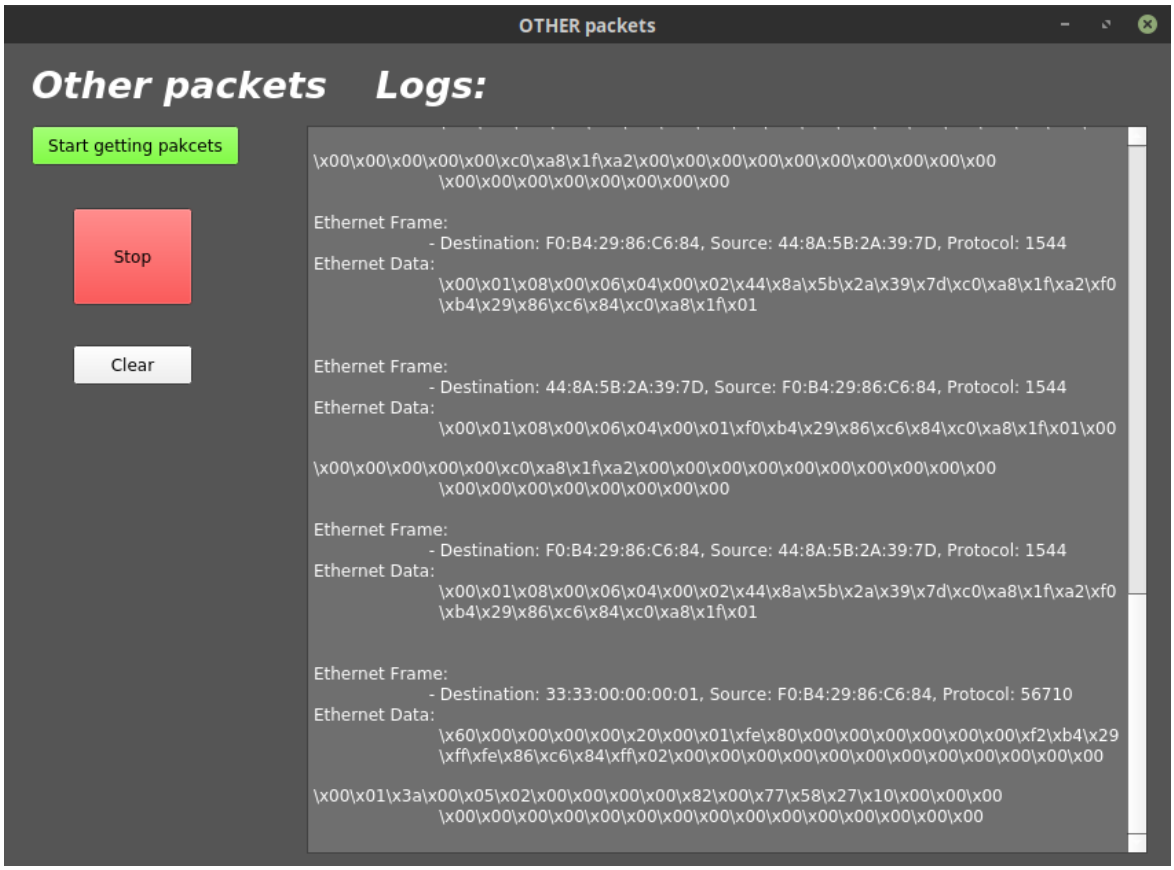

Рисунок 4.7 – Вікно Other

ВИСНОВКИ

В ході виконання даного дипломного проекту було проаналізовано методи та інструменти для моніторингу та управління якістю обслуговування комп’ютерної мережі. Запропоновано додаток для операційної системи Linux.

Аналіз існуючих рішень в першому розділі показав, що проблема моніторингу комп’ютерної мережі була та є актуальною, як для великих підприємств, так і для звичайного користувача. Були розглянуті методи для моніторингу та управління якістю, а також існуючі засоби. Було вивчено модель OSI, протоколи для моніторингу та параметри якості обслуговування.

У другому розділі були розглянуті інструменти для реалізації, обґрунтування їх вибору та описані основні метрики для моніторингу якості обслуговування комп’ютерної мережі. Якщо не стежити за станом якості обслуговування, в майбутньому, можуть виникнути проблеми, що описані в розділі.

У третьому розділі продемонстровано структуру проекта, розбиту на модулі. Описано алгоритми, класи та функції, задіяні в програмі та процес їх розробки.

Четвертий розділ демонструє процес запуску та використання додатку та представляє собою коротке керівництво користувача.

Розроблений додаток має потенціал до розширення функціоналу, так як велика частина одержуваної інформації виводиться у вигляді тексту, але її можна обробляти, сортувати та виводити додаткову інформацію та статистику для моніторингу та управління якістю комп’ютерної мережі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Вілсон Е. Моніторинг і аналіз мережі: підхід до усунення несправностей. Цинциннаті, 1998-2009 рр. 350 с.
2. Чіу Д. М., Садама Р. Моніторинг мережі: Розробка та застосування. Мічиган: Мічиганський університет, 1992. 207 с.
3. Хунг-Чанг Д., Пек-Янг Ч., і Чжі-Лі Ч. Масштабований моніторинг мережі у високошвидкісних мережах. 2011. 148 с.
4. Бежтліч Р. Дао моніторингу безпеки мережі. 2004. 832 с.
5. Бежтліч Р. Практика моніторингу мережної безпеки: розуміння інциденту. 2013. 376 с.