

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук та технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка
до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНО-
АПАРАТНИХ ЗАСОБІВ ВИМІРЮВАННЯ ХАРАКТЕРИСТИК
ПЕРЕДАВАННЯ ДАНИХ.
RESEARCH AND SOFTWARE IMPLEMENTATION OF SOFTWARE
AND HARDWARE FOR MEASURING THE CHARACTERISTICS OF
DATA TRANSMISSION

Виконав: студент 2 курсу, групи КНТ-129м
спеціальності

121 Інженерія програмного забезпечення
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Осташко Р. В.

(прізвище та ініціали)

Керівник Олійник А.О., к.т.н., доцент
(прізвище та ініціали)

Рецензент Гофман Є.О., к.т.н.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
 (повне найменування закладу вищої освіти)

Інститут, факультет _____ ІРЕ, КНТ _____
 Кафедра _____ програмних засобів _____
 Ступінь вищої освіти _____ магістр _____
 Спеціальність _____ 121 Інженерія програмного забезпечення _____
 (код і назва)
 Освітня програма (спеціалізація) _____ Інженерія програмного забезпечення _____
 (назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри проф., д.т.н. _____

С. О. Субботін

“ ” _____ 2020 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА

Осташка Романа Віталійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) _____ Дослідження та реалізація програмно-апаратних засобів вимірювання характеристик передавання даних. Research and Software Implementation of Software and Hardware for Measuring the Characteristics of Data Transmission _____

керівник проєкту (роботи) _____ Олійник А.О., к.т.н., доцент _____,
 (прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “03” листопада 2020 року №301

2. Строк подання студентом проєкту (роботи) _____ 5 грудня 2020 року _____

3. Вихідні дані до проєкту (роботи) _____ технічне завдання _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____ 1. Аналіз предметної області. 2. Вибір та обґрунтування структури системи, яка проєктується. 3. Основні рішення щодо реалізації компонентів системи. 4. Керівництво програміста. 5. Керівництво оператора. 6. Економіко-організаційна частина. 7. Охорона праці та безпека у надзвичайних ситуаціях. _____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
 _____ Слайди презентації _____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1–5	Олійник А.О., доцент		
6	Пожуєва Т.О., професор		
7	Коробко О.В., ст. викладач		
Нормо-контроль	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання 02.10.20

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	1 тиждень	
2	Розробка та затвердження технічного завдання	2 тиждень	Технічне завдання
3	Проектування структури системи	3-4 тиждень	Структурна схема
4	Розробка схеми функціонування системи та модулів програми	5-6 тиждень	Функціональна схема. Модулі
5	Створення програмного коду системи	7-8 тиждень	Текст програми
6	Тестування та відлагодження програми	9 тиждень	Працездатна система
7	Розробка розділів пояснювальної записки	10 тиждень	ПЗ
8	Розробка слайдів презентації	11-12 тиждень	Слайди презентації
9	Захист роботи	13 тиждень	

Студент(ка)

(підпис)

Осташко Р.В.

(прізвище, ініціали)

Керівник проєкту (роботи)

(підпис)

Олійник А.О.

(прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
142 с., 27 рис., 4 додатки, 24 джерела.

ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ОПЕРАЦІЙНА СИСТЕМА, SLA, JITTER, WIPROBE, SLAMOND, OWD, RFC, ETHERNET, CLI, IPDV.

Об'єкт дослідження – процес вимірювання якості зв'язку.

Предмет дослідження – засоби вимірювання характеристик передавання даних.

Мета роботи – дослідження та реалізація засобів вимірювання характеристик передавання даних.

Матеріали, методи та технічні засоби: структурне та об'єктно-орієнтоване програмування, мова програмування C++, персональний комп'ютер з монітором, клавіатурою та маніпулятором типу «миша», що функціонує під управлінням операційної системи.

Результати. Створено програмне забезпечення для вимірювання характеристик передавання даних. Розроблена програмна система може успішно експлуатуватися на персональних комп'ютерах під управлінням операційних систем та використовуватися для вимірювання характеристик передавання даних.

Висновки. Розроблено програмне забезпечення для вимірювання характеристик передавання даних. Проаналізовано предметну область, розроблено технічне завдання для реалізації програми, обрано середовище розробки, розроблено структурну схему програми, виконано конструювання програмного забезпечення для вимірювання характеристик передавання даних, здійснено тестування розробленого програмного забезпечення.

Галузь використання – програмно-апаратні комплекси для вимірювання якості зв'язку.

ABSTRACT

Explanatory note to the final qualifying work of the magister: 142 pages, 27 figures, 4 appendixes, 24 sources.

SOFTWARE, OPERATING SYSTEM, SLA, JITTER, WIPROBE, SLAMOND, OWD, RFC, ETHERNET, CLI, IPDV..

Object of study is a process of measuring the quality of communication.

Subject of study are means of measuring the characteristics of data transmission.

The purpose of the work is to research and implementation of means of measuring the characteristics of data transmission.

Materials, methods, and tools: structured and object-oriented programming, programming language C++, computer with a monitor, keyboard and mouse manipulator operating under the control of the operating system.

Results. The software for measuring data transmission characteristics has been created. The developed software system can successfully operate on personal computers running operating system and can be used to measure data transmission characteristics.

Conclusions. Software for measuring data transmission characteristics has been developed. The subject area is analyzed, the technical task for program realization is developed, the development environment is chosen, the structural scheme of the program is developed, the software for measurement of characteristics of data transmission is executed, the developed software is tested.

The field of use is software and hardware systems for measuring the quality of communication.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Аналіз систем вимірювання якості характеристик передавання даних	11
1.1.1 Carrier Ethernet.....	12
1.1.2 EtherNID	15
1.2 CISCO ip sla.....	19
1.3 Порівняльний аналіз засобів вимірювання якості характеристик передавання даних.....	22
1.4 Висновки за розділом 1	25
2 Вибір та обґрунтування структури системи, яка проєктується.....	26
2.1 Аналіз та вибір апаратної платформи для реалізації та використання розроблюваного програмного забезпечення	26
2.2 Аналіз та вибір мови програмування для розробки програмного забезпечення	33
2.3 Аналіз фреймворків для розробки програмного забезпечення	38
2.4 Висновки за розділом 2	42
3 Основні рішення щодо реалізації компонентів системи.....	44
3.1 Визначення вимірюваних метрик.....	44
3.2 Модель програмного забезпечення вимірювання характеристик передавання даних.....	47
3.3 Архітектура головної програми вимірювання характеристик передавання даних	51
3.4 Архітектура конфігураційної програми.....	60
3.5 Функціональне та навантажувальне тестування програмного забезпечення	70
3.6 Висновки за розділом 3	74
4 Керівництво програміста	77

	7
4.1 Призначення й умови застосування програми.....	77
4.2 Характеристики програми.....	77
4.3 Звернення до програми.....	78
4.4 Налаштування програми.....	78
5 Керівництво оператора	79
5.1 Призначення програми	79
5.2 Умови виконання програми	79
5.3 Виконання програми.....	79
5.4 Приклад роботи з програмою	81
6 Економіко-організаційна частина.....	86
6.1 Планування розробки програмного продукту.....	86
6.2 Визначення витрат на розробку програми	87
6.3 Розрахунок економічної ефективності програмного продукту.....	94
7 Охорона праці та безпека у надзвичайних ситуаціях.....	96
7.1 Аналіз потенційних небезпек.....	96
7.2 Заходи щодо забезпечення безпеки.....	97
7.3 Заходи щодо виробничої санітарії та гігієни праці	99
7.4 Заходи з пожежної безпеки	107
7.5 Заходи щодо забезпечення безпеки в надзвичайних ситуаціях	109
Висновки	112
Перелік джерел посилання	114
Додаток А Технічне завдання	116
Додаток Б Опис програми	120
Додаток В Текст програми	123
Додаток Г Слайди презентації	134

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

IP – Internet Protocol;
SLA – service-level agreement;
RFC – request for comments;
DNS – domain name system;
wiSLA – well integrated SLA;
IPDV – IP packet delay variation;
PDV – packet delay variation;
NTP – network time protocol;
RAM – random-access memory;
HTTP – hypertext transfer protocol;
VLAN – virtual local area network
QoS – quality of service;
SAA – Service Assurance Agent;
MEF – metro ethernet forum;
IT – інформаційні технології;
БЗ – база знань;
ОС – операційна система;
ПЗ – програмне забезпечення;
ПО – предметна область;
ПК – персональний комп'ютер;
ПП – програмний продукт;
ТЗ – технічне завдання.

ВСТУП

З розширенням поняття зв'язку від традиційної телефонії до стільникових послуг і доступу в Інтернет на ринку послуг зв'язку все міцніше затверджуються IP-технології. Технології передачі даних на основі TDM виявилися не в змозі встигнути за швидко розвиваються пакетними послугами, в рамках як телефонії (broadband ISDN – ATM, X.25), так і технологій локальних мереж передачі даних (DECnet, BanyanVines, AppleTalk, Novell IPX, TCP/IP та ін.). Однак пакетні технології, незважаючи на величезний потенціал розвитку, принесли з собою і невідомі раніше проблеми, пов'язані з природою пакетної комутації.

На даному етапі оператори зв'язку та споживачі послуг зіткнулися з тим, що показники якості Ethernet – на відміну від технологій старого покоління (E1, PDH, SDH та ін.) – не визначені, не фіксовані і можуть змінюватися в широкому діапазоні. А зважаючи на різноманітність пакетних сервісів, величезної кількості родин протокольних стеків пакетної комутації і постачальників обладнання стало неможливим формування нормативної бази якості IP-послуг.

Для успішної роботи IT-систем і хмарних сервісів потрібен надійний зв'язок між віддаленими компонентами, які часто можуть знаходитися в різних офісах, містах і країнах. За відсутності нормативної бази договір про рівень обслуговування між постачальником послуги і споживачем – Service Level Agreement (SLA) – стає єдиним способом врегулювання взаємовідносин, а також нормування показників якості та реалізації політики в галузі контролю та забезпечення якості.

Існує не так багато продуктів, які так чи інакше вирішують задачу моніторингу якості послуг. Одні забезпечують контроль показників продуктивності IP-з'єднань, інші – аналіз ефективності роботи додатків і мережевого устаткування. Вартість комплексу контрольно-вимірювальної апаратури для оцінки параметрів сигналів, що надходять з лінії зв'язку при

існуючих цифро-аналогових методах перетворення становить тисячі доларів США. У зв'язку з цим актуальною є розробка спеціальної вимірювальної системи, яка б виконувала контроль якості ліній зв'язку з максимальною точністю та за доступною ціною.

Таким чином, в даній роботі розробляється програмно-апаратний комплекс, що дозволяє проводити оцінку якісних характеристик ліній зв'язку.

Метою роботи є дослідження та реалізація засобів вимірювання характеристик передавання даних.

Для досягнення поставленої мети необхідно було розв'язати такі завдання:

- аналіз та порівняння існуючих засобів вимірювання характеристик передавання даних;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка моделі програмного забезпечення вимірювання характеристик передавання даних;
- конструювання програмного забезпечення відповідно до технічного завдання;
- тестування розробленого програмного забезпечення.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз систем вимірювання якості характеристик передавання даних

На певному етапі успішного розвитку будь-якого ІТ-проєкту перед тими, хто приймає рішення, виникає дилема. Як підтримувати працездатність бізнесу?

Можливостей дві – власними силами, тобто шляхом непрямих витрат, або на певних договірних умовах залучати для цих функцій сторонні організації, що спеціалізуються в ІТ-аутсорсингу.

У більшості випадків до недавнього часу вибір робився в бік внутрішніх ресурсів – спеціалізованих ІТ-підрозділів, відділів, служб, дочірніх структур.

Однак безліч негативних прикладів змусило керівників задуматися: чи варто йти на настільки відчутні внутрішні фінансові та – перш за все – людські витрати? За словами керівників багатьох ІТ-компаній, вони побоюються втратити багатьох цінних інженерів, які можуть бути незадоволеними цілодобовим робочим графіком.

Не тільки середні, а й досить великі компанії останнім часом вичерпали власні резерви для підтримки своєї повної функціональності. Таку ситуацію можна назвати типовою.

З одного боку, різке збільшення аутсорсингу і більш ніж оптимістичні прогнози на майбутнє. Крім того, певні складності в управлінні створює виникнення все нових і нових проєктів в ІТ-сфері.

З іншого боку, необхідна чітка схема взаємин між організаціями, що надають інформаційні послуги, і споживачами цих послуг. На сцену виходить необхідність регулювання таких питань шляхом угод на надання послуг з належною якістю.

Як підійти до опису необхідного рівня якості, чим керуватися? Подібна практика отримала досить вичерпну назву – Service Level Agreement (SLA). Зміст терміна SLA досить чітко розкрито в ньому самому.

SLA – це контракт, який регламентує відносини між сервіс-провайдером і його клієнтом. Ступінь важливості цього документа передбачає належні зусилля для його грамотної розробки [1].

На поточний день існує безліч програмних і апаратних рішень контролю якості передачі даних по мережі Інтернет. Можливості SLA часто використовується для оцінки якісних характеристик мережі різними програмно-апаратними комплексами. Найбільш цікавими серед них є Carrier Ethernet, EtherNID, CISCO ip sla. Перераховані системи моніторингу мережі є найбільш відомими на даний момент. У зв'язку з цим в даному пункті проводиться аналіз даних систем, призначених для контролю якості стану мережі.

1.1.1 Carrier Ethernet

ETX-203A – це демаркаційний пристрій Carrier Ethernet (рис. 1.1), що дозволяє оператору надавати бізнес-послуги згідно SLA аж до майданчика користувача на основі Ethernet, як в мережах SDH/SONET. За допомогою розробленої в RADe мікросхеми, пристрій ETX-203A передає трафік таких послуг, як IP VPN, VoIP, виділений доступ в Інтернет і LAN-to-LAN Layer 2, все з диференційованим рівнем якості послуг і наскрізним моніторингом.



Рисунок 1.1 – Carrier Ethernet ETX-203A

Особливості та переваги:

- демаркаційне рішення для передачі трафіку бізнес-послуг Ethernet на основі SLA;
- сертифікація MEF-9 і MEF-14 для послуг EPL і EVPL;

- модульна версія: 4 користувальницьких порту Fast Ethernet або Gigabit Ethernet; мережеві модулі: GbE, E1 / t1, t3, ShDSL;
- фіксована версія: 4 порту Fast Ethernet або Gigabit Ethernet (мережевих і призначених для користувача);
- поліпшене керування трафіком з формуванням по потокам Ethernet і h-QoS по кожному EVC / EVC.Cos;
- OAM Ethernet, моніторинг продуктивності, вбудований тестер rFC-2544, перевірки по шлейфу L2 / L3;
- захисне перемикання лінії Ethernet (ELPS) згідно ItU-t G.8031;
- підтримка Sync-E, 1588 tC;
- rADview-EMS керування; конфігурація через CLI;
- підтримка рішень rAD для моніторингу продуктивності.

Пристрій ETX-203A (рис. 1.1), може поставлятися в модульній або фіксованій конфігурації. Фіксована версія містить 4 порту Fast Ethernet або Gigabit Ethernet, з них 3 можуть слугувати для користувача інтерфейсами. Модульна версія містить 4 користувальницьких порту Gigabit Ethernet (UTP і / або SFP), а з'єднання з мережею може здійснюватися або через 2 порт Fast Ethernet або Gigabit Ethernet, або за допомогою будь-якого з наступних модулів.

Цей модуль підтримує до 4 пар SHDSL.bis та пов'язані EFM, а також передає до 22.8 Мбіт/с призначеного для користувача трафіку. Він дозволяє оператору зв'язку розширити охоплення сервісів Ethernet по інфраструктурі DSL, між майданчиком замовника і місцевим пристроєм DSLAM, надаючи послуги замовниками, до яких не дотягується оптоволоконні кабелі.

За допомогою цього модуля пристрій ETX-203A може передавати трафік послуг Carrier Ethernet по 4 або 8 каналах E1 / T1 або 2 каналах T3. Модуль підтримує стандартний функціонал Ethernet over NG-PDH, такий, як протоколи GFP, VCAT і LCAS, тому оператори зв'язку можуть виділяти пропускну здатність своїм замовникам, просто міняючи число каналів, що пов'язані в віртуальних групах не перериваючи при цьому обслуговування. OAM Ethernet на апаратній основі і моніторинг продуктивності для забезпечення SLA

Пристрій ETX-203A підтримує повний набір OAM Ethernet, включаючи Ethernet Link OAM згідно IEEE 802.3-2005 (колишній 802.23ah), Ethernet Service OAM згідно IEEE 802.1ag і моніторинг продуктивності згідно ITU-T Y.1731. Ці можливості доповнені вбудованим механізмом генерації перевірок по шлейфу згідно RFC-2544 і перевірками на Layer 1, Layer 2 і Layer 3 з обміном MAC і IP-адресами вихідного і кінцевого пунктів.

Завдяки над швидким апаратних можливостей обробки, ETX-203A виконує вимірювання продуктивності (PM) і OAM в нано секундах з максимальною точністю. Пристрій ETX-203A пропонує передові засоби для забезпечення SLA, включаючи визначається користувачем порогової конфігурації ключових показників ефективності (KPI), попередження в реальному часі про порушення SLA і щоденну статистичну звітність для кожного потоку.

Пристрій ETX-203A оснащений розвиненими програмними засобами, які дозволяють для кожного потоку обробити трафік з різними пріоритетами. Він підтримує класифікацію трафіку згідно з численними критеріями. Крім того, функції вимірювання, формування і згладжування дозволяють операторам обмежувати трафік по швидкості згідно з заздалегідь заданим профілем гарантованої швидкості (CIR) та крайньої швидкості (EIR). Покращена якість обслуговування, та забезпечується ієрархічний механізм пріоритетної передачі, в якому черга займається згідно безумовному пріоритету (SP) і чесна черга з ваговими коефіцієнтами (WFQ), а також політика WRED для інтелектуального управління чергами та маркування кадрів Ethernet P-bit з використанням «кольоровості».

Пристрій ETX-203A підтримує подвійне резервування підключення абонента до мережі і агрегацію каналу (LAG) за допомогою LCAP згідно 802.3ad. Більш того, це обладнання підтримує захисне перемикання лінії Ethernet (ELPS) згідно ITU-T G.8031.

Пристрій ETX-203A включає функціонал передачі тактової частоти, необхідний в додатках з синхронізацією трафіку в пакетних мережах, як

наприклад, при транспорті стільникового трафіку. Підтримується Синхронний Ethernet (Sync-E) для відновлення і розподілу тактової частоти, а також прозора передача синхронних імпульсів (TC) згідно 1588v2.

Пристрій ETX-203A має можливості керування, включаючи локальне через ASCII термінал (RS-232). Віддалене управління може здійснюватися через внутрішні смуги, тобто через мережевий або призначений для користувача порт, або зовнішні смуги – через виділений порт керування, причому трафік управління і призначений для користувача трафік розділені на різні ВЛБС. Розвинені засоби FCAPS і діагностики надаються операторським додатком управління елементами мережі RADview-EMS через зручний призначений для користувача інтерфейс на основі SNMP. ETX-203A підтримує різні типи доступу: CLI через Telnet, SNMP і TFTP. Засоби захисту включають SSH і SSL, SNMPv3, RADIUS і TACACS +, а також список контролю доступу до управління (ACL).

ETX-203A має компактні розміри і ідеально підходить для установки в обмеженому просторі [2].

1.1.2 EtherNID

Сімейство вимірювальних зондів Metrotek EtherNID (рис. 1.2) являє собою пристрої демаркації для мереж Ethernet. Дані пристрої виявляються цікавими для вирішення завдання демаркації (поділу) відповідальності в точках «оператор-оператор», «клієнт-оператор», а також «департамент-департамент», «сегмент-сегмент», «виробник-виробник» і т. д. Пристрої застосовуються скрізь, де є демаркаційна точка з інтерфейсом Ethernet. Оскільки цей інтерфейс має найширше розповсюдження, прилади можуть бути цікаві операторам дротових і бездротових систем зв'язку, магістральними та місцевими операторам, операторам корпоративних і державних мереж зв'язку, Інтернет-провайдерам і т. д.

Особливо ефективне використання зондів EtherNID для експлуатації

рішень прямого Ethernet доступу до базових станцій мереж 3G/4G/LTE/WiMAX (завдання wireless backhaul) або для моніторингу якості послуг широкосмугового доступу (business services applications).

Забезпечуючи стандартний інтерфейс Ethernet, зонд EtherNID може використовуватися не тільки як пристрій демаркації, але й як медіа-конвертер з функціональністю моніторингу якості.

Основними характеристиками пристроїв сімейства є:

- вимірювання базових показників якості каналу Ethernet;
- вимірювання смуги пропускання каналу без відключення клієнта;
- вимірювання обсягів інформації, що передається;
- вимірювання по rfc-2544;
- реалізація функції інтелектуального шлейфу (il);
- фільтрація трафіку;
- аналіз каналу на відповідність вимогам угоди про якість обслуговування SLA.

Зонди EtherNID, які представлені на рисунку 1.2, можуть мати різні варіанти виконання. З точки зору інтерфейсів виділяються три типи зондів:

- зонди з електричними портами;
- зонди з електричними і оптичними портами;
- зонди тільки з оптичними портами.



Рисунок 1.2 – Сімейство вимірювальних зондів Metrotek EtherNID

Існує чотири варіанти застосування зондів і розміщення їх на вузлах

зв'язку. За своїм застосуванням зонд EtherNID повинен включатися в точку демаркації. Зазвичай така точка знаходиться недалеко від лінійного обладнання. Таким чином, принципи розміщення зондів поруч зі стійкою обладнання є важливою складовою проєкту.

Для проведення вимірювань показників якості зонди можуть підключатися в розрив з'єднання (рекомендоване підключення), або на дзеркальні порти або в точки моніторингу (можливе безпечне з'єднання).

Особливе значення має безпеку і безвідмовність роботи зондів NID, яка мається на увазі:

- наявність системи «bypass»: в разі відключення живлення безперешкодне пропускання трафіку через електричні порти;
- можливість резервування харчування 200в / 48в;
- середній час напрацювання на відмову складає від 52 до 66 років.

Зонди підтримують два великі класи функцій, об'єднаних єдиною моделлю контролю якості в точках демаркації (ESAP):

- функції контролю рівня якості послуг (service assurance);
- функції забезпечення рівня якості послуг (service creation).

Методика підмішування тестового трафіку PAA дозволяє досягати високих показників точності вимірювання параметрів затримки, пакетного джиттера, втрати пакетів. У PAA включена методика вимірювання параметрів в одному напрямку. Методика забезпечує точність вимірювання показника затримки до 1 мкс.

У зондах EtherNID реалізована методика контролю мультисервісного SLA, що передбачає мульти-канальність, мульти-сервісність, мульти-нормативи, мульти-точність і забезпечення вимірювання SLA в режимі реального часу в будь-якій точці мережі.

Зонди EtherNID забезпечують генерацію службового трафіку до швидкості фізичного порту і вимір пропускної здатності без перерви передачі клієнтського трафіку сервісів. Зонди дозволяють поділити послуги та під канали E-Line и E-LAN без необхідності реконфігурації комутаторів Ethernet.

Створення карти сервісів дозволяє створювати теги PUSH для окремого VLAN і встановлювати дані по бітам P (на основі DSCP або VLAN). Карта сервісів дозволяє поділити потоки даних по VLAN ID або PCP тегам (надання пріоритетів в полях C-VLAN PCP, IP DSCP або IP). Потім можна застосувати різні правила поділу смуги, реалізувавши політику управління трафіком і безпосередньо зв'язати її з політикою управління SLA.

Зонд забезпечує формування шлейфів рівня 1-2-3-4 без відключення клієнтського трафіку. При формуванні шлейфу можливо включити функції фільтрації пакетів по VLAN, Service-Level, адресами MAC/IP. Функція автоматичного формування шлейфу дозволяє автоматизувати будь-які вимірювання показників якості з використанням зондів або приладів сторонніх виробників.

Зонди забезпечують збір і відображення в режимі реального часу статистики за показниками якості рівня 1-2-3-4 (по VLAN, Ethertype, ToS, CoS, MAC, IP, і т. д.).

Зонди EtherNID підтримують Jumbo Frames до 10 240 байтів на всіх підтримуваних швидкостях (10/100/1000 Мбіт/с).

Зонди дозволяють реалізувати політику керування трафіком за рахунок встановлення будь-яких порогів з швидкості лінії вгору і вниз за показникам CIR/EIR, а також фільтрацію за всіма видами трафіку.

Зонди забезпечують функції фільтрації на рівнях 1-2-3-4 (L2CP, BDPU, по VLAN, Ethertype, типу протоколу, MAC, IP, за правилами, які задаються користувачем).

Для збору інформації зонди EtherNID підтримують протоколи SNMP v1 і v2c. Ця функція широко застосовується для інтеграції зондів в систему wiSLA або будь-які зовнішні системи моніторингу.

Для забезпечення процедури аутентифікації в зондах реалізована підтримка протоколу RADIUS.

Зонд забезпечує одиночну або подвійну точку моніторингу та відведення трафіку. Відведення трафіку може виконуватися із застосуванням

інтелектуального стекового фільтра, що полегшує подальший аналіз трафіку.

Зонди забезпечують внесення і видалення тегів VLAN з потоку, що транслюється через зонд, що дозволяє реалізувати в мережі політику надання тегів VLAN і стекового VLAN (Q-in-Q) [3].

1.2 CISCO ip sla

Програмне забезпечення Cisco IOS (Internetwork Operating System) Software – це операційна система, яка забезпечує функціонування мережного обладнання Cisco, що є основою мережі Інтернет і найбільших приватних мереж. Підтримуючи широкий спектр обладнання Cisco, операційна система Cisco IOS забезпечує програмну платформу загальним набіром функцій і інтерфейсом командного рядка в рамках всієї мережевої інфраструктури.

Cisco IOS реалізує широку функціональність в різних областях мережеских технологій.

Деякі основні особливості Cisco IOS:

- підтримка широкого спектру мережеских протоколів;
- інтеграція даних, голосу і відео в рамках єдиної IP мережі;
- механізми забезпечення якості обслуговування (QoS);
- засоби забезпечення безпеки;
- підтримка протоколу IPv6;
- підтримка мобільності користувачів;
- підтримка багатоадресної розсилки (IP multicast);
- засоби управління.

Підтримка широкого спектру мережеских протоколів Cisco IOS завжди відрізнялася підтримкою численних мережеских протоколів. Серед них – IP версії 4, IP версії 6, IPX, AppleTalk, OSI, SNA, DECnet, VINES, XNS і ін. В Cisco IOS реалізований найширший в галузі спектр протоколів маршрутизації для корпоративних і операторських мереж. Серед них – OSPF, IS-IS, EIGRP, BGP, протоколи маршрутизації багатоадресного трафіку та інші протоколи.

Основною метою впровадження мережевих технологій є забезпечення поділу ресурсів між користувачами і додатками для підвищення продуктивності, що дозволяє отримати конкурентні переваги в бізнесі. Механізми забезпечення якості обслуговування (QoS), реалізовані в операційній системі Cisco IOS, гарантують передбачуване та кероване обслуговування широкого спектру додатків, забезпечуючи при цьому максимальну ефективність роботи мережі.

QoS забезпечує пріоритетну роботу важливих додатків, одночасно розділяючи мережеві ресурси з менш важливими додатками. QoS гарантує необхідну смугу пропускання і мінімальні затримки, необхідні мультимедійними і голосовими додатками. Механізми забезпечення якості обслуговування є фундаментом для реалізації прогресивних технологій, таких як інтеграція голосу і даних, відеоконференції і майбутні додатки в локальних, територіально-розподілених і глобальних мережах.

Cisco IOS відрізняється повнотою реалізації механізмів якості обслуговування. Серед них:

- засоби класифікації трафіку – по заголовкам канального і мережевого рівнів, а також функціональність Network-Based Application Recognition (NBAR) для розпізнавання додатків, QoS Policy Propagation Through BGP;
- засоби організації черг і управління перевантаженнями – Priority Queuing (PQ), Custom Queuing (CQ), Weighted Fair Queuing (WFQ), Class-Based Weighted Fair Queuing (CBWFQ), Low Latency Queuing (LLQ), Weighted Random Early Detection (WRED);
- засоби обмеження і згладжування трафіку – Committed Access Rate (CAR), Generic Traffic Shaping (GTS), Frame Relay Traffic Shaping (FRTS);
- механізми підвищення ефективності використання каналів зв'язку – Compressed Real Time Protocol (CRTTP), Link Fragmentation and Interleaving (LFI), стискання даних.

Процес управління мережею вимагає наявності відповідних інструментів і можливостей. Cisco IOS включає в себе функціональність,

необхідну для ефективного управління мережею, в тому числі:

Cisco Service Assurance Agent (SAA) – вбудована функціональність IOS, яка проводить моніторинг продуктивності мережі шляхом вимірювання ключових параметрів SLA (Service-Level Agreement), таких як затримка передачі пакетів, варіація затримки, втрати пакетів, пропускна здатність, доступність і продуктивність служб прикладного рівня, а також інших параметрів. SAA проводить моніторинг продуктивності мережі між маршрутизатором Cisco і віддаленим пристроєм, в якості якого може виступати інший маршрутизатор Cisco або будь-який хост IP. При виявленні відхилення вимірюваних параметрів від заданих значень SAA може відправляти відповідні повідомлення по протоколу SNMP:

- інструменти віддаленого налаштування, налаштування та моніторингу: інтерфейс командного рядка Cisco IOS забезпечує доступ до всіх інструментів управління; підтримка протоколів Telnet, SSH, HTTP і HTTPS; зручні вбудовані інструменти віддаленого web-управління SDM, CRWS, CMS; підтримка протоколу SNMP версій 1, 2 і 3; підтримка функціональності RMON; підтримка протоколу NTP (Network Time Protocol);

- технологія NetFlow, розроблена компанією Cisco Systems, дозволяє операторам зв'язку і корпоративним замовникам здійснювати облік трафіку, проводити оперативний моніторинг мережі, моніторинг і профілювання користувачів і додатків, планування розвитку мережі та вирішувати інші завдання. За допомогою NetFlow можна отримати вичерпну інформацію про використання мережевих ресурсів, зокрема, про кількість переданих пакетів і байт, інформацію з заголовків мережевого і транспортного рівнів, про час початку і завершення передачі потоку і іншу інформацію. Cisco розробила ряд додатків для збору, фільтрації та агрегації даних NetFlow.

Партнери Cisco доповнюють рішення додатками обробки та відображення даних. Продукти, розроблені Cisco і її партнерами, дозволяють запропонувати замовникам готове рішення, зокрема, білінгову систему, засоби планування розвитку мережі і т. д.

Суть технології IP SLA в Cisco IOS заснована на вбудованих програмних тестах стану мережі. На поточний момент існує більше десятка тестів. Різні IOS підтримують різні набори тестів. Базовими тестами є:

- dhcp – DHCP Operation;
- dns – DNS Query Operation;
- echo – Echo Operation;
- frame-relay – Perform frame relay operation;
- ftp – FTP Operation;
- http – HTTP Operation;
- jitter – Jitter Operation;
- pathEcho – Path Discovered Echo Operation;
- pathJitter – Path Discovered Jitter Operation;
- slm – SLM Operation;
- tcpConnect – TCP Connect Operation;
- udpEcho – UDP Echo Operation;
- voip – Voice Over IP measurement.

При різних способах конфігурації Cisco дозволяє:

- виконати тест та вивести результати тесту у командному рядку;
- результати тестів зберігаються і доступ до них можна отримати по SNMP;
- налаштувати відправку SNMP trap і syslog повідомлень, які інформують про незадовільних і задовільних результатах тестів;
- конфігурувати Cisco, так щоб він приймав рішення про маршрутизації на основі результатів тестів [4].

1.3 Порівняльний аналіз засобів вимірювання якості характеристик передавання даних

На підставі проведених досліджень поточного ІТ ринку вимірювальних пристроїв були обрані основні параметри, що дозволяють успішно проводити

аналіз якості наданих мережевих послуг на лініях зв'язку. Порівняльні результати необхідних параметрів представлені в таблиці 1.1.

Таблиця 1.1 – Порівняння характеристик пристроїв

Критерій порівняння	Пристрої		
	EtherNID	Cisco IP Monitor	Carrier ETX-203A
Наявність CLI доступу	+	+	+
Наявність SSH доступу	+	-	-
Performance assurance тест	+	+	+
DNS тест	-	-	-
RFC2554 тест	+	+	+
Максимальна кількість підтримуваних тестів (одного виду)	50	40	50
Загальна кількість одночасно виконуваних тестів	50	40	50
Кількість планованих RFC тестів	1	1	1
Можливість модифікації	неможливо	неможливо	неможливо
Критичні межі вимірювань односторонніх показників (PL – Packet Loss; RTT - Round Trip Time)	до 20% PL; до 300мс RTT	до 30% PL; до 400мс RTT	до 30% PL; до 400мс RTT
Ціна за одиницю (\$)	2000	2500	2300

На відміну від звичайних комутаторів демаркаційні пристрої мають широкі можливості по управлінню і моніторингу. Найважливішою їх особливістю є вбудовані засоби тестування каналів зв'язку. Підтримка IEEE 802.3ah [5], IEEE 802.1ag, ITU-T Y.1731 і вбудований RFC2544 тестер забезпечує безпрецедентні функції, включаючи оперативний контроль, пошук і усунення несправностей і локалізацію несправностей в мережах. Також дозволяє вести контроль і обмежувати смугу пропускання по окремим VLAN.

Представлені вище демаркаційні пристрої характеризуються великим вимірювальним функціоналом, який часто надмірний і викликає труднощі в конфігурації, а присутні недоліки значно знижують наявні переваги.

Недоліки, знайдені в вимірювальних приладах, представлені у вигляді:

- відсутності підтримки нових типів тестів, таких як DNS;
- обмеження максимально підтримуваних кількостей тестів;
- обмеження підтримуваних одночасно виконуваних тестів;
- обмеження у виконанні планових RFC тестів;
- обмеження в кількості запланованих RFC тестів;
- відсутності можливості модифікації вимірювального ПЗ;
- вартості за одиницю пристрої.

У зв'язку з цим існує гостра необхідність розробити власний програмно-апаратний аналог демаркаційного пристрою, який забезпечить якісні виміри станів ліній зв'язку та інших характеристик згідно RFC2544. Також пристрій має бути конкурентоспроможним на ринку телеметрії і мати привабливу для споживача ціну.

Виходячи з перерахованого вище, були виявлені основні можливості, якими повинна володіти розробляється система, що забезпечує вимірювання якості IP з'єднань.

- можливість одночасного проведення великої кількості (до 100) вимірювальних тестів;
- наявність великої кількості вбудованих типів вимірювальних тестів (udp, dns, rfc2554, icmp);
- управління SLA послуги IP VPN L2 / L3, контроль якості доступу в Інтернет, моніторинг програм L5-L7;
- Plug & Go автоматичне визначення в системі, завантаження сценаріїв з системи і виконання сценаріїв тестування;
- захист алгоритмів від перешкод під час роботи на каналах, що мають високий ступінь навантаження;
- гнучка система налаштувань;
- універсальний протокол для взаємодії із зовнішніми системами;
- різні системи оновлення;
- ціна однієї вимірювальної одиниці не повинна перевищувати 100 \$.

1.4 Висновки за розділом 1

Проаналізовано засоби вимірювання якості характеристик передавання даних. На підставі проведених досліджень поточного ІТ ринку вимірювальних пристроїв були обрані основні параметри, що дозволяють успішно проводити аналіз якості наданих мережеских послуг на лініях зв'язку.

Визначено, що існуючі засоби характеризуються великим вимірювальним функціоналом, який часто надмірний і викликає труднощі у використанні. Крім того, вартість таких засобів, як правило, є досить великою.

Обґрунтовано доцільність розробки власного програмно-апаратного засоби, який дозволить вимірювати якість характеристик передавання даних.

2 ВИБІР ТА ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ, ЯКА ПРОЄКТУЄТЬСЯ

2.1 Аналіз та вибір апаратної платформи для реалізації та використання розроблюваного програмного забезпечення

Проаналізуємо відомі апаратні засоби SheevaPlug, Raspberry Pi та МК802.

В рамках нової ініціативи Plug Computing, яка передбачає створення високопродуктивних і при цьому дуже економних за показниками енергоспоживання ПК, Marvell представила мініатюрний комп'ютер SheevaPlug (рис. 2.1).



Рисунок 2.1 – Мініатюрний комп'ютер SheevaPlug

Особливо цінною новинка може виявитися, працюючи в якості домашнього сервера – завдяки споживаній потужності всього 5 Вт SheevaPlug можна залишати увімкненим цілодобово, а 1,2-ГГц процесора йому цілком вистачає для виконання функцій простого серверного ПК, таких як спільне використання файлів, сервер друку, резервне копіювання даних (рис. 2.2).

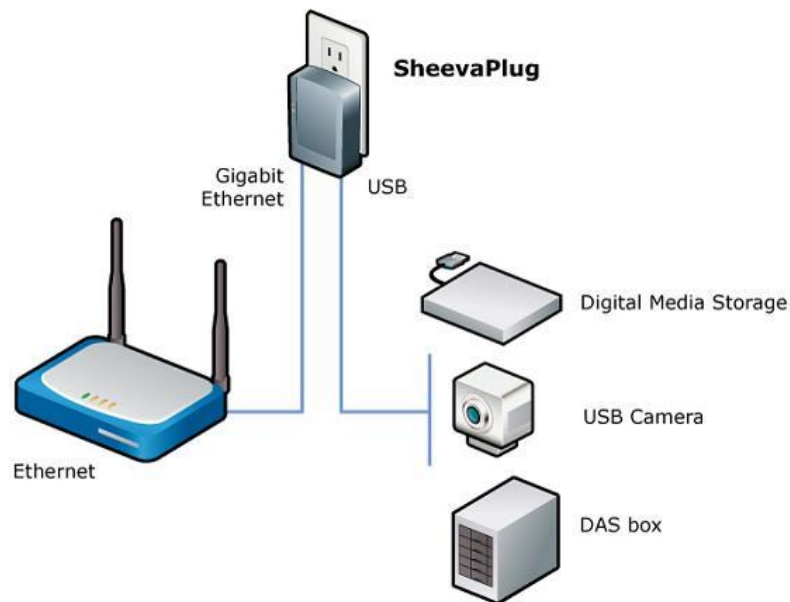


Рисунок 2.2 – Схема використання

За зовнішнім виглядом новинку легко сплутати із звичайним адаптером живлення. Незважаючи на це, SheevaPlug має по 512 Мб оперативної і флеш-пам'яті, і оснащений досить потужним 1,2-ГГц SoC-чіпом Marvell Kirkwood на базі процесора Sheeva. Для підключення до домашньої мережі передбачений інтерфейс Gigabit Ethernet, а периферію можна приєднувати через USB-порт. В якості операційної системи пропонується один з дистрибутивів на ядрі Linux 2.6, в тому числі Fedora, Ubuntu, Debian, Gentoo. Також в планах включення підтримки віртуальної Java-машини і стека OSGI (рис. 2.3).

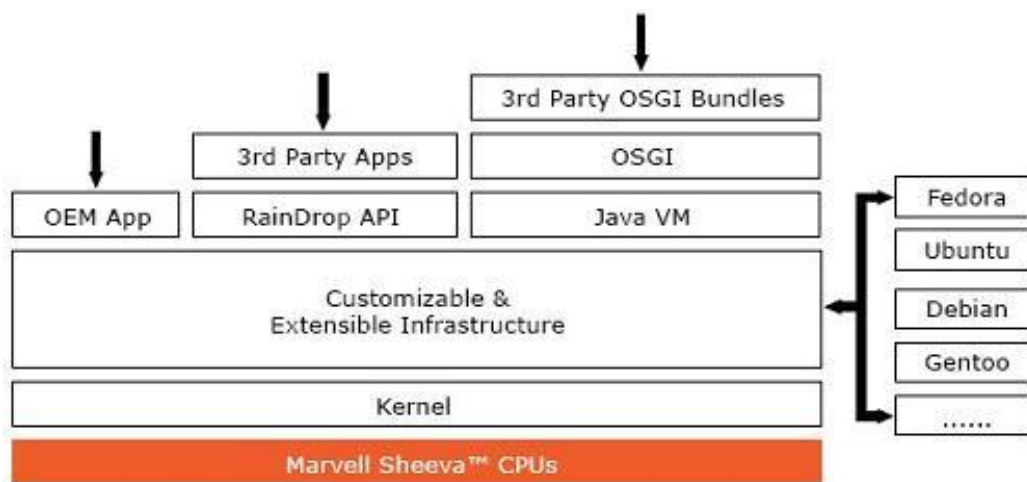


Рисунок 2.3 – Архітектура розробки програмного забезпечення

Хотілося б відзначити, що SheevaPlug, в першу чергу, це набір для розробників і орієнтований на OEM-виробників. У комплект з продуктом, окрім апаратної частини, також входять програмне забезпечення, діагностичні утиліти, детальна документація. Системна плата підтримує режим налаштування, включаючи пряме з'єднання з ПК за допомогою mini-USB кабелю, інтерфейс JTAG, послідовний інтерфейс (рис. 2.4).

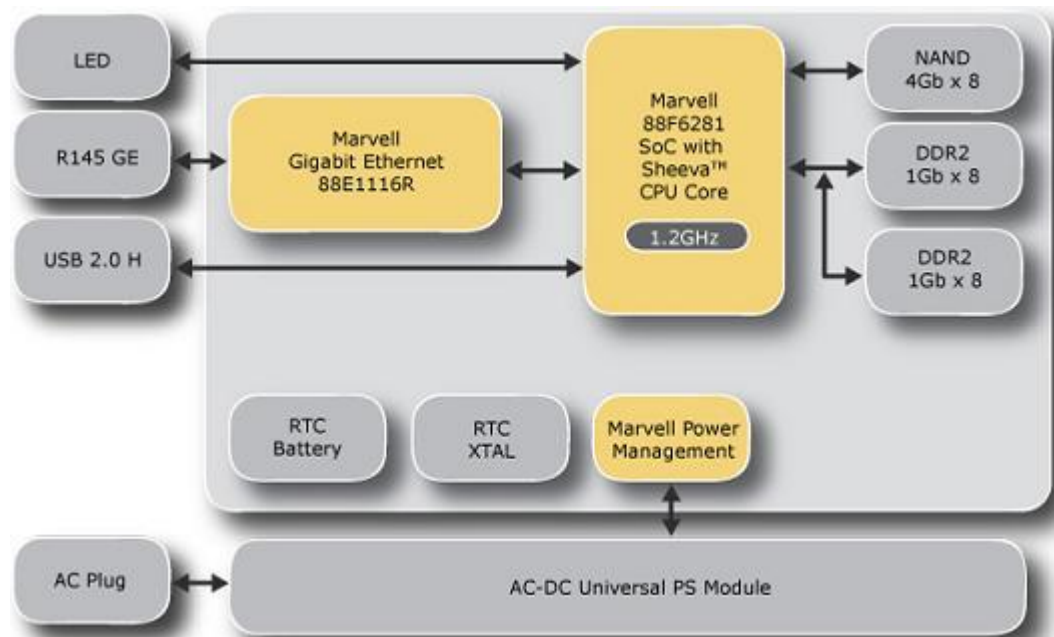


Рисунок 2.4 – SheevaPlug блок-схема

Деякі компанії вже запропонували свої цікаві розробки на базі SheevaPlug. Наприклад, Cloud Engines Pogorplug дозволяє віддалено переглядати файли на зовнішніх накопичувачах через web-інтерфейс, завантажувати мультимедіа файли. Ctera Networks CloudPlug здатне перетворити будь-який USB-драйв в NAS-пристрій і пропонує широкі можливості для резервного копіювання даних. Компанія Axentra перенесла свій додаток для домашніх медіа серверів на платформу SheevaPlug. Остання версія Axentra HipServ 2.0 забезпечує підтримку стандартів UPnP-AV, DLNA, WMC, iTunes.

Набір розробника SheevaPlug доступний вже сьогодні за ціною \$ 100. Крім того, Marvell сподівається, що для великих оптових замовників ціну вдасться знизити до \$ 50 [6].

Raspberry Pi – це мініатюрний, розміром з кредитну карту, не дорогий комп'ютер (рис. 2.5), створений Девідом Брабеном. Raspberry Pi заснований на процесорі з архітектурою ARM11 процесором Broadcom BCM2835 з тактовою частотою 700 МГц і модулями оперативної пам'яті 512 МБ, розміщеними за технологією «package-on-package» безпосередньо на процесорі. Це дозволяє досягти прийнятної продуктивності при низькому енергоспоживанні.

Raspberry Pi (рис. 2.5), володіє вражаючими можливостями – в упаковку розміром з кредитку поміщається повноцінний системний блок, що за характеристиками близький до масових комп'ютерів всього лише п'ятирічної давності. Завдяки низькими вимогами відкритого програмного забезпечення (OpenSource) до апаратної частини, на плату можна встановити операційну систему Linux, а також набір супутнього програмного забезпечення. Наприклад, у варіанті, який передбачається зараз, передбачається встановлення операційної системи Ubuntu Linux, Debian або Fedora, браузеру Iceweasel (клон Mozilla Firefox), офісного пакету Koffice (аналог Word/Excel під Windows) і мови програмування Python. Все це програмне забезпечення є безкоштовним, і не надто вимогливим до ресурсів. Крім основного ядра, BCM2835 включає в себе графічне ядро з підтримкою OpenGL ES 2.0, апаратного прискорення і FullHD-відео і ядро цифрового сигнального процесора. Цікавою особливістю є відсутність годинника реального часу [7].

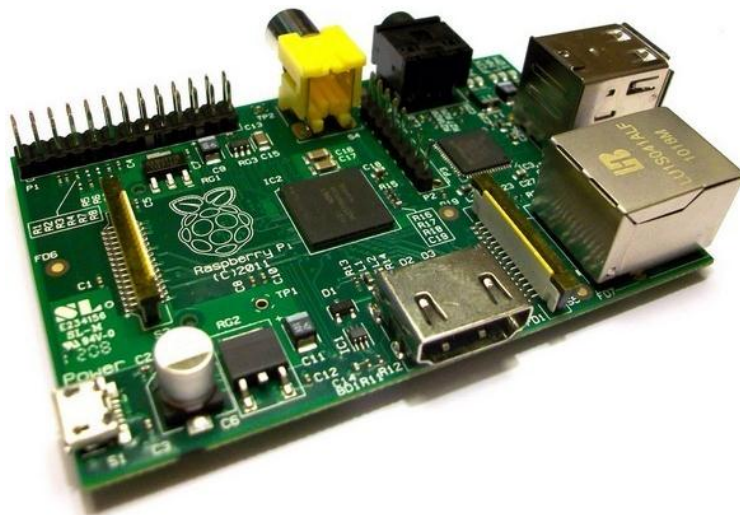


Рисунок 2.5 – Raspberry Pi

МК802: за розмірами комп'ютер (рис. 2.6) трохи більше звичайної флешки. Розміри пристрою – 1,2х8,8х3,5 сантиметри. Важить цей мініатюрний комп'ютер всього 200 грам. Пристрій виконується в двох колірних рішеннях: білому і чорному. Незалежно від забарвлення на ньому зображений логотип Android і напис "Android 4.0 Mini PC". Пристрій вироблено з пластика, що робить його ударостійким, і дозволяє без проблем брати собою, коли це потрібно.

На передній грані пристрою розташований USB-порт, який дає можливість підключати гарнітуру або додаткове обладнання: миша, клавіатура та ін. Поруч з ним знаходиться роз'єм, через який відбувається живлення комп'ютера. На одній зі сторін розташований micro-USB порт. На протилежній від повноцінного USB-порту межі розташовано mini-HDMI роз'єм, через який здійснюється підключення до телевізора або ж монітора. І нарешті, на останній грані розташований роз'єм для micro-SD, що підтримує карти об'ємом до 32 Гб.

Мікро-комп'ютер працює під управлінням операційної системи Android 4.0. Серцем пристрою є Allwinner A10 ARM Cortex-A8 1.5GHz – процесор, який відповідає за швидкодію всіх програм, що запускаються на пристрої. Процесор не можна назвати потужним, а сам комп'ютер за своїми технічними характеристиками приблизно працює, як смартфон середнього класу. Для запуску вимогливих програм, у девайса є 512 Мб DDR3 оперативної пам'яті. Цей показник не можна назвати високим, але для запуску будь-яких програм, що розповсюджуються через Google Play, його вистачить. Для зберігання встановлених програм, налаштувань системи і інших мультимедійних файлів у пристрої є 4 Гб флеш-пам'яті. Вона розширюється за рахунок вищезгаданого роз'єму для micro-SD і карти пам'яті.

Важливим є те, що Android 4.0 Mini PC МК802 готовий працювати прямо з коробки, оскільки не має потреби щось встановлювати, чекати, вибирати ім'я користувача та інше. Також немає на пристрої і кнопок вмикання / вимикання. Щоб розпочати роботу з пристроєм, необхідно підключити його до монітора або ТБ, використовуючи роз'єм HDMI, і підключити USB для

зарядки до розетки, або знову до телевізора. Якщо в ТВ немає роз'єму HDMI, то можна скористатись перехідником HDMI-DVI. Цей роз'єм більш популярний на старих моделях моніторів. Але, в разі використання перехідника якість зображення погіршиться.

Важливо згадати, що в пристрої є вбудований Wi-Fi модуль, що дозволить використовувати контролери, які можна підключати за допомогою бездротового зв'язку. У майбутніх версіях розробники обіцяють наявність Bluetooth-інтерфейсу, що дозволить підключати додаткові контролери, не займаючи при цьому нечисленні роз'єми.

Як згадувалося раніше, в пристрої MK802 (рис. 2.6) немає кнопки живлення, тому після підключення до дисплея пристрій автоматично почне завантаження. В середньому, завантаження триває трохи більше хвилини. Досить непоганий варіант, як для девайса, що наближається до рівня ПК. Окремої кнопки для вимкнення або переходу в режим сну, також немає, правда за допомогою додаткових утиліт, є можливість перезавантажити пристрій або ж вимкнути живлення.



Рисунок 2.6 – Mini PC MK802

На пристрої встановлена «чиста» версія Android 4.0 Ice Cream Sandwich. Вона ніяк не обмежена, тому користувач отримує той же функціонал, який він мав би, використовуючи мобільний пристрій з даною ОС на борту [8].

Порівняльну характеристику міні комп'ютерів наведено у таблиці 2.1.

Таблиця 2.1 - Порівняльна характеристика міні комп'ютерів

Прилад	Характеристики								
	Операційна система	Тактова частота	Обсяг накопичувача	Обсяг пам'яті	USB	Ethernet	WiFi	Power supply	Ціна(\$)
SheevaPlug	Linux	1.2 GHz	512 MB	512 MB	2	1	–	embedded	50
Raspberry Pi	Linux	700 MHz	External SD card	512 MB	2	1	–	external	78
MK802	Android	1.5GHz	4 GB	512 MB	1	–	1	external	60

Виходячи з представленого в таблиці 2.1 аналізу міні комп'ютерів, використовувана апаратна платформа повинна мати процесор з частотою не нижче 1 ГГц, який дозволить повноцінно працювати операційній системі і розробленому програмно вимірюваному забезпеченню. Вбудований носій інформації з об'ємом від 512Мб дозволяє розмістити файли операційної системи і програмного забезпечення. Об'єм оперативної пам'яті повинен бути не менше 256Мб, в зв'язку з тим, що операційна система половину з цього обсягу буде використовувати для своєї роботи. Інтерфейс USB дозволить підключати зовнішні пристрої, які дозволять в подальшому розширити функціональні можливості розроблюваного комплексу. Один мережевий інтерфейс необхідний для проведення вимірювальних тестів розроблюваної

програми. Блок живлення пристрою повинен бути вбудований, для компактності і зручності використання. Пристрій повинен підтримувати Linux подібні операційні системи. Основний параметр – це вартість одиниці пристрою, яка дозволить здешевити кінцевий програмно-апаратний комплекс і зробити його більш доступним для кінцевого споживача.

Виходячи з перерахованих вище апаратних характеристик, найбільш прийнятним є пристрій SheevaPlug, що має необхідні характеристики і прийнятну ціну, яка може зменшуватися в залежності від обсягу закупуваної партії.

2.2 Аналіз та вибір мови програмування для розробки програмного забезпечення

Проаналізуємо відомі мови програмування, які можуть використовуватися для створення програмного забезпечення.

C++ – компільована статично типізована мова програмування загального призначення.

Підтримує такі парадигми програмування як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування, забезпечує модульність, роздільну компіляцію, обробку винятків, абстракцію даних, оголошення типів (класів) об'єктів, віртуальні функції. Стандартна бібліотека включає, в тому числі, загальноновживані контейнери і алгоритми. C++ поєднує властивості як мов високого рівня, так і низького рівня. У порівнянні з його попередником – мовою C найбільшу увагу приділено підтримці об'єктно-орієнтованого і узагальненого програмування.

Назва «C++» походить від назви мови C, в якому унарний оператор ++ позначає інкремент змінної.

Будучи однією з найпопулярніших мов програмування, C++ широко використовується для розробки програмного забезпечення. Область його застосування включає створення операційних систем, різноманітних

прикладних програм, драйверів пристроїв, додатків для вбудованих систем, високопродуктивних серверів, а також розважальних програм (ігор). Існує безліч реалізацій мови C ++, як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C ++, Intel C ++ Compiler, Embarcadero (Borland) C ++ Builder і інші. C ++ зробив величезний вплив на інші мови програмування, в першу чергу на Java і C #.

Синтаксис C ++ успадкований від мови C. Одним з принципів розробки було збереження сумісності з C. Проте, C ++ не є в суворому сенсі надмножиною C. Існує безліч програм, які можуть однаково успішно транслюватися як компіляторами C, так і компіляторами C ++, досить велике, але не включає всі можливі програми на C [9].

Bash (від англ. Bourne again shell) – вдосконалена і модернізована варіація командної оболонки Bourne shell. Одна з найбільш популярних сучасних різновидів командної оболонки UNIX. Особливо популярна в середовищі Linux, де вона часто використовується в якості попередньо встановленою командної оболонки.

Bash – це командний процесор, що працює, як правило, в інтерактивному режимі в текстовому вікні. Bash також може читати команди з файлу, який називається скриптом (або сценарієм). Як і всі Unix-оболонки, він підтримує автодоповнення імен файлів і папок, підстановку виведення результату команд, змінні, контроль за порядком виконання, оператори розгалуження і циклу. Ключові слова, синтаксис і інші основні особливості мови були запозичені з sh. Інші функції, наприклад, історія, були скопійовані з csh і ksh. Bash в основному задовольняє стандарту POSIX, але з рядом розширень.

Назва «bash» є акронімом від Bourne-again-shell («ще-одна-командна-оболонка-Борна») і являє собою гру слів: Bourne-shell – одна з популярних різновидів командної оболонки для UNIX (sh), автором якої є Стівен Борн (1978), вдосконалена в 1987 році Брайаном Фоксом. Прізвище Bourne (Борн)

перегукується з англійським словом *born*, що означає «народжений», звідси: ново-народжена командна оболонка [10].

Java – об'єктно-орієнтована мова програмування, розроблений компанією Sun Microsystems (в подальшому придбана компанією Oracle). Програми Java зазвичай компілюються в спеціальний байт-код, тому вони можуть працювати на будь-якій віртуальній Java-машині (JVM) незалежно від комп'ютерної архітектури. Дата офіційного випуску – 23 травня 1995 року.

Програми на Java транслюються в байт-код, що виконується віртуальною машиною Java (JVM) – програмою, що оброблює байтовий код і предає інструкції обладнанню як інтерпретатор.

Перевагою подібного способу виконання програм є повна незалежність байт-коду від операційної системи і устаткування, що дозволяє використовувати Java-додатки на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки завдяки тому, що виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером) викликають негайне переривання.

Часто до недоліків концепції віртуальної машини відносять те, що виконання байт-коду віртуальною машиною може знижувати продуктивність програм і алгоритмів, реалізованих на мові Java. Останнім часом було додано ряд удосконалень, які дещо збільшили швидкість виконання програм на Java:

- застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді;

- широке використання платформно-орієнтованого коду (native-код) в стандартних бібліотеках;

– апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, що підтримується деякими процесорними фірми ARM).

Ідеї, закладені в концепцію і різні реалізації середовища віртуальної машини Java, надихнули безліч ентузіастів на розширення переліку мов, які могли б бути використані для створення програм, що виконуються на віртуальній машині. Ці ідеї знайшли також місце в специфікації загально мовної інфраструктури CLI, що закладена в основу платформи .NET компанією Microsoft [11].

Perl – мова високого рівня, що інтерпретується як динамічна мова програмування загального призначення, яка була спочатку створена для маніпуляцій з текстом, але на даний момент використовується для виконання широкого спектру завдань, включаючи системне адміністрування, веб-розробку, мережеве програмування, ігри, біо-інформатику, розробку графічних користувацьких інтерфейсів.

Мову можна охарактеризувати скоріше як практичну (легкість у використанні, ефективність, повнота), ніж як гарну (елегантність, мінімалістичний). Головними перевагами мови є підтримка різних парадигм (процедурний, об'єктно-орієнтований і функціональний стилі програмування), контроль пам'яті (без збирача сміття, заснованого на циклах), вбудована підтримка обробки тексту, а також велика колекція модулів сторонніх розробників.

Загальна структура Perl в загальних рисах веде свій початок від мови Cі. Perl – процедурна за своєю природою, має змінні, вирази присвоювання, блоки коду, що відокремлюються фігурними дужками, керуючі структури і функції.

Perl також запозичує ряд властивостей з мов програмування командних оболонок UNIX. Всі змінні маркуються провідними знаками, які точно виражають тип даних змінної в цьому контексті (наприклад, скаляр, масив, хеш). Важливо, що ці знаки дозволяють змінним бути інтерпольованим в рядках. Perl має безліч вбудованих функцій, які забезпечують інструментарій,

часто використовуваний для програмування оболонки, наприклад сортування або виклик системних служб.

Perl запозичила масиви з Lisp, регулярні вирази з AWK і sed, з AWK також запозичені хеші («асоціативні масиви»). Регулярні вирази полегшують виконання багатьох завдань по парсингу, обробці тексту та маніпуляцій з даними.

Всі версії Perl виконують автоматичну типізацію даних і автоматичний контроль над пам'яттю. Інтерпретатор знає тип і запити пам'яті кожного об'єкта програми, він розподіляє і звільняє пам'ять, виробляючи підрахунок посилань. Зміна одного типу даних в інший – наприклад, числа в рядок – відбувається автоматично під час виконання, неможливі для виконання зміни типів даних призводять до фатальної помилки [12].

У таблиці 2.2 представлено результати порівняльного аналізу різних мов програмування, що можуть використовуватися для розробки програмної частини [13].

Таблиця 2.2 – Порівняння мов програмування

Можливість	Мова			
	C++	Java	Perl	Bash
Імперативна	+	+	+	—
Об'єктно-орієнтована	+	+	+	—
Функціональна	+/-	-/+	+	—
Статична типізація	+	+	+/-	—
Динамічна типізація	—	—	+	+
Явна типізація	+	+	-/+	—
Неявна типізація	-/+	—	+	+
Open-source компілятор (інтерпретатор)	+	+	+	+
Можливість компіляції	+	+	+	—
Створення об'єктів на стеку	+	—	—	—
Ручне управління пам'яттю	+	—	—	—
Прибирання сміття	-/+	+	+	+
Інтерфейси	+	+	+/-	—
Макроси	+	—	+	—
Шаблони / generics	+	+	—	—
Перевантаження функцій	+	+	—	+

Bash є інтерпретованою мовою програмування і її основне призначення написання системних скриптів. Вона не має основних властивостей, що властиві основним мовам програмування – класи, структури даних, створення об'єктів, типізація, тому її не можна вибрати основною мовою програмування для проєкту.

Perl є інтерпретованою, динамічною мовою програмування, основною сферою якої є робота з текстом. Так як вона є інтерпретованою, то має досить великий час на виклик системної функції, а так як наш додаток буде викликати велику кількість системних функцій, то дані затримки в часі не припустимі.

Java є інтерпретованою, статичною мовою програмування. Має вбудований збирач сміття, тому неможливо прогнозувати точні часові інтервали. Також Java потрібні високі мінімальні ресурси, якими обрана платформа не володіє.

В результаті проведеного аналізу мов програмування була вибрана мова C ++. Основною її перевагою є швидкість виконання програми, це пов'язано з тим, що обрана апаратна платформа має досить низькі обчислювальні здібності, що зумовлені використанням ARM процесора і застосування різних мов з підтримкою збірки сміття не є доцільним. Розроблюване програмне забезпечення буде здійснювати велику кількість системних викликів, тому зайві системи FFI, будуть приводити до втрати продуктивності. C ++ компілюється в машинні коди і має максимальну швидкість роботи при виклику системних функцій. Також в C ++ відсутнє збирання сміття, що знижує швидкість роботи і унеможливорює прогнозування виконання тимчасових інтервалів.

2.3 Аналіз фреймворків для розробки програмного забезпечення

Проаналізуємо відомі фреймворки, які можуть використовуватися для розробки програмного забезпечення.

Бібліотека BOOST C ++ - це сукупністю великої кількості незалежних бібліотек, створених незалежними розробниками і ретельно перевіреними на

різних платформах. Можна вважати, що BOOST C ++ - це розширення стандартної бібліотеки C ++. Багато речей, які пропонувалися для нового стандарту C ++ , але були відкинуті комітетом по стандартизації, осіли в BOOST. Використання компонентів BOOST C ++ в своїх розробках економить значні зусилля - особливо якщо планується перенесення проєкт на різні платформи [14]. Вільно поширюються за ліцензією Boost Software License разом з вихідним кодом. Проєкт був створений після прийняття стандарту C ++, коли багато хто був незадоволений не включенням в стандарт деяких бібліотек. Проєкт є свого роду «випробувальним полігоном» для різних розширень мови і частин бібліотек, що є кандидатами на включення в наступний стандарт C ++.

Boost має помітну спрямованість на дослідження і розширюваність (мета програмування і узагальнене програмування з активним використанням шаблонів). Деякі програмісти вважають його стандартом де-факто і необхідним доповненням до STL. Інші, навпаки, уникають будь-якого використання бібліотек в проєктах, побоюючись введення зайвої залежності в проєкт і вважаючи, що використання цих бібліотек занадто підвищує вимоги до знань програміста C ++, так як деякі частини Boost є досить складними [14,15].

POCO (або C ++ Portable Components) – це колекція бібліотек класів з відкритим вихідним кодом, що спрощує і прискорює розробку мережових мультиплатформних додатків на C ++. Бібліотеки добре інтегруються з Стандартною бібліотекою шаблонів (STL) і органічно заповнюють безліч функціональних прогалин в ній. Модульна структура і ефективна реалізація робить POCO ідеальним кандидатом для використання з розробці для вбудованих пристроїв (прошивки та інше). Область, в якій C ++ стає все більш і більш популярною, так як підходить як для мов низького рівня (пристрої введення-виведення, обробники переривань та інше), так і для мов високого рівня об'єктно-орієнтованої розробки. Природно, POCO підходить і для розробки корпоративних додатків.

Бібліотеки POCO звільняють розробника від винаходу власного велосипеда і дозволяють йому витрачати час на більш вартісні речі. Наприклад,

розробляти швидше і працювати над конкретними особливостями свого застосування відрізняють його від інших.

POCO створений для мережеских мультиплатформних розробок на C ++, як Cocoa від Apple для Mac OS X, або Ruby on Rails для Web-розробок – потужна і в той же час проста у використанні платформа для побудови ваших програм.

POCO побудований з жорстким дотриманням C ++ стандарту ANSI / ISO включає стандартну бібліотеку. Розробники POCO намагалися створити рівновагу між використанням складних механізмів C ++ і утриманням класів і коду в зрозумілому і простому стані [16].

Qt – крос-платформний інструментарій розробки ПЗ на мові програмування C ++. Дозволяє запускати написане з його допомогою ПО на більшості сучасних операційних системах шляхом простої компіляції програми для кожної ОС без зміни вихідного коду. Включає в себе всі основні класи, які можуть знадобитися при розробці прикладного програмного забезпечення, починаючи від елементів графічного інтерфейсу і закінчуючи класами для роботи з мережею, базами даних і XML. Qt є повністю об'єктно-орієнтованим, легко розширюваним і підтримує техніку компонентного програмування.

Існують версії бібліотеки для Microsoft Windows, систем класу UNIX з графічною підсистемою X11, iOS, Mac OS X, Microsoft Windows CE, QNX, вбудованих Linux-систем і платформи S60. В даний момент розглядається можливість впровадження підтримки Qt в Windows Phone.

До недавнього часу бібліотека Qt також поширювалася ще в одній версії: Qt / Embedded. Тепер ця платформа перейменована в Qtopia Core і поширюється як окремий продукт. Qtopia Core забезпечує базову функціональність для всієї лінійки платформ, призначених для розробки додатків для вбудованих і мобільних пристроїв (КПК, смартфонів і т. П.).

Відмітна особливість Qt від інших бібліотек – використання Meta Object Compiler (МОС) – попередньої системи обробки вихідного коду (тобто, Qt – це бібліотека не для чистого C ++, а для його особливого подання, з якого і

«переводить» МОС для подальшої компіляції будь-яким стандартним C ++ компілятором). МОС дозволяє у багато разів збільшити потужність бібліотек, вводячи такі поняття, як слоти і сигнали. Крім того, це дозволяє зробити код більш лаконічним. Утиліта МОС шукає в заголовних файлах на C ++ опис класів, що містять макрос Q_OBJECT, і створює додатковий вихідний файл на C ++, що містить мета об'єктний код. Також Qt дозволяє створювати власні плагіни і розміщувати їх безпосередньо в панелі візуального редактора [17].

Результати порівняння характеристик фреймворків, пов'язаних з реалізацією компонентів системи надано у таблиці 2.3.

Таблиця 2.3 – Порівняння фреймворків

Характеристики	QT	POCO	BOOST
Шаблон каркасу застосунку	+	+	-
Шаблон синглтону	-	+	-
Розмір бібліотеки (Мб)	4	1,2	залежить від використовуваних бібліотек
Кросплатформні потоки	+	+	+
Кросплатформні сокети	+	+	+
Покращені контейнери	+	+	+
Розумні показники	+	+	+
Робота з xml	+	+	-
Веб-сервер	-	+	-
Пул потоків	+	+	-
Система подій	+	+	+

Boost – є набором не взаємозалежних бібліотек, він є більше орієнтованим на математичні проблеми, а не на системну розробку (відсутній вбудований веб-серверів, парсерів xml, не має каркаса системного застосування). Бібліотеки написані з рясним використанням шаблонів, що дуже ускладнюють розуміння і коригування вихідних кодів.

Qt є надзвичайно потужним фреймворком і орієнтований більше на створення призначених для користувача програм з графічним інтерфейсом.

Велика різносторонність фреймворка призводить до великого розміру компільованої бібліотеки (наприклад, розмір файлу QtCore становить 4 Мб). Орієнтація на графічний інтерфейс внесла деякі обов'язкові механізми в каркас додатка – асинхронна глобальна черга подій, додаткові обробники системних подій, і система сигналів-слотів які привносять зайве обчислювальне навантаження на центральний процесор.

РОСО є невеликим фреймворком, який не має спеціалізації і годиться для створення як прикладного, так і системного програмного забезпечення. Має невеликий розмір і величезну кількість підтримуваних апаратних платформ. Для всіх підтримуваних апаратних платформ реалізовані крос-платформні бібліотеки потоків, сокетів, робота з файлами, завантаження і збереження файлів конфігурації, пул потоків, система подій і нотифікацій, що дозволяють дуже легко перенести написаний додаток, шляхом перекомпільовання його під зазначену платформу. Фреймворк вимагає мінімальних апаратних ресурсів: всього 75 MHz ARM9 і 8 MB RAM.

В результаті аналізу найбільш підходящим фреймворком є Росо, оскільки:

- має маленький розмір;
- споживає трохи апаратних ресурсів;
- підтримується великою кількістю платформ;
- має крос-платформні бібліотеки потоків, сокетів.

2.4 Висновки за розділом 2

Досліджено та обрано апаратну платформу для реалізації та використання розроблюваного програмного забезпечення. Визначено, що використовувана апаратна платформа повинна мати процесор з частотою не нижче 1 ГГц, який дозволить повноцінно працювати операційній системі і розробленому програмно вимірюваному забезпеченню. Виходячи з проаналізованих апаратних характеристик визначено, що найбільш прийнятним

є пристрій SheevaPlug, який має необхідні характеристики і прийнятну ціну, яка може зменшуватися в залежності від обсягу за куповуваної партії

Проаналізовано та обрано мову програмування для розробки програмного забезпечення. В результаті проведеного аналізу мов програмування була вибрана мова C ++. Основною її перевагою є швидкість виконання програми, це пов'язано з тим, що обрана апаратна платформа має досить низькі обчислювальні здібності, що зумовлені використанням ARM процесора, тому застосування різних мов з підтримкою збірки сміття не є доцільним. C ++ компілюється в машинні коди і має максимальну швидкість роботи при виклику системних функцій. Також в C ++ відсутнє збирання сміття, що знижує швидкість роботи і унеможливлює прогнозування виконання тимчасових інтервалів.

Проаналізовано відомі фреймворки, які можуть використовуватися для розробки програмного забезпечення. Як фреймворк для розробки програмного забезпечення образно POCO.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Визначення вимірюваних метрик

Було розроблено гнучке і зручне рішення для задач вимірювання якісних характеристик IP-з'єднань і верхніх сервісів – апаратних вимірювальних зондів з застосуванням програмних компонентів wiProbe.

Зонди wiProbe забезпечують комплекс вимірювань, необхідний для контролю і управління SLA на рівні транспортних мереж Ethernet, стека протоколу TCP / IP, мережевих і верхніх призначених для користувача сервісів (VoIP, Video conferencing, IPTV і ін.), Зокрема вимірювання наступних параметрів:

- коефіцієнт двосторонньої втрати пакетів;
- час двосторонньої затримки пакетів;
- час односторонньої затримки пакетів;
- час двосторонньої варіації затримки пакетів;
- час односторонньої варіації затримки пакетів;
- час дозволу імені вузла;
- пропускна здатність.

Вимірювання зазначених метрик реалізовано з використанням наступних тестів:

U – test (UDP) – коефіцієнт двосторонньої втрати пакетів, час двосторонньої затримки пакетів, час односторонньої затримки пакетів, час двосторонньої варіації затримки пакетів, час односторонньої варіації затримки пакетів.

P – test (ICMP) – коефіцієнт двосторонньої втрати пакетів, час двосторонньої затримки пакетів, час двосторонньої варіації затримки пакетів.

Тест DNS – час дозволу імені вузла.

Тест RFC – коефіцієнт двосторонньої втрати пакетів, час двосторонньої затримки пакетів, пропускна здатність.

Вимірюваними характеристиками передавання даних за допомогою розробленого програмного забезпечення wiProbe є такі:

– коефіцієнт втрати пакетів (Frame Loss Ratio). Втрати пакетів можуть з'являтися за рахунок спотворень, що виникають при їх доставці або за рахунок перевантажень на лініях передачі і мережевих пристроях, що призводять до їх відкидання. Коефіцієнт втрати пакетів відповідно до Технічної специфікації MEF 10.2 [18] – це відношення кількості відправлених, що не дійшли до адресата пакетів до загальної кількості відправлених пакетів, які були передані по мережі в рамках одного тесту. Для обчислення цієї характеристики зонд wiProbe 1 відсилає послідовність тестових пакетів на вхід зонда wiProbe 2. Зонд wiProbe 2 приймає тестові пакети і відсилає їх назад за допомогою заворотчика, встановленого на ньому. Зонд wiProbe 1 отримує заворот тестових пакетів, який дозволяє сформувати файл з даними для відправки на портал wiSLA. Далі виконується порівняння кількості відісланих пакетів з кількістю прийнятих пакетів для визначення коефіцієнта двосторонньої втрати пакетів. Синхронізація зондів виконується по протоколу Network Time Protocol 4 версії (NTPv4), який описується в RFC 5905.

– одностороння затримка (One-Way Delay, OWD) – час, що необхідний для передачі пакета від джерела до приймача, дозволяє визначити наявність асиметрії в механізмах обслуговування при проходженні тестового пакета для кожного напрямку передачі. Для вимірювання необхідно, щоб джерело і приймач були синхронізовані за часом. Практична схема вимірювання часу односторонньої і двосторонньої затримки з використанням зондів wiProbe реалізована наступним чином. Зонд wiProbe 1, який виступає джерелом тестових пакетів, одночасно виступає і в ролі приймача тестових пакетів. Зонд wiProbe 2 виступає в ролі заворотчика тестових пакетів. Зонд wiProbe 1 вставляє тимчасову мітку 1 в кожен тестовий пакет і відсилає послідовність тестових пакетів на зонд wiProbe2. Зонд wiProbe 2 приймає тестові пакети, вставляє в пакети тимчасову мітку 2 і відсилає їх назад на зонд wiProbe 1, за допомогою заворотчика, встановленого на ньому. Зонд wiProbe 1 отримує назад

послідовність тестових пакетів з проставленими тимчасовими мітками 1 і 2 та проставляє тимчасову мітку 3 (для обчислення двосторонньої затримки). На підставі отриманих даних, виконується обчислення часу односторонньої і двосторонньої затримки пакетів;

– час дозволу імені вузла - це час, необхідний для проходження запиту (із зазначенням доменного імені) з зонда wiProbe на сервер DNS і отримання від нього відповіді з зазначенням ір адреси шуканого вузла. Для вимірювання цієї характеристики зонд wiProbe формує і відправляє запит з зазначенням доменного імені шуканого вузла на сервера DNS, який вказується в налаштуваннях зонда;

– варіація затримки (Jitter, IPDV - IP Packet Delay Variation, PDV - packet delay variation). Визначення варіації затримки пакета можливо тільки для пакетів всередині потоку пакетів. Варіація затримки для пари пакетів в потоці визначена для вибраної пари в потоці, що йде від однієї вимірювальної точки до іншої. Варіація затримки – це різниця між односторонніми затримками цих пакетів. Для вимірювання необхідно, щоб джерело і приймач були синхронізовані за часом. Варіація затримки визначається в RFC 3393 як різниця наскрізних затримок проходження двох пакетів. Практична схема вимірювання з використанням зондів wiProbe реалізована наступним чином. Зонд wiProbe 1, який виступає джерелом тестових пакетів, одночасно виступає і в ролі приймача тестових пакетів. Зонд wiProbe 2 виступає в ролі заворотчика тестових пакетів. Зондом wiProbe 1 і зондом wiProbe 2 вставляються тимчасові мітки в тестові пакети, на підставі яких і обчислюється час односторонньої і двосторонньої варіації затримки. Синхронізація зондів виконується по протоколу Network Time Protocol 4 версії (NTPv4), який описується в RFC 5905 [19]. На практиці, при проведенні вимірювань за допомогою зондів wiProbe реалізовано обчислення середнього значення часу односторонньої затримки пакетів. Тест проходить протягом певного часу (за замовчуванням 5 хвилин), виконується посилка 1 пакету в секунду. Після виконання тесту, проводиться

розрахунок середнього часу (average) односторонньої або двосторонньої затримки;

– пропускна здатність – показує співвідношення граничної кількості переданої інформації в одиницю часу через канал зв'язку. Реалізовано вимірювання пропускної здатності каналу зондами wiProbe за допомогою тесту RFC. Є можливість задати такі параметри тесту RFC: напрям тестового трафіку; розмір тестових пакетів; розклад виконання тестів (час і періодичність).

Тести RFC виконуються за допомогою клієнт-серверної програми rfc, яка встановлена на зонди wiProbe і взаємодіє з системною службою зонда wiProbe – slamon. Для вимірювання пропускної здатності каналу, на зонді wiProbe 1 запускається rfc - клієнт, який генерує навантажувальний трафік для максимального завантаження каналу, що тестується. У свою чергу на зонді wiProbe 2 запускається rfc - сервер, який аналізує трафік, що надходить на вхід зонда wiProbe 2 і вивантажує результати тесту через службу slamon на портал wiSLA. Синхронізація зондів виконується по протоколу Network Time Protocol 4 версії (NTPv4), який описується в RFC 5905. wiSLA (well integrated SLA) – платформа автоматизації і управління якістю послуг зв'язку і хмарних сервісів для операторів зв'язку, центрів обробки даних, державного і корпоративного сегментів ринку.

3.2 Модель програмного забезпечення вимірювання характеристик передавання даних

Модель вимірювальної програми дозволяє проводити вимірювання часових характеристик проходження тестових пакетів через лінії зв'язку. Розраховує результат SLA характеристик і на підставі отриманих даних формує звіт, який надсилається на зовнішній сервер wiSLA (рисунок 3.1).

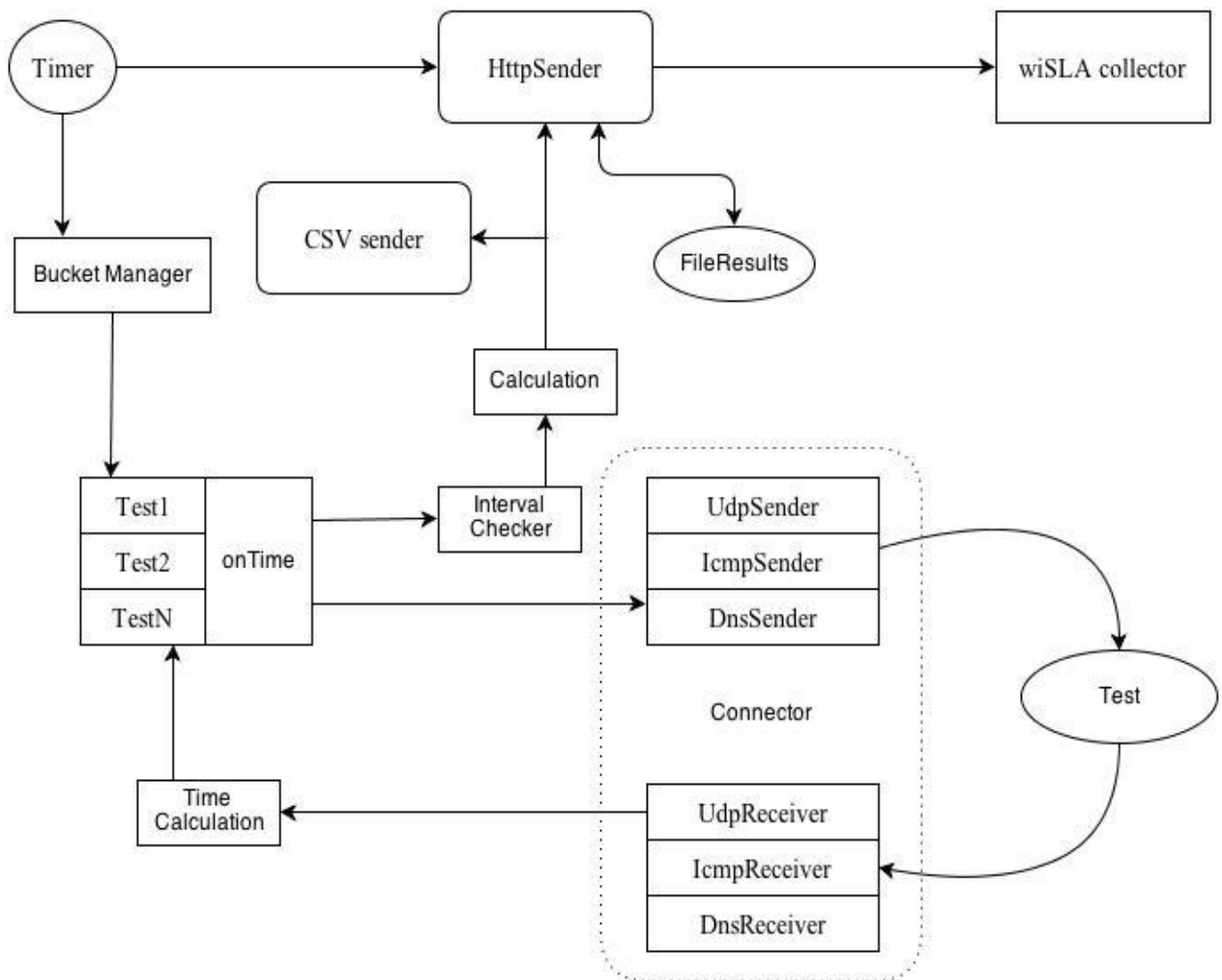


Рисунок 3.1 – Модель програмного забезпечення вимірювання характеристик передавання даних

Основне призначення об'єкту *Timer* полягає в синхронізації подій з відправки тестових пакетів і результату вимірів на зовнішній сервер.

Timer генерує 40 переривань в секунду і викликає функції з відправлення тестових пакетів, таким чином, щоб вони були рівномірно розподілені протягом секундного інтервалу. Цей інтервал дозволяє отримувати більш точні значення затримок відправлених пакетів, завдяки тому, що вони не накладаються один на одного при прийомі та передачі.

Один раз в хвилину, *Timer* викликає функцію відправки даних *HttpSender*, забезпечуючи досить оперативне інформування про стан вимірювальних каналів.

Bucket Manager – це модуль, який управляє активними тестами, які є актуальними в даний момент часу, виконуючи додавання і видалення їх з контейнера, виділяючи під них необхідні системні ресурси. Пошук тестів здійснюється по імені. Bucket Manager є сховищем векторів виконуваних тестів. Також модуль використовується для взаємодії з модулем telnet і ConfigServer, який взаємодіє із зовнішнім сервером wiSLA.

DataSender. HTTPSender виконує роль передачі отриманого результату SLA характеристик на зовнішній сервер. Дані в HTTPSender надходять з модуля Calculation і зберігаються в модулі FileResults, де контролюється максимальна кількість записів за певний проміжок часу, який глобально задається в налаштуваннях системи. Перед відправкою даних, перевіряється доступність сервера, для того щоб зменшити навантаження на файлову систему, де використовується компресія, що викликає високе завантаження ресурсів центрального процесора. Перед передачею, дані приводяться до необхідного формату, згідно з протоколом взаємодії із зовнішнім сервером. Потім зчитуються з файлової системи і формується запит до сервера, з подальшим відправленням. При успішній відповіді від сервера, файли відправлених даних видаляються. У разі невдачі, процедура відправки буде продовжувати виконуватися по Timer до отримання успішного відповіді.

CSVSender основне завдання полягає в збереженні результатів вимірювальних тестів в файл у форматі csv. Результати зберігаються за певний часовий проміжок, який задається в конфігурації системи. Дані надходять в модуль CSVSender з модуля, Calculation, також як в HTTPSender. Відмінність від зберігання інформації в модулі HTTPSender полягає в тому, що для кожного тесту створюються індивідуальні файли з назвами, що відповідають назві тесту на зовнішньому сервері.

Connector виконує функцію відправки та прийому тестових пакетів. Відправка тестового пакета ініціалізується тестом по Timer. Тест звертається до відповідного Connector згідно типу протоколу зазначеного в налаштуваннях тесту і передає параметри: номер пакета в тестовій послідовності і

ідентифікатор тесту. Connector на основі отриманих даних, формує і відправляє тестовий пакет. Для різних типів тестів використовуються різні структури тестових пакетів. Для icmp і udp використовується наступна структура пакету: номер пакета в тестовій послідовності, ідентифікатор тесту, час відправлення пакета і контрольна сума. DNS пакети формуються відповідно до стандартів протоколу [21], а прив'язка до тестів здійснюється в зіставленні ідентифікатора тесту з адресою DNS сервера і ім'ям хосту. Після відправки пакета модуль повертає час відправки в масив оброблюваного тіста.

Прийом тестового пакета відбувається асинхронно. Для icmp перевіряється контрольна сума тестових даних і здійснюється пошук ідентифікатора тесту. Для DNS перевіряється адреса сервера і ім'я хоста у відповіді і на підставі цих даних знаходиться тест. У udp перевіряється контрольна сума і ідентифікатор тесту, якщо тест відноситься до заведених на зонді, то повертається час прийому тестового пакета в масив тесту. У разі відповідності тільки контрольної суми відбувається приміщення часу прийому в тестовий пакет, розрахунок нової контрольної суми і пересилання пакета назад відправнику. Після прийому пакета, час прийому зберігається в масив тесту і проводиться розрахунок SLA характеристик для даного пакета: час односторонньої-двосторонньої затримки і Джітер.

Основне завдання модуля Calculation складається в розрахунку різних показників, таких як час односторонньої, двосторонньої затримки і час варіації затримки (jitter). За кожним показником також відбувається розрахунок середнього, мінімального і максимального значення, за певний інтервал часу, який вказаний в налаштуваннях тесту і подальшим відправленням результатів в DataSender. Модуль Calculation обчислює метрики безперервно - при отриманні нового тестового пакета, він розраховує показник, потім на основі його розраховує нове середнє значення і також перевіряє мінімальні і максимальні показники. Безперервний розрахунок усуває різкі скачки завантаження центрального процесора.

Таким чином, розроблено модель програмного забезпечення вимірювання характеристик передавання даних, яка характеризується об'єктами Timer, Bucket Manager, DataSender, Connector, Calculation, та дозволяє розробляти програмне забезпечення для вимірювання різних характеристик передавання даних.

3.3 Архітектура головної програми вимірювання характеристик передавання даних

На основі моделі, запропонованої у попередньому підрозділі, розроблено архітектуру програмного забезпечення.

В основу архітектури програми взято за шаблон singleton з framework РОСО. Після аналізу технічного завдання і проведення декомпозиції можна виділити наступні модулі:

- частина конфігурацій;
- керуюча частина;
- модуль для взаємодії із зовнішніми системами;
- модуль з відсилання результатів.

Після декомпозиції кожного модуля, вимірювальний додаток було розділене на дві програми: головну і керуючу. Головна програма проводить вимірювання і надсилання результатів. Керуюча дозволяє конфігурувати вимірювальний комплекс. Архітектура головної програми зображена на рисунку 3.2.

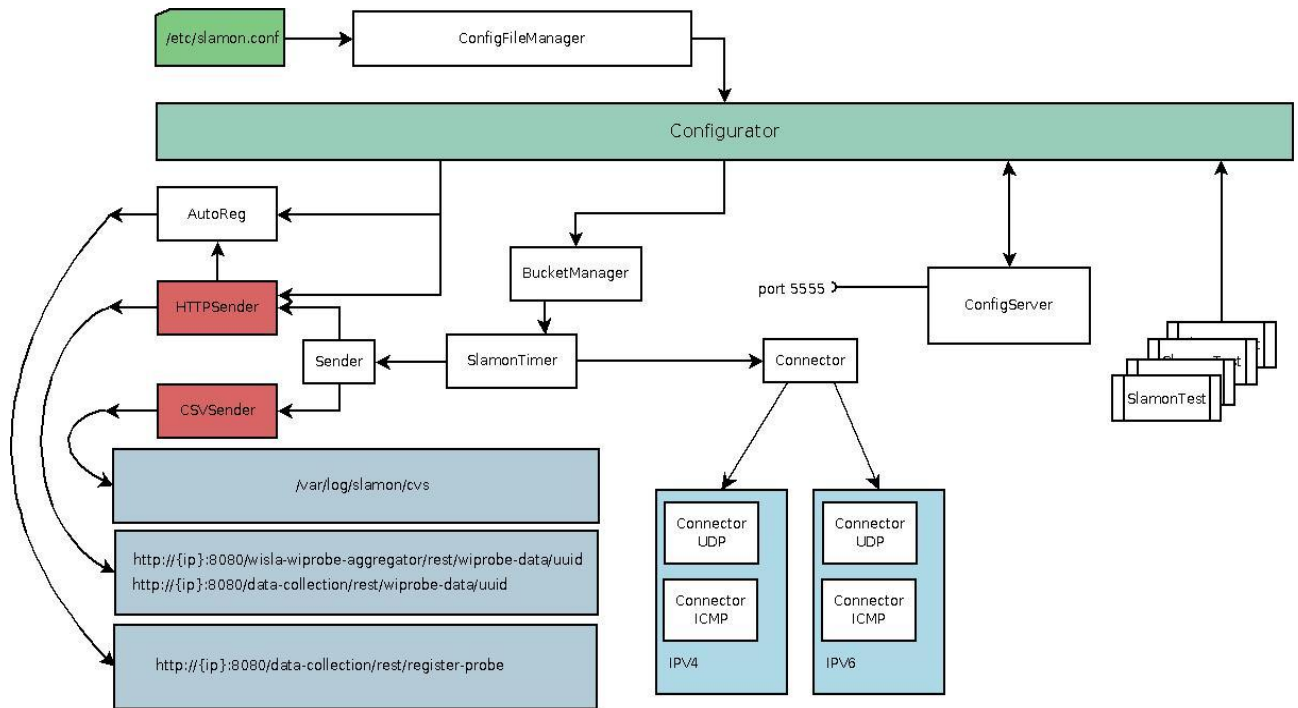


Рисунок 3.2 – Архітектура головної програми

Головний сервіс конфігурацій реалізований у вигляді класу `Configurator`, який виконаний за шаблоном `singleton`. Як і для всіх `singleton` сервіс є методом виклику статичної функції `instance()`. Наприклад, виклик у конфігуратору методу `getTestsQuantity()` - `Configurator::instance().getTestsQuantity()`.

`Configurator` реалізує:

- надання API для всіх модулів системи;
- конфігурація системи при запуску;
- конфігурація системи по службовому протоколу (`telnet/wisla`);
- керування тестами (запуск \ зупинка \ видалення).

Менеджер завантаження налаштувань з системного файлу `ConfigFileManager` завантажує інформацію про тести і системні параметри. Конфігураційний файл складається з наступних секцій:

- `system` – містить інформацію про системні налаштування, допустимі теги в секції `system`: `name` – ім'я вимірювального пристрою; `wisla-url` – ip - адреса зовнішньої системи, що збирає і аналізує дані вимірювального пристрою; `log-level` – рівень логування; `autoreg freq` – частота роботи керуючого

протоколу з зовнішньої вимірювальної системою; master-probe enable – опції роботи через proxy server;

- autoreg-url – список ip адрес зовнішніх систем для керуючого протоколу при роботі в кластерному режимі;

- properties – містить різні динамічні властивості;

- tests – містить інформацію про тести, допустимі теги в секції tests: name – ім'я тесту; dst – адреса вузла, що тестується; type – тип тесту. Реалізовані наступні типи: udp, icmp, dns, meta; interval – частота агрегації отриманих даних, вимірюється в секундах; frequency – період відсилання тестових пакетів в секундах; wtid – ідентифікатор тестів у зовнішній системі; size – розмір тестового пакета в байтах; tos – значення поля в ip заголовку.

Приклад конфігураційного файлу який містить один тест з ім'ям wislatest - який вимірює якість зв'язку з вузлом 192.168.2.93.

```
<config>
  <system>
    <name>probe_92</name>
    <wisia-url>http://192.168.2.123:8080</wisia-url>
    <log-level>debug</log-level>
    <autoreg freq="300"/>
    <master-probe enable="true" host="127.0.0.1" login="proxy"
password="INFlswBmCuGx" port="3128"/>
    <autoreg-url>
      <url>http://192.168.2.123:8080</url>
      <url>http://127.0.0.1:8080</url>
    </autoreg-url>
  </system>
  <properties>
    <put key="AGENCY" value="wellink.ru"/>
  </properties>
  <tests><test>
    <name>wislatest</name>
    <dst>192.168.2.93:8787</dst>
    <type>udp</type>
    <interval>300</interval>
    <frequency>1</frequency>
    <wtid>28</wtid>
    <size>128</size>
    <tos>7</tos>
  </test>
</tests>
</config>
```

Сервер конфігурацій ConfigServer виконаний у вигляді singleton і призначений для взаємодії з управляючою програмою. Взаємодія здійснюється через засоби універсального протоколу. ConfigServer створює сокет, який слухає на порту 5555. При встановленні зв'язку з сокета зчитуються всі доступні дані, розбиваються по «\n», ітеративно інтерпретуються і виконуються.

Для керування вимірювальною програмою був розроблений універсальний керуючий протокол, який дозволяє уніфіковано конфігурувати комплексом з зовнішньої системи і telnet сервера. Протокол текстовий порядковий, виконується через підрядник.

Протокол підтримує такі функції:

- GET_SETTING - 0 - читання параметра;
- SET_SETTING - 1 - запис параметра;
- GET_PROPERTY - 2 - читання властивості;
- SET_PROPERTY - 3 - запис властивості;
- GET_ALL_CHANNELS_ID - 4 - отримання всіх id тестів у вигляді списку;
- GET_TEST_BY_ID - 5 - отримання повного тесту по id;
- SAVE - 6 - збереження в конфігураційний файл;
- DELETE_CHANNEL - 7 - видалення тесту;
- ADD_CHANNEL - 8 - додання тесту;
- UPDATE_CHANNEL - 9 - оновлення тесту (спочатку видалення, потім додавання);
- START - 10 - старт тесту (використовується для RFC);
- STOP - 11 - стоп тесту (не використовується);
- OK - 12 - порожній запит;
- BACK_INFO - 13 - будь-яка додаткова інформація яка повернеться у відповіді з токеном EXTRA_INFO;
- ADD - 14 - не використовується;
- REMOVE - 15 - не використовується;

- GET - 16 - не використовується;
- SET - 17 - для спеціальних потреб;
- IPERF - 18 - для RFC тестів, використовую тільки slamon;
- ASSIGN_TEST_ID - 19 - призначити новий wisla-test-id;
- VALIDATE_CHANNELS - 20 - інвалідація активних каналів;
- STOP_ALL_CHANNELS - 21 - зупинити всі канали.

Список допустимих параметрів які можуть бути використані в GET_SETTING, SET_SETTING надані в таблиці 3.1.

Таблиця 3.1 – Список налаштувань протоколу

Ім'я	Код	Опис
NAME	0	ім'я пристрою
WISLA_IP	1	ip wisla
SELF_IP	2	власний ip
MASTER_ENABLE	3	використовувати чи проху для відправки на wisla
MASTER_IP	4	ip проху (повинна бути 127.0.0.1)
MASTER_PORT	5	порт проху
MASTER_LOGIN	6	логін проху
MASTER_PASSWORD	7	пароль проху
AUTOREG_ENABLE	8	включена автореєстрація (завжди true)
AUTOREG_FREQ	9	період спрацьовування автореєстрації
AUTOREG_IP	10	ip який шлется на wisla (не використовується)
UUID	11	UUID
SET_UUID	12	дозволено змінювати UUID (не використовується)
MAC_ADDR	13	мак
SLAMON_VERSION	14	версія slamon
DATA_URL	15	повний url к wisla для даних
AUTOREG_URL	16	повний url к wisla для автореєстрації
AUTOREG_URL_SIZE	17	кількість адрес автореєстрації

Для більш гнучкого конфігурування доступні наступні модифікатори:

- SKIP_TOKENIZER = 8000 - відключити токенайзер;

- WISLA_CMD = 8001 - команда wile;
- EXTRA_INFO = 8002 - будь-яка додаткова інформація;
- INTERFACE = 8005 - інформація про інтерфейс;
- HOST = 8006 - інформація про хост;
- UPTIME = 8007 – uptime;
- CPU_USAGE = 8008 - завантаження ЦПУ;
- TESTS_QUANTITY = 8009 - кількість активних тестів;
- TEST = 8010 - список тестів;
- MODE = 8011 - режим пристрою (master\slave);
- ROUTE = 8012 - список активних маршрутів;
- SYNCRONIZATION_MODE = 8013 - режим синхронізації (ntp\ptp).

У протоколі визначені наступні відповіді на запити:

- OK = 0;
- FAIL = 1;
- NOT_IMPLEMENTED = 2;
- BAD_REQUEST = 3;
- TIMEOUT = 4;
- VALUE = 5 // будь-яке число.

Приклади використання команд:

– “ADD_CHANNEL SKIP_TOKENIZER test 192.168.2.2:8 192.168.2.2:8 UDP 300 1 32 7” – запуск тесту UDP с ім'ям test на 192.168.2.2, з відправкою пакетів кожну секунду, розмір пакета 32 байта, TOS-7

– “ADD_CHANNEL SKIP_TOKENIZER dns-test 192.168.2.2:8 192.168.2.2:8 META:DNS 300 1 ya.ru mail.ru msn.com” – запуск DNS тесту на 192.168.2.2 і перевіркою ya.ru, mail.ru, msn.com

– “DELETE_CHANNEL dns-test” – видалення тесту dns-test

– “SET_SETTING DATA_URL http://192.168.2.132:8080/data-collection”

– зміна wisla-url для даних на http://192.168.2.132:8080/data-collection

– “10 8000 RFC@RFC 192.168.2.31:123 192.168.2.31:123 RFC 60 0 TX 1000000 32 TEST_ID:20150” – запуск RFC тесту.

Модуль Plug & Go призначений для автоматичного визначення в системі, завантаження сценаріїв з системи і виконання сценаріїв тестування. Модуль реалізований в класі AutoReg. В основі класу AutoReg бібліотечний клас POCO :: Runnable який в потоці, через певний проміжок часу (заданий в параметрі AutoReg_Freq) відправляє запит автореєстрації зовнішньої система по ip, що вказаний в параметрі WISLA_URL. У відповіді зовнішня система формує список сценаріїв, для виконання використовуючи універсальний протокол. Якщо зовнішня система при відправці даних надіслала код PARTIAL_CONTENT який означає, що у зовнішній системі є сценарії для виконання, то проводиться негайна відправка запиту автореєстрації.

Система відправки результатів HttpSender призначений для відправки результатів в зовнішню систему. Для економії трафіку використовується метод HttpSender :: checkUrl () який відправляє порожній http запит. Успішною перевіркою є код ACCEPTED (202). Решта кодів і станів кажуть про недоступність вузла.

Після перевірки віддаленого вузла починається відправка даних. Якщо в процесі відправки виникає помилка, то встановлюється спеціальна змінна - isNetOk в false - яка сигналізує, що на попередній ітерації виникла проблема з відправкою даних і відправку необхідно повторити REPEAT_COUNTER раз (зараз 5 разів). В результаті HttpSender гарантовано надасть результати тестів на зовнішню систему.

Конфігурація HttpSender складається з установки наступних опцій:

- HttpData_Path – де http зберігає свої дані на відправку;
- HttpCounts_Files – максимальна кількість збережених результатів;
- Outdate – час зберігання файлів з результатами тестів;
- HttpSend_Freq – період відправки даних;
- HttpSend_Url – ip-адреса для відсилання даних на зовнішню систему (за замовчуванням використовується WiSLA_URL).

Вся інформація по тесту зберігається в структурі типу SlamonTest і може мати ієрархічну або атомарну структуру. У всіх тестах крім DNS використовується атомарна структура. Кореневим вузлом DNS тесту є звичайний тест, який виконує роль контейнера і має тип тесту META. Всі дочірні тести успадковують властивості контейнера і мають можливість для перевизначення деяких властивостей (frequency, type).

Для реалізації тестів навантаженням за стандартом RFC2554 використовується програма iperf.

Iperf – кросплатформна консольна клієнт-серверна програма, що використовується як генератор TCP і UDP трафіку для тестування пропускної здатності мережі. Схема взаємодії програми з iperf зображена на рисунку 3.3.

Алгоритм проведення RFC2554 в прямому напрямку описаний нижче.

Зовнішня система формує і відправляє завдання на проведення RFC2554 тесту. Slamon1 приймає завдання інтерпретує і синхронно відправляє завдання на старт iperf в режимі сервера на пристрої Slamon2, Slamon1 запускає iperf в режимі клієнта. Slamon2 приймає завдання, інтерпретує її і запускає iperf в режимі сервера. При успішному запуску iperf сервера, Slamon2 повертає код ОК, в іншому випадку повертає FAIL. Якщо отримана відповідь має статус ОК, то Slamon1 конфігурує і запускає iperf в режимі клієнта. Iperf клієнт конфігурує сервера iperf і починає генерувати udp трафік. Slamon2 зчитує результати з сервера iperf. Slamon2 інтерпретує отримані результати з сервера iperf, перетворює їх до внутрішнього формату і відправляє на зовнішню систему.

При виконанні тестів в зворотному напрямку послідовність тестів аналогічна прямому напрямку, за винятком положення iperf сервера і клієнта, міняються місцями.

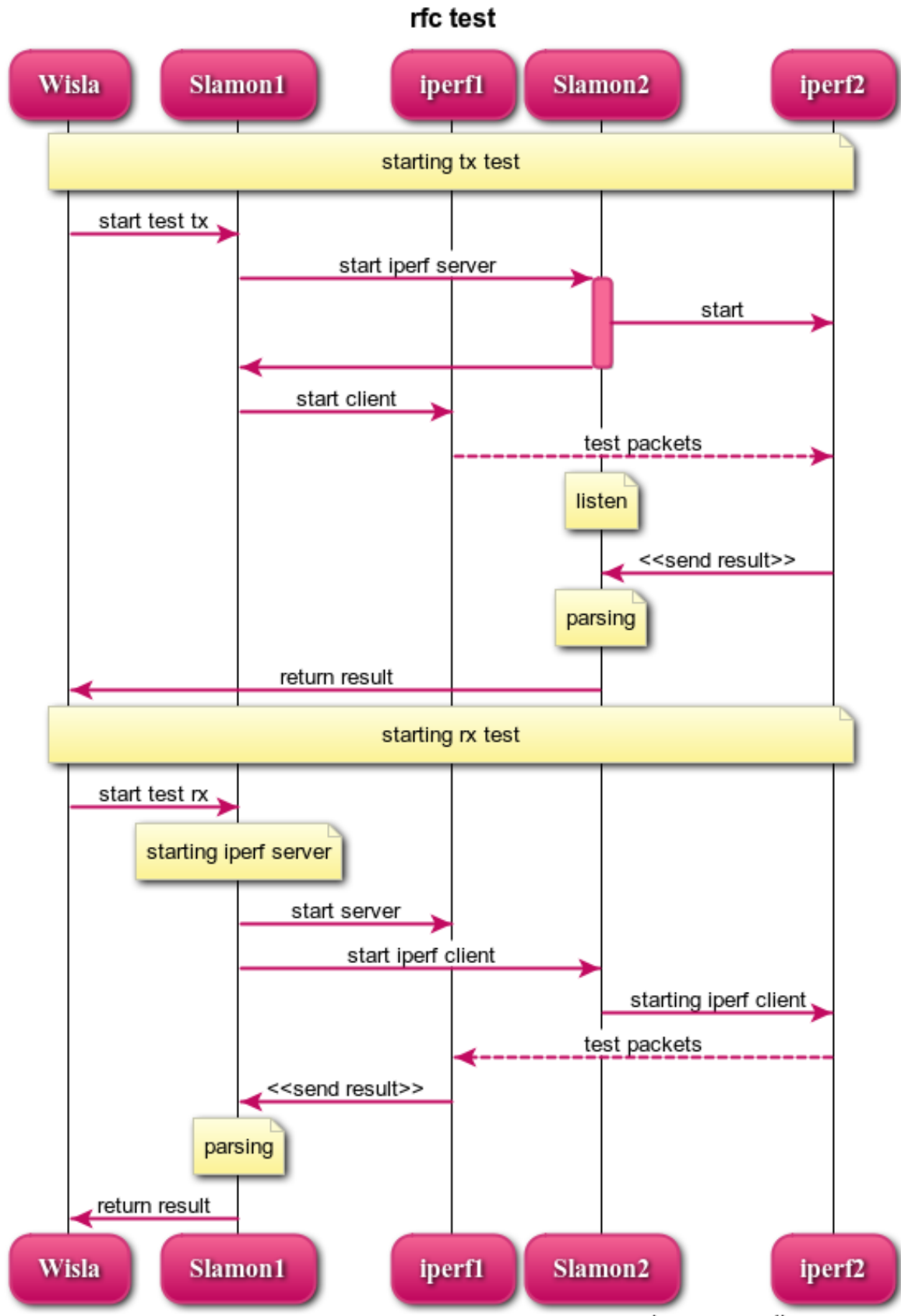


Рисунок 3.3 – Схема проведення RFC 2554 тесту

3.4 Архітектура конфігураційної програми

За основу архітектури програми було взято шаблон каркаса додатку з framework `POCO::ServerApplication`, який надає необхідний функціонал для створення системного сервісу, а саме:

- читання і збереження налаштувань програми;
- поділ на логічні сервіси за допомогою шаблону `poco::singleton`;
- обробка ключів переданих програмі.

Після декомпозиції вимог наданих до керуючої програмі були виділені наступні сервіси:

- telnet сервер;
- керування сесіями користувача;
- інтерпретація і виконання команд, що вводяться користувачем;
- підсистема взаємодії з вимірюваною програмою;
- система взаємодії з операційною системою;
- система конфігурації мережевих інтерфейсів.

Кінцева архітектура програми конфігурацій представлена на рис. 3.4.

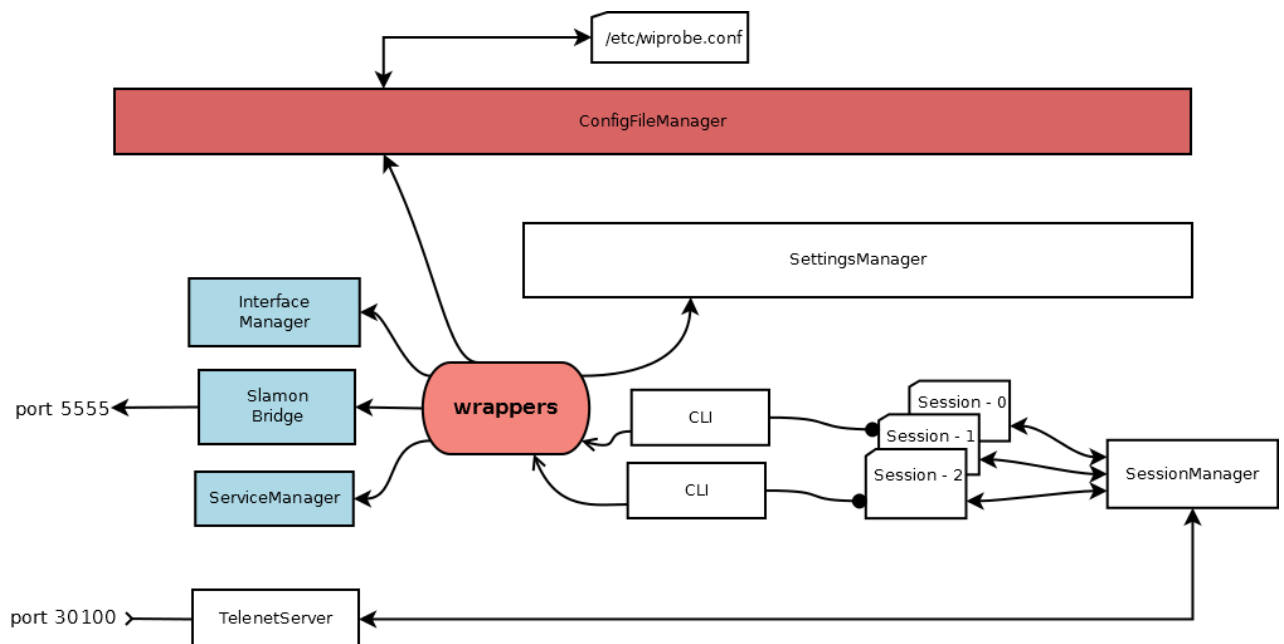


Рисунок 3.4 – Архітектура програми конфігурацій

Менеджер завантажень налаштувань з системного файлу. Файл з конфігураціями розташований в /etc/wiprobe.conf і представляє з себе xml структуру. Приклад файлу конфігурацій зображено нижче.

```
<config>
  <system>
    <server-port>30100</server-port>
    <max-connections>3</max-connections>
    <max-queued>2</max-queued>
    <session-timeout>5</session-timeout>
    <mode>slave</mode>
    <master-ip>192.168.2.92</master-ip>
    <ftp-url>ftp://user:user2010@ftp.wellink.ru/uploads/wiprobe-update</ftp-url>
  </system>
  <ntp>
    <server>192.168.2.92</server>
  </ntp>
  <interfaces>
    <interface access="ROOT" enable="true" static="true" vlan="false">
      <name>eth0</name>
      <ip>192.168.1.1</ip>
      <netmask>255.255.255.0</netmask>
    </interface>
  </interfaces>
  <users>
    <user>
      <login>admin</login>
      <password>admin</password>
      <role>ADMIN</role>
    </user>
  </users>
</config>
```

Файл конфігурацій складається з 3-х головних секцій:

- system – зберігається опис системних параметрів;
- interfaces – зберігається інформація про мережеві інтерфейси пристрої;

- users – зберігає інформацію про користувачів;

Допустимі опції секції system:

- server-port – слухає порт telnet сервера;
- max-connections – максимальна кількість одночасних сесій;
- max-queued – максимальну кількість сесій, що очікуються;

– session-timeout – максимальний час не активності через, який сесія буде видалена;

– mode – режим пристрою master/slave;

– master-ip – ip адреса головного пристрою якщо обраний режим slave (якщо режим master дане поле ігнорується);

– ftp-url – ftp адрес серверу для отримання нових версій програми.

Допустимі опції секції users:

– login – ім'я користувача;

– password – пароль користувача;

– role – повноваження користувача (може бути admin або operator).

У пристрої реалізовано 2 ролі admin і operator. Admin має максимальні повноваження і має доступ до всіх команд. Operator дозволено переглядати встановлені параметри, поточний стан і заборонено змінювати конфігурацію системи.

Допустимі опції в секції interfaces:

– interface – оголошує новий інтерфейс і має наступні параметри:

– access – необхідний рівень повноваження користувача для здійснення операцій з інтерфейсом (включити, відключити,...);

– enable – чи буде інтерфейс запускатися при старті системи;

– static – чи буде інтерфейс отримувати ip адресу від dhcp сервера;

– vlan – чи є даний інтерфейс vlan.

– name – ім'я інтерфейса всередині системи;

– ip – ip-адреса інтерфейсу (якщо параметр static встановлений в false дане поле ігнорується і адреса виходить за коштами dhcp);

– netmask – мережева маска інтерфейсу;

– default-gw – шлюз може бути призначений тільки одному інтерфейсу;

– route – список маршрутів. Допустимі параметри:

– fn – дії виконанні над маршрутом при вмиканні або вимиканні мережевого інтерфейсу;

- ip/mask – підмережа для якого призначено маршрут;
- gateway – шлюз, що обслуговує маршрут.

Сервіс управління системними інтерфейсами IntrerfaceManager призначений для управління мережевими інтерфейсами. У його завдання входить:

- створення, конфігурування, видалення інтерфейсів;
- робота з ipv6 адресами;
- генерування системного файлу /etc/networking/interfaces;
- додавання, редагування, видалення маршрутів на заданий інтерфейс;
- управління інтерфейсами (запуск, установка);
- управління інтерфейсами типу vlan.

Вся інформація про структуру зберігається в класі InterfaceManager :: Interface. При збереженні налаштувань запускається генератор, який послідовно генерує інформацію в системний файл, таким чином файл / etc / networking / interfaces завжди створюється з /etc/wiprobe.conf і ручна правка не призведе до змін в інтерфейсах.

Сервіс взаємодії з головною вимірювальною програмою SlamonBridge призначений для взаємодії з головною вимірювальною програмою. Управління відбувається за універсальним протоколом. Взаємодію реалізовано у вигляді RPC виклику SlamoBridge :: CallSlamonFn, в який передається номер функції і список параметрів. При зверненні до slamon керуюча програма відкриває socket, встановлює з'єднання до керуючого сервера, посилає передану функцію з параметрами, зчитує результат і закриває з'єднання. Результат повертається програмі для подальшого використання. Особливістю взаємодії з головною програмою в тому, що відсутнє поняття сесії, було зроблено для спрощення взаємодії і відсутності реалізації додаткової підсистеми, для управління сесією.

Менеджер управління системними службами (системний менеджер) призначений для взаємодії з операційною системою. У його завдання входить:

- робота з системними службами;
- керування службами синхронізації часу;

- взаємодія з системними службами: 3prox, stunnel;
- перемикання режиму пристрою.

ServiceManager дозволяє додавати/видаляти служби, переглядати їх поточний стан, запускати і зупиняти необхідні служби, а також формувати списки спостереження за обраними сервісами. Підсистема конфігурування синхронізації часу працює зі службами ntp і rtp. Дозволяючи переключити необхідну систему синхронізації. Також служба дозволяє переглядати поточний стан синхронізації, а для служби ntp дозволяє додавати, видаляти і редагувати сервера синхронізації часу.

Взаємодія зі службами stunnel і 3proxу полягає в генеруванні їх файлів налаштувань враховуючи поточний режим пристрою.

У пристрої реалізовано 2 можливих режиму роботи master і slave. У режимі slave пристрій може звертатися тільки до master пристрою, таким чином формуючи свою власну ізольовану мережу. Master представляє собою зашифрований проху сервер, який має доступ до інших мереж і з'єднує ізольовану підмережу slave пристроїв із зовнішнім світом. При перемиканні режиму роботи пристрою ServiceManager наступні файли генеруються заново:

- wget.rc;
- 3proxy.cfg;
- stunnel.cfg;
- ntp.conf;
- rtp.conf.

Потім проводить маніпуляції щодо зупинки, запуску, видалення, додавання, рестарту необхідних сервісів.

Telnet сервер використовує стандартний шаблон tcp сервера РОСО :: TcpServer. У ньому міститься пул потоків на кожне з'єднання, який слухає сокет і фабрику сесій. У модулі була імплементована фабрика сесій TelnetSessionCreator, яка проводить базову ідентифікацію користувача і якщо аутентифікація пройшла успішно, то створена сесія передається на управління

SessionManager. Також фабрика встановлює timeout, це час не активності, після якого сесія буде автоматично видалена.

Менеджер сесій SessionManager виконує керування сесіями. Після того як користувач пройшов аутентифікацію і сесія була додана в SessionManager, він зберігає її на карту пам'яті, встановлює id сесії, час аутентифікації і список дозволених дій з огляду на роль користувача. SessionManager зберігає змінну globalLock, яка дозволяє тільки одному користувачу виконувати монопольне конфігурування пристрою. Після введення команди cfg відбувається блокування mutex, читання змінної globalLock і якщо вона не встановлена у true користувачеві передається можливість монопольної конфігурації пристрою, змінна globalLock встановлюється в true і розблокується mutex. Монопольну конфігурацію введено для усунення конфліктів, у випадку коли кілька користувачів намагаються змінити один параметр. SessionManager також містить функції з видалення, додавання і редагування користувачів.

Модуль надання доступу по CLI

Підсистема command line interface імплементована в класі CLI і має один допоміжний клас, який описує команду, LEAF. Системи команд представляють собою ієрархічні дерева які складаються з класів LEAF .

```
class Leaf
{
    string process(deque<string>&);
    string directCall(deque<string>&);

private:
    string helpMsg;
    cmd action;
    string name;
    string alias;
    vector<Leaf*>childs;
    Cli* parent;
    string defAction(Session* ctx, deque<string>& params);
};
```

Клас LEAF складається з наступних полів:

- helpMsg – рядок допомоги по команді;
- action – покажчик на функцію обробник;

- name – ім'я команди;
- alias – коротке ім'я команди;
- childs – список доступних підкоманд;
- parent – покажчик на батьківське дерево.

Клас LEAF містить наступні методи:

- process – виконує інтерпретацію і виконання команди користувача;
- directCall – сприймає команду від користувача як дані не викликаючи інтерпретатор.

Приклад дерева команд представлений на рисунку 3.5.

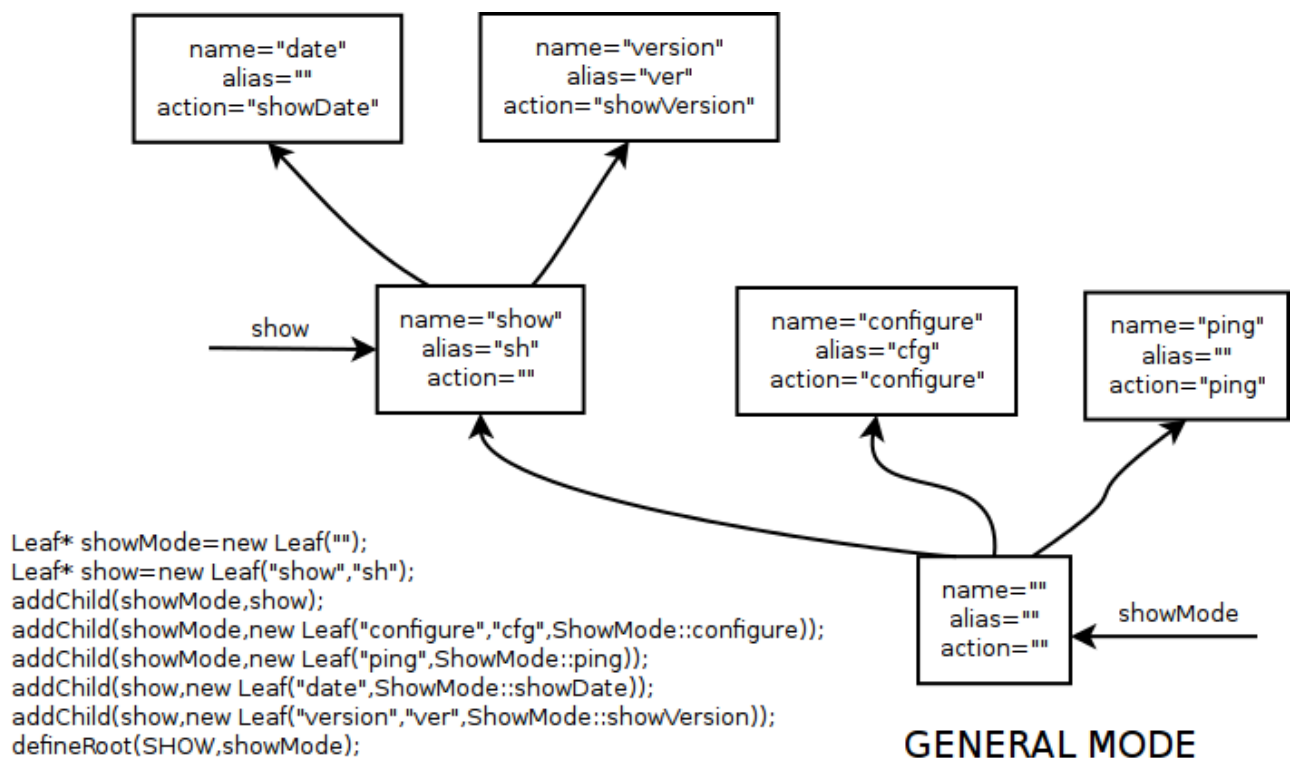


Рисунок 3.5 – Древоподібна структура команд

В даному прикладі описані команди show, cfg, ping. Команда show має підкоманди date и version.

Алгоритм інтерпретації та виконання команди show ver описаний нижче.

Sh ver розбивається на два токена sh и ver. Список токенів передається обробнику showMode. В showMode не встановлено ім'я, псевдонім і починає

послідовно порівнювати значення першого токена (show) з дочірніми командами. Після виявлення збігу за ім'ям прибирає з вектора перший токен і отриманий вектор передає оброблювачу showMode. Так як в команді showMode не заданий обробник за умовчанням, то відбувається виявлення збігу в дочірніх командах, якщо збіг не буде виявлено то буде викликана функція defAction(). Після виявлення збігу за ім'ям прибирає з вектора перший токен і отриманий вектор, вже порожній, передає оброблювачу version. У команді version встановлений обробник showVersion (), який викликається і повертає користувачеві результат команди.

Всі команди згруповані в режими, що представляють собою логічні та функціональні групи команд. Реалізовані наступні режими:

- INIT - режим авторизації, не містить команд;
- SHOW - основний непривілейований режим, доступний всім, призначений тільки для перегляду інформації;
- CONFIGURE - основний режим конфігурації, встановлює змінну globalLock в true, забороняючи іншим сесій вхід в цей режим, вимагає права доступу типу admin.

Наступні режими доступні тільки з режиму конфігурації, вихід з них здійснюється командою done:

- TEST - режим редагування тесту;
- INTERFACE - режим редагування інтерфейсу;
- SLAMON - режим редагування параметрів slamon.

Взаємодія режимів являє собою кінцевий автомат представлений на рисунку 3.6.

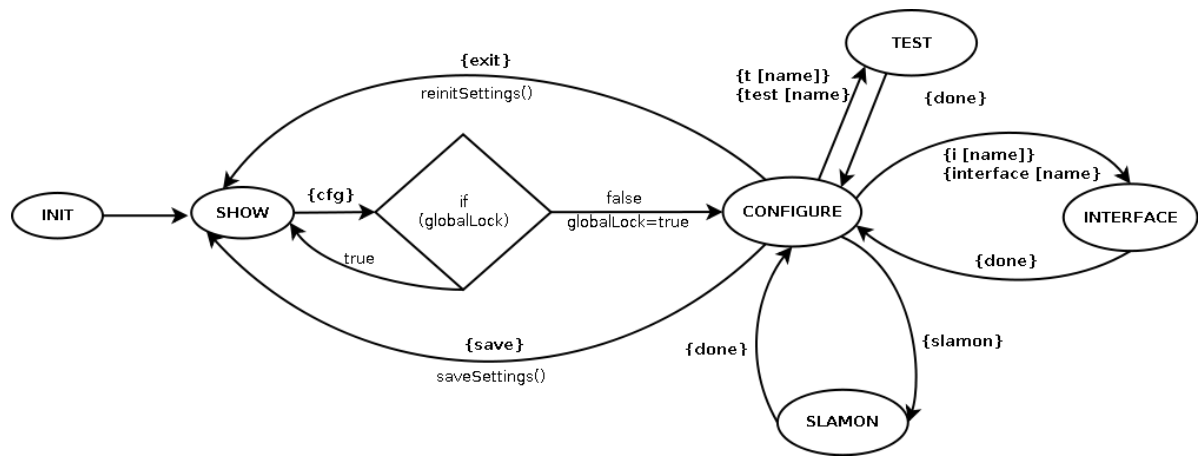


Рисунок 3.6 – Схема взаємодії режимів програми конфігурації

Після авторизації користувача встановлюється перший головний режим SHOW, який має безліч програм, які дозволяють переглядати різні параметри системи. При вводі користувачем команди `cfg` системою перевіряється глобальна змінна `globalLock`. Якщо вона встановлена, то користувач повертається в SHOW режим. Якщо ж вона не встановлена, вона встановлюється і користувач переходить в режим CONFIGURE. З цього режиму є безліч команд по редагуванню різних параметрів і властивостей системи. Так само є 3 додаткових режими в яких можливо редагувати параметри інтерфейсу тесту і параметрів вимірювальної програми. Вихід з додаткових режимів в режим CONFIGURE здійснюється командою `done`. Вихід з режиму CONFIGURE можливий з використанням двох команд `exit` і `save`. При введенні команди `save` викликається функція `saveSettings()`, яка застосовує і зберігає всі зміни в файлі. При введенні команди `exit` викликається функція `reinitSettings()`, яка скасовує всі зміни зроблені користувачем.

Setting Manager призначений для керування всіми параметрами системи. Основним завданням Setting Manager є запис всіх змін внесених користувачем в журнал і одночасне виконання їх після виходу з конфігураційного режиму. Всі змінювані параметри заносяться в журнал змін, які можна переглянути командою `show new`, перед збереження. На екран будуть виведені всі зміни які зробив користувач за кожним типом сервісу. Після введення команди `save` всі

зміни зберігаються в файл `wiprobe.conf` і потім система виконує всі дії які були занесені в журнал. Якщо користувач ввів команду `exit`, то всі зміни скидаються в стан описаний в файлі `wiprobe.conf`.

Системний помічник. Майстри або системні помічники призначені для спрощення взаємодії з користувачем, позбавляючи їх від необхідності вивчення складної системи команд і надають можливість вводити відповіді на поставлені запитання. В основі системного помічника лежить абстрактний шаблонні клас `AbstractMaster`. Вихідний код абстрактного класу представлений на нижче.

```
template <class T>
class AbstractMaster
{
public:
    AbstractMaster(){}
    virtual void setAnswer(string val){answer=val;}
    virtual void init()=0;
    virtual void reinit()=0;
    virtual string getNext()=0;
    bool isComplete(){return complete;}
    bool isReady(){return ready;}
    virtual T*getResult(){return data;}
    virtual ~AbstractMaster(){}
protected:
    bool complete;
    bool ready;
    string answer;
    T* data;
};
```

Завдання, що виконуються `AbstractMaster`:

- управління пам'яттю – `AbstractMaster` має методи `init`, `reinit` `getResult`, які реалізують інтерфейс з видалення, додавання і реініціалізації об'єктів в пам'яті;
- надає уніфікований інтерфейс для взаємодії з майстром через функції `setAnswer` та `getNext`;
- визначає стан майстра через функції `complete` і `ready`.

Кожен майстер представляє собою кінцевий автомат, який повинен імплементуватися користувачеві функцією `getNext` та в залежності від стану та переданої відповіді автомат може перейти на інший стан і згенерувати

відповідь на введені користувачем дані. В системі реалізовані наступні системні помічники:

- додавання нового тесту;
- додавання нового користувача;
- установка маршруту за замовчуванням;
- додавання інтерфейсу;
- додавання ipv6 інтерфейсу на існуючий ipv4.

3.5 Функціональне та навантажуваче тестування програмного забезпечення

У функціональному тестуванні беруть участь один тестований пристрій, один еталонний пристрої (landslide) і сервер зі створення штучних затримок і втрат пакетів - tc. Функціональне тестування проведено у 8 режимах, де в кожному наступному режимі буде збільшуватися час затримки і коефіцієнт втрати пакетів. Характеристики режиму зображені в таблиці 3.2.

Таблиця 3.2 - Конфігурація режимів тестування

Номер режиму	Коефіцієнт втрати пакетів, %	Час затримки пакетів, мс.
1	1	10
2	10	50
3	19	90
4	28	130
5	37	170
6	46	210
7	55	250
8	64	290

Результати тестування представлені на рисунку 3.7.

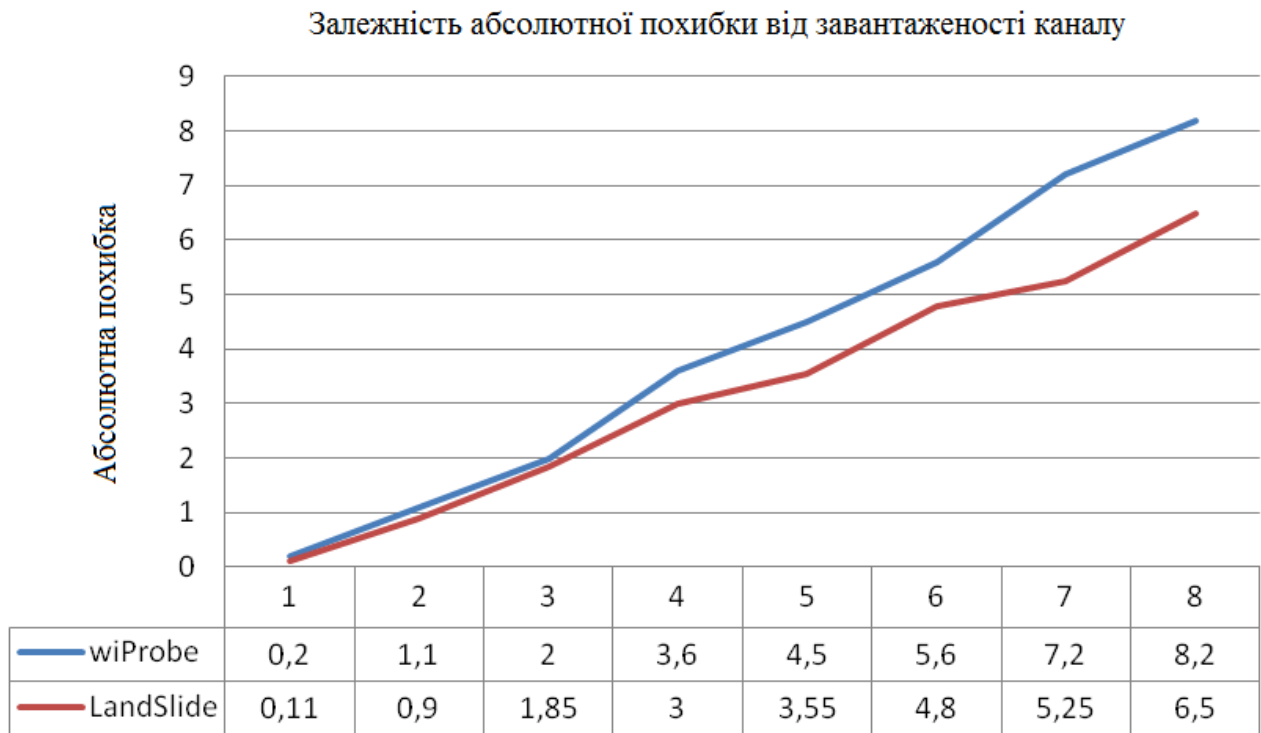


Рисунок 3.7 – Залежність коефіцієнта похибки від завантаженості каналу

В результаті проведеного тестування можна спостерігати, що розроблений пристрій має невеликі похибки щодо еталонного пристрою.

Тестування навантаженням. Для перевірки твердження про велику кількість одночасно проведених тестів необхідно провести тестування навантаженням. При цьому виді тестування пристрій буде проводити одночасно 100 різних типів тестів. Тестування буде супроводжуватися моніторингом наступних параметрів:

- завантаження центрального процесора;
- споживання оперативної пам'яті;
- достовірність результатів.

Тестування буде проводитися за схемою зображеною на рисунку 3.8.

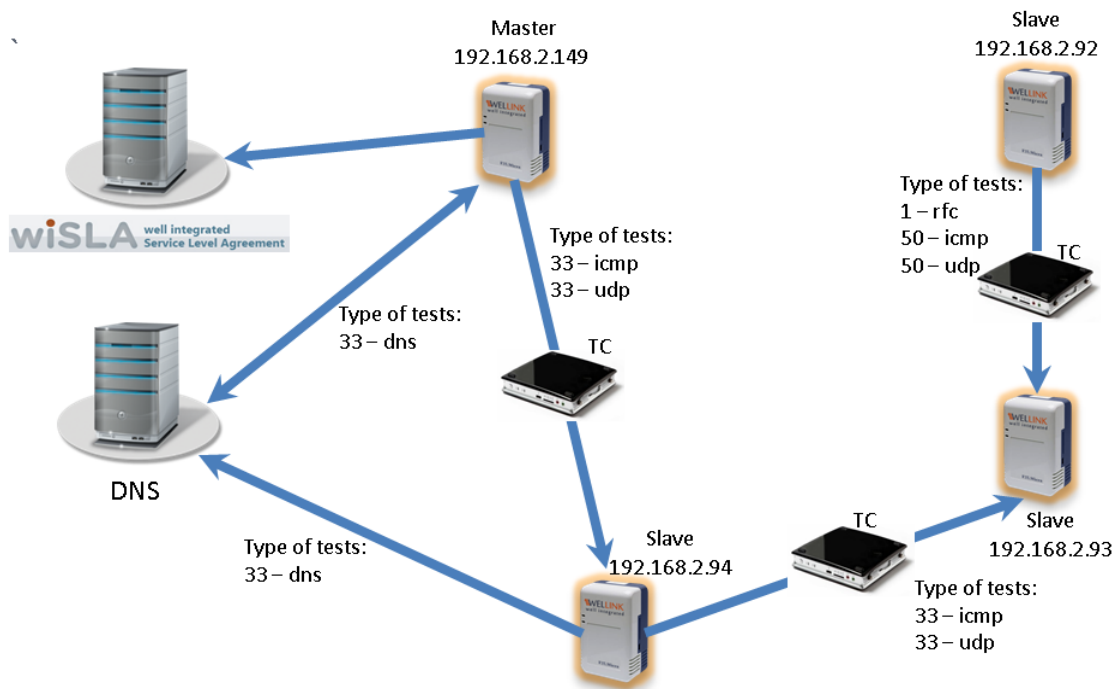


Рисунок 3.8- Схема тестування wiProbe

У тестуванні навантаженням бере участь 3 slave пристрої, які знаходяться у своїй власній підмережі 192.168.0.0/24, одного master який шифрує і проксує на зовнішні підмережі 192.168.4.0/24, одного DNS сервера.

На пристрої 192.168.2.149, який є master, заведено 33 тесту на DNS сервер і 66 вимірювальних тестів на вузол 192.168.2.94 через систему штучного зашумлення каналу tc. На пристрої 192.168.2.94 заведена подібна конфігурація. Пристрій 192.168.2.92 має відмінну конфігурацію від попередніх, яка складається в одному rfc2554 тесті і 100 вимірювальних тестів на вузол 192.168.2.93.

В результаті тестування навантаженням були отримані наступні результати, показані на рисунках 3.9 – 3.11.

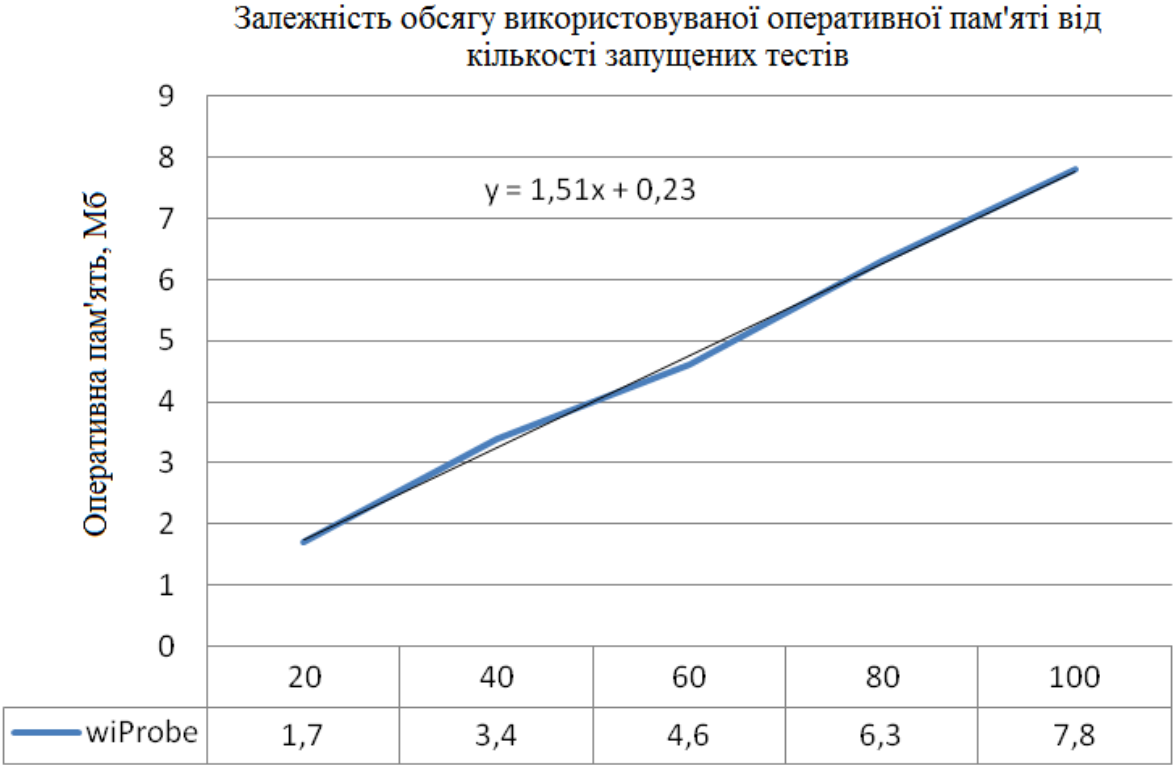


Рисунок 3.9 – Залежність використовуваної оперативної пам'яті від кількості тестів

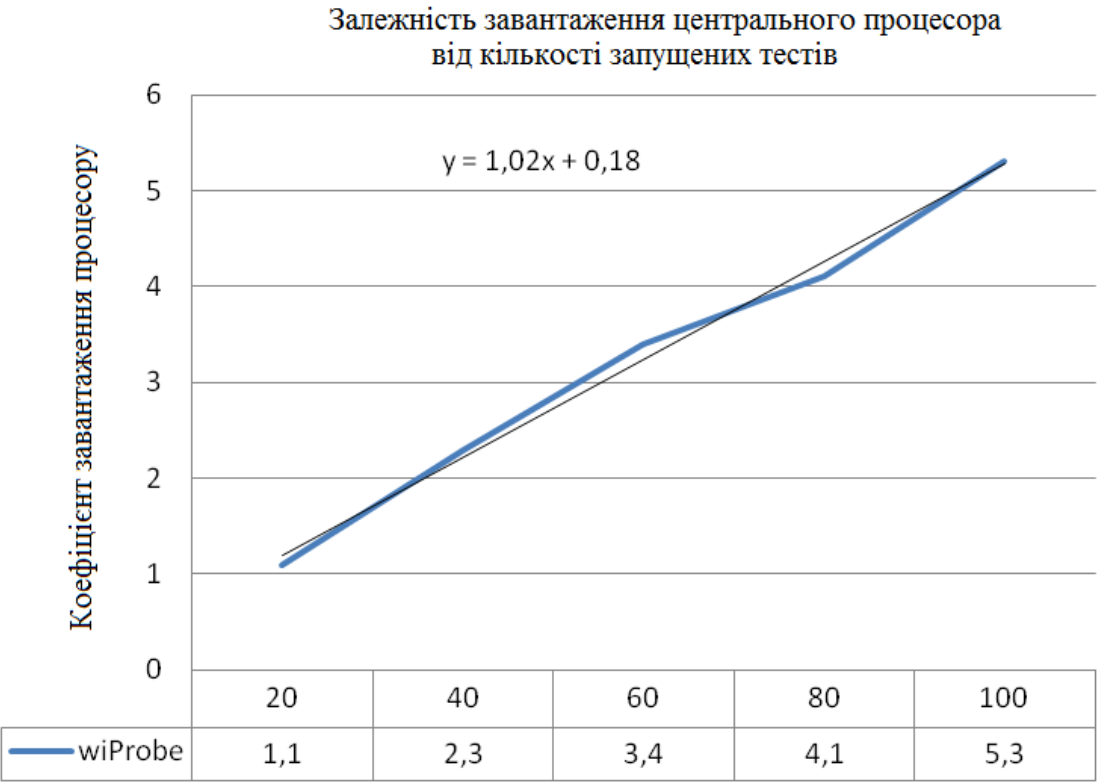


Рисунок 3.10 – Залежність завантаження центрального процесора від кількості запущених тестів

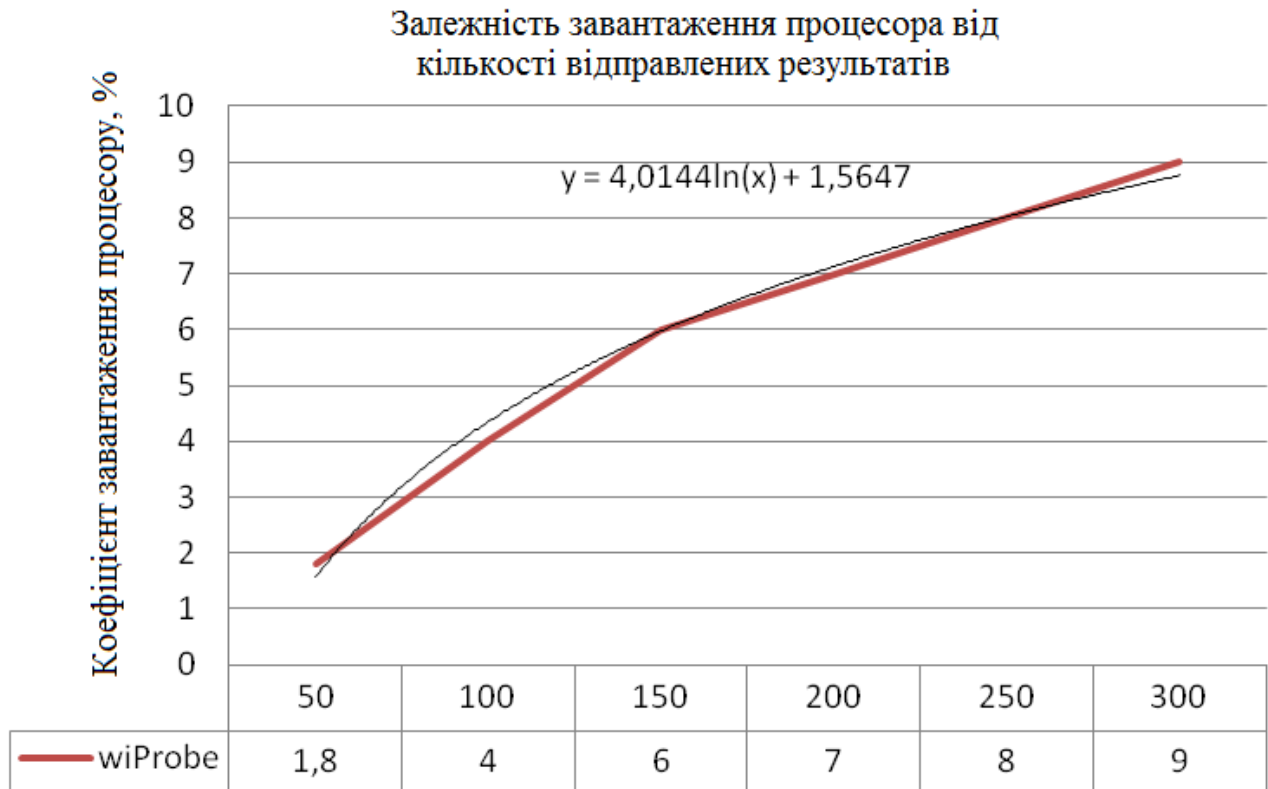


Рисунок 3.11 – Залежність завантаження процесора від кількості відправлених результатів

Після інтерполяції навантажувальних кривих можна зробити висновок, що кожні 20 тестів збільшують завантаження процесору на 1%, залежність прямолінійна, і 1,5 Мб оперативної пам'яті. Завантаження master має логарифмічну залежність від кількості відправлених результатів.

3.6 Висновки за розділом 3

Визначення основні метрики, що повинні вимірюватися розроблюваним програмним забезпеченням.

Розроблено модель програмного забезпечення вимірювання характеристик передавання даних, яка характеризується об'єктами Timer, Bucket Manager, DataSender, Connector, Calculation, та дозволяє розробляти програмне забезпечення для вимірювання різних характеристик передавання даних.

На основі запропонованої моделі розроблено архітектуру програмного забезпечення вимірювання характеристик передавання даних.

Проведено декомпозиції керуючої програми відповідно до поставлених вимог. Програма була розділена на 2 підпрограми: та, що вимірює (головна програма) та конфігураційна програма. Кожна програма в свою чергу була розбита на логічні модулі. Розроблено уніфікований протокол, який дозволяє налаштувати програму, як за сожливостями зовнішньої системи, так і telnet сервером.

Визначено, що архітектура головної програми вимірювання характеристик передавання даних складається з таких модулів: частина конфігурацій; керуюча частина; модуль для взаємодії із зовнішніми системами; модуль з відсилання результатів. Імплементовано головний вимірювальний додаток, який підтримує dns, icmp, udp, rfc2554 тести які можуть працювати одночасно у великій кількості. Реалізовано модуль Plug & Go який реєструє пристрій у зовнішній системі, зчитує сценарії з зовнішніх систем і виконує їх на пристрої, за коштами уніфікованого протоколу. Розроблено модуль з відправки і збереження готових даних, який дозволяє зберігати дані у вигляді csv файлів і передавати дані зовнішньої системі за допомогою http протоколу.

Запропоновано архітектуру конфігураційної програми. Розроблено програму, яка дозволяє створювати, видаляти, редагувати інтерфейси, взаємодіяти з операційною системою, перемикає режими пристрою і сервіси синхронізації часу, взаємодіяти з вимірювальною програмою (створювати, видаляти, редагувати тести). Реалізовано системні помічники, які дозволяють виробляти складні дії відповідаючи на питання системи. Опрацьована гнучка система команд, реалізований їх парсер і виконання.

Проведено функціональне і тестування навантаженням, що дозволило об'єктивно оцінити коефіцієнт похибки вимірювань, відповідність всім функціональним вимогам, а також споживання ресурсів пристрою при проведенні великої кількості одночасних тестів.

В результаті проведених функціонального тестування та тестування навантаженням було виявлено, що розроблений комплекс володіє необхідними вимогами визначеними в технічному завданні. Експериментально доведено, що розроблені алгоритми мають невеликий коефіцієнт похибки при роботі в завантажених лініях зв'язку. Тестування навантаженням довело, що вимірювальний пристрій при проведенні великої кількості тестів використовує не велику кількість ресурсів, залишаючи великий запас ресурсів для різноманітних призначених для користувача завдань (проведення додаткових тестів, виконання різного призначеного для користувача програмного забезпечення).

Режим подвійної синхронізації дозволив скоротити похибки на завантажених каналах до 1%, а використання мови програмування C++ дозволило істотно зменшити споживання ресурсів центрального процесора і обсягу використовуваної оперативної пам'яті до 1% CPU і 1,5Мб при виконанні 20 тестів.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови застосування програми

Програмно-апаратний комплекс реалізований у вигляді керуючої і вимірювальної програм, що призначена для запуску на POSIX сумісних операційних системах.

Підтримуються доступні платформи описані в POCO: Platform.

Для роботи вимірювального комплексу необхідний IBM PC або ARM - комп'ютер не нижче наступної конфігурації:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 50 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша;
- мережева карта, не менше 100Mbps.

4.2 Характеристики програми

Програма дозволяє виконувати вимірювання характеристик передавання даних. Для установки програмного забезпечення необхідно скопіювати установочний архів, розпакувати його і виконати `install.sh` з правами супер користувача. При виконанні скрипта будуть:

- створені скрипти сервісів в папці `/etc/init.d`;
- додані сервісні скрипти в автозавантаження;
- виконані копіювання бінарних файлів в `/usr/local/bin` і конфігураційних в `/etc`;
- згенерована конфігурація для системного планувальника `cron`;
- копійовані конфігураційні файли `stunnel` и `3proxy`;

- встановлена система скидання в початковий стан;
- створено деінсталлятор `/usr/local/bin/uninstall.sh`;
- створені супервізори;
- створений скрипт для оновлення ПЗ `/usr/local/bin/update.sh`.

4.3 Звернення до програми

Для запуску вимірювальної програми необхідно виконати команду `/etc/init.d/slamond start`. Для запуску конфігураційної програми необхідно виконати `/etc/init.d/wiprobd start`. Для скидання налаштувань в початковий стан необхідно утримувати кнопку перезавантаження пристрою більш ніж 15 секунд.

4.4 Налаштування програми

Програмне забезпечення має такі конфігураційні файли:

- `/etc/wiprobe.conf` – конфігураційний файл керуючої програми;
- `/etc/salmon.conf` – конфігураційний файл вимірювальної програми;
- `/etc/cron.d/wiprobe.cron` – містить дані про автоматичне оновлення і супервізор системи.

Програма `wiprobe` генерує або налаштовує такі файли:

- `/etc/stunnel/stunnel.conf` – змінює інформацію про проху сервері;
- `/usr/local/etc/3proxy/3proxy.cfg` – вимірює режим проксування;
- `/etc/networking/interfaces` – генерує мережеві інтерфейси використовуючи `/etc/wiprobe.conf`;
- `/etc/wgetrc` – конфігурує використання проксі сервера;
- `/etc/ntp.conf` – змінює список серверів за якими буде синхронізуватися час;
- `/etc/ptp.conf` – змінює режим синхронізації.

5 КЕРІВНИЦТВО ОПЕРАТОРА

5.1 Призначення програми

Програма призначена для вимірювання характеристик передавання даних.

Програмно-апаратний комплекс реалізований у вигляді керуючої і вимірювальної програм, що призначена для запуску на POSIX сумісних операційних системах. Інтерфейс користувача забезпечує просту і ефективну взаємодію користувача з програмним продуктом, а також передбачає можливість його адаптації до вимог користувача.

5.2 Умови виконання програми

Для роботи вимірювального комплексу необхідний IBM PC або ARM - комп'ютер не нижче наступної конфігурації:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 50 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша;
- мережева карта, не менше 100Mbps.

5.3 Виконання програми

Для установки програмного забезпечення необхідно скопіювати установочний архів, розпакувати його і виконати `install.sh` з правами супер користувача. При виконанні скрипта будуть:

- створені скрипти сервісів в папці `/etc/init.d`;
- додані сервісні скрипти в автозавантаження;

– виконані копіювання бінарних файлів в /usr/local/bin і конфігураційних в /etc;

- згенерована конфігурація для системного планувальника cron;
- копійовані конфігураційні файли stunnel и 3proxy;
- встановлена система скидання в початковий стан;
- створено деінсталлятор /usr/local/bin/uninstall.sh;
- створені супервізори;
- створений скрипт для оновлення ПЗ /usr/local/bin/update.sh.

Для запуску вимірювальної програми необхідно виконати команду /etc/init.d/slamond start. Для запуску конфігураційної програми необхідно виконати /etc/init.d/wiprobe start. Для скидання налаштувань в початковий стан необхідно утримувати кнопку перезавантаження пристрою більш ніж 15 секунд.

Програмне забезпечення має такі конфігураційні файли:

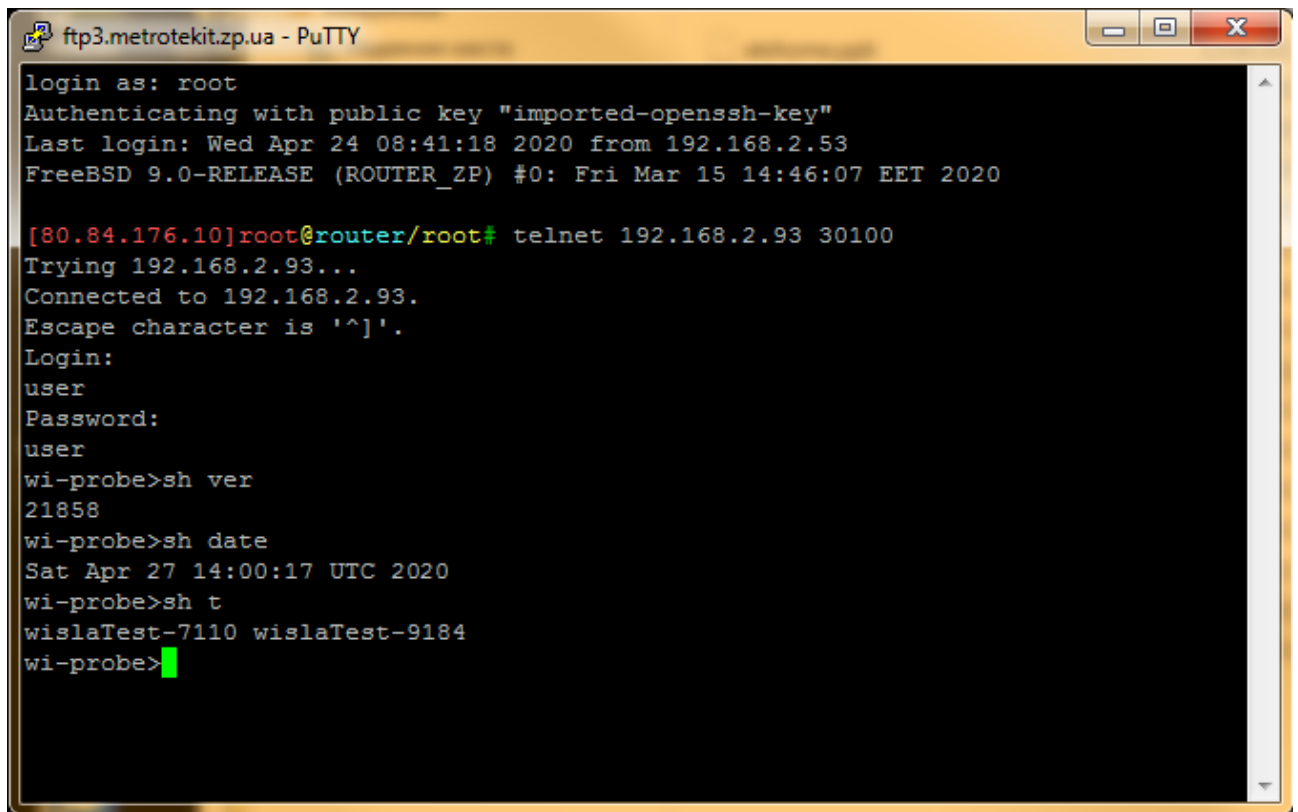
- /etc/wiprobe.conf – конфігураційний файл керуючої програми;
- /etc/salmon.conf – конфігураційний файл вимірювальної програми;
- /etc/cron.d/wiprobe.cron – містить дані про автоматичне оновлення і супервізор системи.

Програма wiprobe генерує або налаштовує такі файли:

- /etc/stunnel/stunnel.conf – змінює інформацію про проху сервері;
- /usr/local/etc/3proxy/3proxy.cfg – вимірює режим проксування;
- /etc/networking/interfaces – генерує мережеві інтерфейси використовуючи /etc/wiprobe.conf;
- /etc/wgetrc – конфігурує використання проксі сервера;
- /etc/ntp.conf – змінює список серверів за якими буде синхронізуватися час;
- /etc/ptp.conf – змінює режим синхронізації.

5.4 Приклад роботи з програмою

На рисунку 5.1 представлено приклад авторизації, перевірки часу, версії і списку тестів, використовуючи telnet клієнт для доступу до управління пристроєм.



```
ftp3.metrotekit.zp.ua - PuTTY
login as: root
Authenticating with public key "imported-openssh-key"
Last login: Wed Apr 24 08:41:18 2020 from 192.168.2.53
FreeBSD 9.0-RELEASE (ROUTER_ZP) #0: Fri Mar 15 14:46:07 EET 2020

[80.84.176.10]root@router/root# telnet 192.168.2.93 30100
Trying 192.168.2.93...
Connected to 192.168.2.93.
Escape character is '^]'.
Login:
user
Password:
user
wi-probe>sh ver
21858
wi-probe>sh date
Sat Apr 27 14:00:17 UTC 2020
wi-probe>sh t
wislaTest-7110 wislaTest-9184
wi-probe>
```

Рисунок 5.1 – Робота з використанням telnet клієнта Putty

Список допустимих команд і параметрів наведено у таблицях 5.1 – 5.5.

Таблиця 5.1 – Список команд головного інформаційного режиму

Команда	Призначення			
exit	Вихід з telnet			
whoami	Виведення інформації про поточного користувача			
restart	Рестарт telnet			
reboot	Перезавантаження пристрою			
kill {id}	Завершити сесію			
configure (cfg)	Вхід в режим конфігурації			
sessions (s)	Перегляд сесій			
date	Виведення поточної дати			
slamon	Виведення налаштувань і властивостей сламона			
services (ss)		Виведення статусу всіх сервісів		
	{name}	Виведення статусу зазначеного сервісу		
interface (i)		Виведення інтерфейсів у вигляді списку		
	-t	Виведення інтерфейсів у вигляді таблиці		
	{name}		Детальне виведення	
		-i	Виведення інтерфейсу з файлу /etc/network/interface	
		-s	Виведення інтерфейсу з ifconfig	
tests (t)		Виведення тестів у вигляді списку		
	{name}	Детальне виведення інформації про тест		
mode	Виведення інформації про поточний режим (master \ slave)			
system	Виведення системних налаштувань			
ntp-servers (ns)	Виведення ntp-серверів			
version (ver)	Виведення версії			
ftp-url	Виведення ftp url			

Таблиця 5.2 – Список команд головного конфігураційного режиму

Команда	Призначення	
interface (I) {name}	Вибір інтерфейсу для редагування і перехід в режим інтерфейсу	
test (t) {name}	Вибір тесту для редагування і перехід в режим тесту	
ntp-sync	Провести синхронізацію з встановленими серверами	
slamon	Перехід в режим редагування сламона	
save	Збереження параметрів і вихід в загальний режим	
exit	Вихід в загальний режим без застосування змін	
diff	Перегляд змін	
set {property} {value}	{property}	
	listen-port	Порт який слухає Telnet
	sessionTimeout	Таймаут для сесії в секундах
	maxThreads	Максимальна кількість одночасних сесій
	maxQueued	Максимальна кількість сесій, що очікують
	masterIp	Адреса майстер зонда
	date	Встановлення дати
	mode [ip]	Встановлення режиму (master \ slave)
	ftp-url {url}	Встановлення url для wiprobe-updater
	default-gateway (gw) {interface} {ip}	Встановлення шлюзу
passwd [login]	Зміна пароля собі або користувачеві з логіном login	
add {entity}	{entity}: test interface(i) service ntp-server(ns) user	
remove {entity}	{entity}: test interface(i) service ntp-server(ns) user	
start {entity}	{entity}: service	
stop {entity}	{entity}: service	
restart {entity}	{entity}: service	

Таблиця 5.3 – Список команд в додатковому конфігураційному режимі інтерфейсу

Команда	Призначення
show (sh)	Перегляд поточного інтерфейсу
add route	Додавання маршруту
remove route	Видалення маршруту
ip {value}	Встановлення ip
netmask {value}	Встановлення мережевої маски
dhcp {on off}	Встановлення dhcp
done	Вихід в конфігураційний режим

Таблиця 5.4 – Список команд в додатковому конфігураційному режимі вимірювальної програми

Команда	Призначення
name {value}	Встановлення імені
wisla-ip {value}	Встановлення адреси wile
master-login {value}	Встановлення логіна майстер-зонда
master-password {value}	Встановлення пароля майстер-зонда
master-port {value}	Встановлення порту майстер-зонда
autoreg-freq {value}	Встановлення частоти відправлення запиту автореєстрації
done	Вихід в конфігураційний режим

Таблиця 5.5 – Список команд в додатковому конфігураційному режимі вимірювального тесту

Команда	Призначення
show (sh)	Перегляд налаштувань поточного тесту
name {value}	Встановлення імені
dest {ip:port}	Встановлення адреси призначення
interval {value}	Як встановити періодичність (час агрегації в сек. (300))
period {value}	Як встановити інтервал в секундах (к-ть секунд для пакета (1))
tos {value}	Встановлення TOS
packet-size (pz) {value}	Встановлення розміру пакетів в байтах
done	Вихід в конфігураційний режим

6 ЕКОНОМІКО-ОРГАНІЗАЦІЙНА ЧАСТИНА

У дипломній кваліфікаційній роботі магістра досліджено та реалізовано програмне забезпечення для вимірювання характеристик передавання даних.

Очікується, що впровадження в експлуатацію розробленого програмного забезпечення на підприємстві дозволить скоротити 0,5 ставки одного системного адміністратора з заробітною платою 14 тис. гривень на місяць.

У розробці беруть участь програміст із заробітною платою 16000 грн. на місяць і консультант з заробітною платою 12000 грн. на місяць. На розробку проєкту було витрачено 2 місяці. Розробка програми розпочалася першого вересня і повинна бути виконана до кінця жовтня 2020 року.

6.1 Планування розробки програмного продукту

Весь комплекс розробки програмного продукту можна поділити на етапи [23]. Для кожного етапу вказуються трудомісткість, кількість виконавців і тривалість робіт. Інформація з характеристики робіт з розробки програмного забезпечення зведена у таблицю 6.1.

Таблиця 6.1 – Характеристика робіт з розробки програми

Найменування робіт	Трудомісткість		Виконавці	Тривалість, днів
	люд-дн	%		
Аналіз ПЗ	2	3	Програміст Консультант	1 1
Визначення вимог до ПП	16	23,9	Програміст Консультант	8 8
Проектування структури програми	26	38,8	Програміст Консультант	13 13
Розробка схеми функціонування програми	3	4,5	Програміст	3
Створення програмного коду	9	13,4	Програміст	9
Тестування і налаштування програмного продукту	4	5,97	Програміст	4
Складання програмної документації	7	10,4	Програміст	7
Разом	67	100	Програміст Консультант	45 22

6.2 Визначення витрат на розробку програми

Для визначення витрат на розробку програми складають калькуляцію кошторисної вартості робіт, що включає статті:

- основна заробітна плата;
- додаткова заробітна плата;
- відрахування на єдиний соціальний внесок (ЄСВ);
- матеріали та комплектуючі вироби;
- витрати на спеціальне обладнання (на оплату машинного часу ЕОМ для написання і налаштування програмних засобів);
- накладні витрати.

Витрати за статтею «Основна заробітна плата» складаються з планового фонду зарплати всіх категорій працівників, зайнятих з розробки програми. Розрахунок зарплати на підставі даних про трудомісткість, представлений в таблиці 6.2.

Таблиця 6.2 – Основна заробітна плата

Посада виконавця	Чисельність, чол	місячний оклад	Кількість місяців роботи	Сума ЗП, грн
Програміст	1	16000	2	32000
Консультант	1	12000	1	12000
Разом	2			44000

Додаткову заробітну плату приймають рівною 10% від основної зарплати. Вона розраховується за формулою (6.1):

$$ЗП_{\text{дод}} = ЗП_{\text{осн}} \cdot 10\%. \quad (6.1)$$

Підставивши величину основної заробітної плати в дану формулу отримаємо: $ЗП_{\text{дод}} = 44000 \cdot 0.1 = 4400$ грн.

Відрахування на єдиний соціальний внесок (ЄСВ) визначають у відсотковому відношенні від сум основної та додаткової зарплати з урахуванням премій і доплат. Відрахування вираховуються за формулою (6.2):

$$ОТ = (ЗП_{\text{осн}} + ЗП_{\text{дод}}) \cdot 0,22. \quad (6.2)$$

$$ОТ = (44000 + 4400) \cdot 0,22 = 10648 \text{ грн.}$$

Використовується три найменування матеріалів: флеш-носій – 400 грн., картридж – 900 грн, папір – 200 грн. Витрати на матеріали розраховують за формулою (6.3):

$$M = \sum_{i=1}^n (C_i \cdot N_i \cdot (1 + K_{m.з.}) - C_{io} \cdot N_{io}), \quad (6.3)$$

де M – витрати на матеріали, покупні напівфабрикати і комплектуючі вироби, грн .;

$K_{m.з.}$ – коефіцієнт, що враховує транспортно-заготівельні витрати;

C_i – ціна і-го найменування матеріалу, напівфабрикату і комплектуючого, грн .;

N_i – потреба в і-му матеріалі, напівфабрикаті і комплектуючому;

C_{io} – ціна зворотних відходів і-го найменування матеріалу, грн;

N_{io} – кількість зворотних відходів і-го найменування;

n – кількість найменувань матеріалів, напівфабрикатів і комплектуючих.

$$C_{io} = 0; \quad N_{io} = 0; \quad K_{m.з.} = 0,05;$$

$$M = (1 + 0,05) \cdot (400 + 900 + 200) + 1575 \text{ грн.}$$

Таким чином, разом витрати на матеріали та комплектуючі вироби становлять 1575 гривень.

У статтю «Витрати на спеціальне обладнання» входять витрати на придбання, транспортування, монтаж і налаштування нестандартного обладнання. Практично, в даному випадку, в цій статті враховуються витрати на оплату машинного часу ЕОМ для написання і налаштування програмних засобів. Для цього необхідно скласти кошторис «витрат на утримання і експлуатацію устаткування», виходячи з якого визначиться вартість одного машино-години роботи ПК, після множення якої на машинний час пішло на написання і налаштування програми отримаємо витрати на оплату машинного часу.

Амортизаційні відрахування визначають за формулою (6.4):

$$A = BB \cdot \frac{H_a}{100}, \quad (6.4)$$

де BV – балансова вартість обчислювальної техніки, грн;

H_a – норма амортизаційних відрахувань на повне оновлення обчислювальної техніки, %. Приймаємо, що техніка повинна працювати 4 роки, тому $H_a = 25\%$.

Балансова вартість обчислювальної техніки становить 36000 грн. Тому амортизаційні відрахування становлять:

$$A = 36000 \cdot 0,25 = 9000 \text{ грн}$$

Статтю "Експлуатація обладнання" розраховують підсумовуванням витрат на силову електроенергію і допоміжні матеріали. Витрати на електроенергію визначають за формулою (6.5):

$$C_{\varepsilon} = N_n \cdot \Phi_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot K_{\varepsilon\phi} \cdot C_{\varepsilon}, \quad (6.5)$$

де N_n – номінальна потужність ЕОМ, кВт;

$\Phi_{\varepsilon\phi}$ – річний ефективний фонд часу роботи ЕОМ, машино-год;

$K_{\varepsilon\phi}$ – середній коефіцієнт завантаження за часом;

$K_{\varepsilon\phi}$ – коефіцієнт завантаження по потужності;

C_{ε} – ціна одного кВт · год електроенергії, грн ./ (кВт · год).

Номінальна Потужність ЕОМ – 0,2 кВт. Річний ефективний фонд часу роботи ЕОМ ставити 1800 годин. Середні коефіцієнти завантаження за часом та за потужністю рівні відповідно 0,9 та 0,6. Ціна однієї кіловат-години електроенергії ставить 2,11 грн. Отримуємо:

$$C_{\varepsilon} = 0,2 \cdot 1800 \cdot 0,9 \cdot 0,6 \cdot 2,11 = 410,18 \text{ грн.}$$

Приймаємо, що $C_{\varepsilon} = 410$ грн.

Витрати за статтею "Заробітна плата обслуговуючих робітників та відрахування на соціальне страхування в інші фонди" визначають за формулою (6.6):

$$ЗП_{обсл.} = \frac{\Phi ЗП \cdot (1 + K_{відрах.}) \cdot \frac{t_{обсл.}}{\Phi_{эф.обсл.}}}{(6.6)}$$

де $\Phi ЗП_z$ – річний фонд заробітної плати (основної та додаткової) обслуговуючих робітників, грн.;

$K_{відрах}$ – коефіцієнт, що враховує відрахування на соціальне страхування і в інші фонди;

$t_{обсл}$ – час протягом року, необхідне технічне обслуговування ЕОМ, годин / рік.

Приймаємо $t_{обсл} = 24$ год.;

$\Phi_{эф.оюсл}$ – річний ефективний фонд часу обслуговуючого персоналу, годин / рік.

Приймаємо, що річний ефективний фонд часу обслуговуючого персоналу становить 1800 годин

Заробітна плата обслуговуючого персоналу = 10500 грн/місяць.

$$\Phi ЗП_z = 10500 \cdot 12 = 126000 \text{ грн.}$$

$$ЗП_{обсл.} = 126000 \cdot (1 + 0,22) \cdot 24 / 1800 = 2050 \text{ грн.}$$

Суму витрат за статтею "Поточний ремонт обладнання" обчислимо як 3% від балансової вартості обладнання, оскільки неможливо заздалегідь визначити, скільки коштів необхідно витратити на придбання запасних частин наявного обладнання. Балансова вартість обчислювальної техніки становить 36000 грн. Тому сума витрат за статтею "Поточний ремонт обладнання" становить: $36000 \cdot 0,03 = 1080$ грн.

Інших витрат за проектом не передбачено, тому сума витрат за цією статтею дорівнює нуль гривень.

Кошторис витрат на утримання і експлуатацію устаткування наведений в таблиці 6.3.

Таблиця 6.3 – Кошторис витрат на утримання і експлуатацію устаткування

Найменування статей витрат	Сума, грн.
Амортизація обладнання	9000
Заробітна плата основна і додаткова обслуговуючих робітників з відрахуваннями на соціальні заходи	2050
Експлуатація обладнання (крім витрат на поточний ремонт)	410
Поточний ремонт обладнання	1080
Інші витрати	0
Разом	12540

Витрати на оплату машинного часу ЕОМ для написання і налаштування програмних засобів визначаються за формулою (6.7):

$$C_{mo} = P_{екс} \cdot t_{mo}, \quad (6.7)$$

де C_{mo} – витрати на оплату машинного часу, грн;

$P_{екс}$ – експлуатаційні витрати на одну годину машинного часу цієї цифрової ЕОМ, грн. / машино–год.;

t_{mo} – машинний час цифрової ЕОМ для написання і налаштування даного програмного продукту, машино–год.

$$P_{екс} = 12540/1800 = 6,97 \text{ грн. / машино–год.}$$

Для написання і налаштування даного програмного продукту ЕОМ експлуатується протягом 45 днів в одну зміну по 8 годин, що складає в сумі 360

годин. Таким чином отримуємо суму за статтею «Витрати на спеціальне обладнання»:

$$C_{\text{мо}} = 6,97 \cdot 360 = 2510 \text{ грн.}$$

До накладних витрат відносяться витрати на загальне управління і загальногосподарські потреби (заробітна плата апарату управління, канцелярські витрати і т.д.), утримання та експлуатацію будівель. Накладні витрати включаються до вартості розробки програми непрямым шляхом – у відсотках до основної заробітної плати розробників. Для обчислення цієї статті видатків у дипломному проєкті приймемо, що накладні видатки становлять 33% від основної заробітної плати розробників, що складає 44000 грн. Тому сума накладних видатків становить: $44000 \cdot 0,33 = 14520$ грн.

Калькуляція кошторисної вартості робіт з розробки програми наведена в таблиці 6.4.

Таблиця 6.4 – Калькуляція кошторисної вартості робіт з розробки програми

Найменування статей витрат	Сума, грн.
Основна заробітна плата	44000
Додаткова заробітна плата	4400
ЄСВ	10648
Матеріали і комплектуючі	1575
Витрати на спеціальне обладнання	2510
Накладні витрати	14520
Разом	77653

6.3 Розрахунок економічної ефективності програмного продукту

Річну економію від впровадження програми, що дозволяє знизити витрати машинного часу ЕОМ для рішення певного завдання в порівнянні із програмою, що застосовувалася раніше, визначають за формулою:

$$E = (T_{м1} - T_{м2}) \cdot B_{екс.ктз} + E_n^a \cdot \frac{\Phi_{б.ктз} \cdot (T_{м1} - T_{м2})}{\Phi_{эф.ктз}} - \frac{S_{рп}}{T_c} + E_{зп}, \quad (6.8)$$

де $T_{м1}, T_{м2}$ – машинний час, необхідний для розв’язання поставлених завдань, відповідно в старому та у новому варіантах, машино–год./рік;

$B_{екс}$ – експлуатаційні витрати, що доводяться на 1 год машинного часу ЕОМ, грн/ машино–год.;

E_n^a – нормативний коефіцієнт ефективності капітальних вкладень у засоби автоматизації;

$\Phi_{б.ктз}$ – балансова вартість ЕОМ (КТЗ), грн.;

$\Phi_{эф.ктз}$ – річний ефективний фонд часу роботи ЕОМ, машино–год.;

$S_{рп}$ – сумарні витрати на розробку програми, грн.;

T_c – термін служби впроваджуваної програми до її морального зношування, років. Приймаємо цей термін 5 років;

$E_{зп}$ – економія на заробітній платі.

До впровадження даного ПЗ дані функції виконувались співробітником, що відповідає за обробку рукописних символів, 1800 годин на рік, а після 900 годин на рік (економія складає 0,5 ставки одного системного адміністратора з заробітною платою 14 тис. гривень на місяць).

$$E = (1800 - 900) \cdot 6,97 + 0,2 \cdot (36000 \cdot (1800 - 900)) / 1800 - 77653/5 + \\ + 14000 \cdot 0,5(1 + 0,22) \cdot 12 = 96823 \text{ грн.}$$

Коефіцієнт ефективності капітальних вкладень E и строк їхньої окупності $T_{ок}$ розраховуються за формулами (6.9) і (6.10):

$$Ef = \frac{E}{S_{pn}}, \quad (6.9)$$

$$T_{ок} = \frac{S_{pn}}{E}, \quad (6.10)$$

$$Ef = 96823 / 77653 = 1,25,$$

$$T_{ок} = 77653 / 96823 = 0,8 \text{ року.}$$

Зведемо отримані техніко–економічні показники в таблицю 6.5.

Таблиця 6.5 – Техніко–економічні показники

Показники	Сума	Одиниця виміру
Витрати на розробку програми	77653	грн.
Річна економія	96823	грн.
Строк окупності програми	0,8	рік

Таким чином, впровадження в експлуатацію розробленого програмного забезпечення на виробничому підприємстві дозволить скоротити 0,5 ставки одного системного адміністратора з заробітною платою 14 тис. гривень на місяць. Економічні розрахунки показали, що строк окупності програми 0,8 року, що значно менше часу її нормативного строку окупності (5 років). Крім того, річна економія при використанні програми становить 96823 грн. Виходячи з виконаних розрахунків, можна зробити висновок, що розробка даного програмного продукту є економічно доцільною.

7 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

7.1 Аналіз потенційних небезпек

При розробці дипломного проекту на розробника, працюючого за ПК, постійно або періодично діють такі небезпечні й шкідливі фактори:

- невідповідність нормам параметрів мікроклімату (температури повітря, відносної вологості, швидкості руху повітря) внаслідок низької/високої температури джерела опалення/охолодження, відсутності або надлишкового провітрювання приміщення, що може призвести до захворювань загального характеру та підвищеної втомлюваності;

- підвищений рівень шуму внаслідок використання шумлячого устаткування, може призвести до втомлюваності, порушень сконцентрованості, дратівливості, головного болю, запаморочення, зниження пам'яті;

- підвищений рівень статичної електроенергії при неправильному проектуванні робочої зони може призвести до електричних травм;

- недостатній рівень штучного освітлення робочої зони внаслідок неправильної організації системи освітлення приміщення або використання несправних освітлювальних приладів може призвести до порушення роботи органів зору людини;

- підвищений рівень природного освітлення може виникнути при неправильній орієнтації вікон в робочому приміщенні або відсутності захисних штор на вікнах, що може ускладнити роботу з монітором та погіршити зір;

- підвищений рівень електромагнітного випромінювання, внаслідок використання застарілого та неправильно працюючого обладнання (працююча ЕОМ), може призвести до втоми, нудоти, головного болю; випромінювання може впливати на нервову та серцево-судинну системи людини;

- широкий спектр випромінювання від дисплея, в основному з електронно-променевою трубкою, який включає рентгенівську,

ультрафіолетову та інфрачервону області, а також широкий діапазон електромагнітних випромінювань інших частот може порушити роботу внутрішніх органів людини;

- підвищена напруга електромережі, замикання якої може виникнути через тіло людини;

- висока іонізація повітря, внаслідок використання принтерів, моніторів або іншого обладнання, може призвести, наприклад, до накопичення пилу в органах дихання людини;

- небезпека виникнення пожежі внаслідок короткого замикання в електронних пристроях, перегріві апаратури, несправній ізоляції може призвести до опіків шкіри та порушень здоров'я людини.

Психофізіологічні небезпечні та шкідливі виробничі фактори:

- статичні перевантаження внаслідок того що робоче місце не відповідає вимогам ергономіки;

- емоційні перевантаження внаслідок підвищеної відповідальності, неякісного програмного забезпечення, дефіциту часу;

- розумове перенапруження внаслідок наявності складних задач, алгоритмів і т. п.;

- перенапруження зорового аналізатора внаслідок відблисків на екрані, недотримання правил та режимів праці;

- монотонність праці може виникнути внаслідок багаторазового повторення великої кількості однакових операцій [24].

7.2 Заходи щодо забезпечення безпеки

Обчислювальні центри відносяться до приміщень без підвищеної небезпеки відповідно до ПУЕ [24]. При проектуванні систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та споруд для комп'ютерної техніки необхідно дотримуватись вимог ПУЕ,

ДСТУ 7237:2011 «Система стандартів безпеки праці. Електробезпека. Загальні вимоги та номенклатура видів захисту» [25].

Комп'ютерна техніка, периферійні пристрої ПК та устаткування для обслуговування, ремонту та налагодження ПК, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники і т.д.), електропроводка та кабелі за виконанням та ступенем захисту відповідають класу зони згідно ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів.

Лінія електромережі для живлення комп'ютерної техніки, периферійних пристроїв ПК та устаткування для обслуговування, ремонту та налагодження ПК виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ПК, на помітному та доступному місці встановлено аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення.

Відеотермінали, комп'ютерна техніка, спеціальні периферійні пристрої ПК та устаткування для обслуговування, ремонту та налагодження ПК відповідають вимогам діючих в Україні стандартів, нормативних актів з охорони праці. Відеотермінали, комп'ютерна техніка, спеціальні периферійні пристрої ПК закордонного виробництва відповідають вимогам національних стандартів держав-виробників і мають відповідну позначку на корпусі, в паспорті або іншій експлуатаційній документації [26].

За способом захисту людини від ураження електричним струмом відеотермінали, комп'ютерна техніка, периферійні пристрої ПК та обладнання для обслуговування, ремонту та налагодження ПК відповідають I класу захисту згідно НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [27].

Неприпустимо використання клем функціонального заземлення для підключення захисного заземлення.

Виходячи з вимог щодо допустимих значень неіонізуючого електромагнітного випромінювання було встановлено, що щільність електричної напруги не перевищує:

- в діапазоні частот 5 кГц - 2 кГц - 25 В/м;
- в діапазоні частот 2 кГц - 400 кГц - 2,5 В/м;
- щільність магнітного потоку відповідно:
- в діапазоні частот 5 кГц - 2 кГц - 250 нТл;
- в діапазоні частот 2 кГц - 400 кГц - 25 нТл;

потужність дози рентгенівського випромінювання на відстані 5 см від екрану та інших поверхонь ВДТ не повинна перевищувати 100 мкР/год [26].

7.3 Заходи щодо виробничої санітарії та гігієни праці

Для підтримки нормованих значень концентрації негативних і позитивних іонів передбачена штучна іонізація. У приміщеннях з дисплеями проводиться щоденне вологе прибирання і регулярні провітрювання протягом робочої зміни. Видалення пилу з екранів дисплеїв проводиться не рідше 1 разу за зміну. У періоди зміни, коли робота не виконується, дисплеї вимкнені.

У залах з обчислювальною технікою і інших виробничих приміщеннях при виконанні робіт операторського типу, пов'язаних з нервово-емоційним напруженням, дотримуються оптимальних величин температури повітря 22-24°C при його відносній вологості 40-60% і швидкості руху повітря не більше 0,1 м/с відповідно до ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» [28].

У приміщенні запроектовані система опалення, системи кондиціонування повітря згідно з ДБН В.2.5-67:2013 «Опалення, вентиляція, кондиціонування» [26].

Згідно ДБН В.2.5-28:2018. «Природне і штучне освітлення» для штучного освітлення приміщень, в яких використовують дисплеї, запроектована система комбінованого освітлення. Передбачено пристрій робочого і евакуаційного освітлення [29].

Освітленість робочих місць в горизонтальній площині на рівні 0,8 м від підлоги – не менше 400 лк. Вертикальна освітленість у площині екранів дисплеїв – не більше 300 лк. Коефіцієнт пульсації освітленості - не більше 10%. Показник дискомфорту – не більше 40. Для обмеження відбитих від екранів відблисків забезпечене зниження яскравості випромінювачів в кутовій зоні більш 55 град. від вертикалі не більше 20 град.

При улаштуванні місцевого освітлення застосовуються лампи на пересувних шарнірах. Світло падає по ліву сторону. Потужність ламп, яка рекомендується – не менше 40-70 Вт. Одне місцеве освітлення неприпустимо, оскільки пов'язано з неминучою адаптацією зору.

Нижче наведено розрахунок загального рівномірного штучного освітлення штучним світлом у обчислювальному центрі (ОЦ).

Приймаємо розміри приміщення: довжина $A = 10$ м, ширина $B = 7$ м, висота $H = 4$ м. Висота робочої поверхні складає $h = 0,7$ м. Коефіцієнти відбиття поверхонь приміщення: $\rho_c=50\%$; $\rho_{ст}=30\%$; $\rho_{п}=10\%$,

Згідно з ДБН В.2.5-28:2018 «Природне і штучне освітлення» рівень нормованого загального освітлення в приміщенні нормуються як $E_n = 300$ лк.

Обираємо світильник типу ЛВП, призначений для приміщень висотою до 4,5 м. $IP=54$ – ступінь захисту світильника від пилу та брызок води, $[L/h] = 1,3$ – числове значення коефіцієнта світильника.

Коефіцієнт запасу k_z враховує зниження рівня освітленості з часом в результаті забруднення та старіння ламп, світильників і поверхонь приміщення, приймається в залежності від виробничих умов. Приймаємо $k_z = 1,4$.

Коефіцієнт нерівномірності (мінімального) освітлення z (відношення середньої освітленості до мінімальної освітленості), для люмінесцентних ламп низького тиску, як правило дорівнює $z = 1,1$.

Розрахуємо кількість рядів світильників у приміщенні:

$$N_p = \frac{B}{(H - h_p) * [L / h]} = \frac{7}{(4 - 0,7) * 1,3} = 1,63 \approx 2 \text{ ряди світильників}$$

Визначимо максимально припустиму відстань між рядами світильників

$$L_{\max} = \frac{B}{N_p},$$

де B – ширина приміщення, N_p – кількість рядів

$$L_{\max} = \frac{7}{2} = 3,5 \text{ м}$$

Розраховуємо висоту підвісу світильника над робочою поверхнею

$$h = \frac{L_{\max}}{[L / h]}$$

$$h = \frac{3,5}{1,3} = 2,7 \text{ м}$$

Знайдемо висоту звисання світильника від стелі

$$h_3 = H - h_p - h$$

де H – висота приміщення, h_p – висота робочої поверхні, h – висота підвісу світильника над робочою поверхнею.

$$h_3 = 4 - 0,7 - 2,7 = 0,6 \text{ м}$$

Чисельне значення індексу приміщення визначаємо за рівнянням:

$$i = \frac{AB}{h(A+B)}$$

де A – довжина приміщення, $м$;

B – ширина приміщення, $м$;

h – висота розміщення світильників над робочою поверхнею, $м$

$$i = \frac{10 \cdot 7}{2,7 \cdot (10 + 7)} = 1,53$$

Значення коефіцієнта використання світлового потоку η вибирається в залежності від виду джерела світла, типу обраного світильника, коефіцієнтів відбиття поверхонь приміщення (ρ_c , $\rho_{ст}$, $\rho_{п}$) та індексу приміщення. В нашому випадку $\eta = 46\%$.

Визначаємо сумарний світловий потік освітлювальної установки у даному виробничому приміщенні:

$$\Phi_{\Sigma} = \frac{E_n \cdot S \cdot k_3 \cdot z}{\eta},$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, $лм$;

E_n – нормоване значення освітленості, $лк$;

S – площа освітлюваної поверхні, $м^2$;

k_3 – коефіцієнт запасу;

z – коефіцієнт нерівномірності (мінімальної) освітленості;

η – коефіцієнт використання світлового потоку

$$\Phi_{\Sigma} = \frac{300 * 10 * 7 * 1.4 * 1.1}{0.46} = 70304 \text{ лм}$$

Визначаємо умовну загальну кількість світильників у приміщенні:

$$N^* = AB/L_{\max}^2$$

$$N^* = \frac{10 * 7}{3.5 * 3.5} = 5.7 \approx 6 \text{ шт.}$$

Кратне числу рядів $N_p = 2$.

Розрахуємо світловий потік умовного джерела світла:

$$\Phi_{\text{л}}^* = \frac{\Phi_{\Sigma}}{N_{\text{л}}}, \text{ лм}$$

де Φ_{Σ} – розрахункове значення сумарного світлового потоку у приміщенні, лм;

$N_{\text{л}}$ – загальна кількість ламп у приміщенні, шт;

$$N_{\text{л}} = N^* \cdot n$$

n – кількість ламп у світильнику (в ЛВП світильнику знаходиться 2 лампи)

$$N_{\text{л}} = 6 * 2 = 12 \text{ шт.}$$

Тоді,

$$\Phi_{\text{л}} = \frac{70304}{12} = 5859 \text{ лм}$$

Обираємо тип стандартної лампи з найближчим значенням фактичного світлового потоку лампи $\Phi_{\text{л}}$ і знайдемо коефіцієнт m (співвідношення між розрахунковим світловим потоком лампи $\Phi_{\text{л}}^*$ та фактичним світловим потоком вибраної стандартної лампи $\Phi_{\text{л}}$):

Обираємо лампу ЛБ із $P_{\text{л}}=80$, $\Phi_{\text{л}} = 5400$ лм

$$m = \frac{\Phi_{\text{л}}^*}{\Phi_{\text{л}}}$$

$$m = \frac{5859}{5400} = 1,085$$

Визначаємо оптимальну (фактичну) кількість світильників у приміщенні:

$$N = N^* \cdot m$$

$$N = 6 \cdot 1,085 \approx 6 \text{ шт.}$$

Фактична кількість ламп у приміщенні:

$$N_{\text{фл}} = N_{\text{ф}} \cdot n$$

$$N_{\text{фл}} = 6 \cdot 2 = 12 \text{ шт.}$$

Визначаємо загальну розрахункову освітленість E_p у приміщенні, що створюється при застосуванні стандартних ламп:

$$E_p = \frac{\Phi_{\text{л}} N_{\text{фл}} \eta}{S k_z z}, \text{ лк}$$

$$E_p = \frac{5400 \cdot 12 \cdot 0,46}{10 \cdot 7 \cdot 1,4 \cdot 1,1} = 278 \text{ лк}$$

Тобто виконується умова $E_p = (-10\% \dots +20\%) \cdot E_{\text{н}}, \text{ лк}$

Робочі місця програмістів обчислювальних машин не мають граничних рівнів звуку вище 50дБА, а в приміщеннях для розміщення гучних агрегатів обчислювальних машин вище 75дБА. При виконанні робіт з вироблення концепцій нових програм рівні звуку не перевищують 40дБА; при виконанні операторської роботи – 65дБА згідно ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку [26].

НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин» [27] регламентує вимоги до організації робочого місця користувача ВДТ. Найкраще розмістити робочі місця з ВДТ рядами, причому відносно вікон вони розміщуються так, щоб природне світло падало збоку, переважно зліва. Це дасть змогу виключити дзеркальне відбиття на екрані джерел природного світла та потрапляння останніх в поле зору користувачів.

Площа, виділена для одного робочого місця з ПК, складає не менше 6 м², а об'єм – не менше 20 м³. При розміщенні робочих місць дотримано наступні вимоги:

- робочі місця з ВДТ розміщені на відстані не менше 1,2 м від стін зі світловими прорізами;
- відстань між бічною поверхнею одного з відеотерміналів не менше за 1,2 м;
- відстань від екрана дисплея до оператора (при розмірі екрану по діагоналі): 14 " -15 " – 600-700 мм; 17 ' – 700-800 мм; 19 ' – 800-900 мм; 21 " – 900 -1000 мм.
- відстань між тильною поверхнею одного відео термінала та екраном іншого не менше 2,5 м;
- прохід між рядами робочих місць не менше 1 м.

До виробничих меблів на робочому місці з ВДТ належить робочий стіл, стілець (крісло), підставка для ніг.

Вимоги до конструкції робочого столу, стільця, підставки для ніг на робочих місцях з ВДТ визначаються НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин».

Конструкція робочого столу відповідає сучасним вимогам ергономіки і забезпечує оптимальне розміщення на робочій поверхні всього обладнання та пристосувань, що використовуються, з урахуванням їх розмірів та конструктивних особливостей. Висота робочої поверхні столу для ВДТ - в межах 680 – 800 мм, а ширина і глибина – забезпечує можливість виконання операцій у зоні досяжності моторного поля. Рекомендовані розміри столу: висота – 725 мм, ширина – 600 – 1400 мм, глибина – 800 – 1000 мм. Робочий стіл має простір для ніг висотою не менше 600 мм, шириною не менше 500 мм, глибиною на рівні колін не менше 450 мм, на рівні витягнутої ноги – не менше 650 мм.

Робочий стіл для ВДТ має підставку для ніг шириною не менше 300 мм та глибиною не менше 400 мм, з можливістю регулювання по висоті в межах 150 мм та кута нахилу опорної поверхні – в межах 20°. Підставка має рифлену поверхню та бортик на передньому краї заввишки 10 мм.

Ширина та глибина сидіння є не меншими за 400 мм. Висота поверхні сидіння має регулюватися в межах від 400 – 500 мм, а кут нахилу поверхні – від 15 град. вперед до 5 град. назад. Поверхня сидіння має бути плоскою, передній край – закругленим.

Конструкція виробничих меблів для користувача ВДТ має бути такою, щоб забезпечувати йому підтримання оптимальної робочої пози з такими ергономічними характеристиками: ступні ніг – на підлозі або на підставці для ніг, стегна на – горизонтальній площині, верхні (плечові) частини рук – вертикальні; кут ліктьового суглоба (між плечем та передпліччям) – 70 – 90 град., зап'ястки зігнуті під кутом не більше 20 град. відносно горизонтальної площини, нахил голови вперед в межах 15 – 20 град.

Співвідношення яскравості екрана і найближчих поверхонь відображення не перевищує 5:1. Покриття столу матове з коефіцієнтом відображення 0,25-0,4.

Приміщення, оснащені дисплеями, розташовуються в північній або північно-східній частині будинку.

У приміщеннях, обладнаних відеотерміналами, стіни пофарбовані фарбами пастельних тонів з коефіцієнтом відображення 40-60%. Забарвленням поверхонь додана матова фактура.

У разі виникнення надмірного зорового навантаження, що супроводжується погіршенням самопочуття, незважаючи на хорошу ергономічну організацію роботи і дотримання режимних моментів, проводиться індивідуальний підхід в обмеженні часу роботи, корекція тривалості перерв для відпочинку або зміна діяльності, не пов'язаної з використанням комп'ютеру [26].

Для підтримання високого рівня працездатності та збереження здоров'я працівників щоденно виконується комплекс нескладних фізичних вправ, що сприяють функціональному стану організму людини.

7.4 Заходи з пожежної безпеки

Заходи з протипожежної безпеки розроблені для виробничого приміщення обладнаного ПК з ВДТ розроблені згідно вимог НАПБ А.01.001-2014 «Правила пожежної безпеки в Україні» [30].

Виходячи з аналізу речовин та матеріалів, які використовуються при роботі у приміщенні, обладнаному ПК з ВДТ:

- згідно ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [31] у приміщенні, обладнаному ПК з ВДТ можлива пожежа класу – А та Е; воно належить до категорії «Д» з пожежної небезпеки – простір у приміщенні, у якому перебувають тверді горючі речовини та матеріали.

Оскільки приміщення належить до категорії «Д» з пожежної небезпеки, тому відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» [31] воно має II ступінь вогнестійкості.

У разі виникнення пожежі у приміщенні обладнаному ПК з ВДТ для евакуації персоналу відповідно до вимог ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва. Загальні вимоги» передбачені виходи, по обидві сторони приміщення, з одного боку вікно (на пожежні сходи), а з іншого – вхідні двері. Згідно СНиП 2.09.02-85* «Производственные здания», відстань від найбільш віддаленого робочого місця до найближчого евакуаційного виходу не обмежується.

Обладнання, силові та освітленні мережі приміщення, обладнаного ПК з ВДТ відповідають вимогам пожежної безпеки, оскільки виконані відповідно до вимог НПА ОП 40.1-1.32-01 «Правила будови електроустановок. Електрообладнання спеціальних установок» [32], та мають ступінь захисту ізоляції обладнання IP44 яка відповідає класу пожежанебезпечної зони П-Па до якої належить приміщення.

Для виявлення початкової стадії загоряння, швидкого й точного оповіщення служби пожежної охорони про час і місце виникнення пожежі встановлені системи автоматичної пожежної сигналізації (АПС). Вони можуть самостійно пускати в хід установки пожежогасіння, коли пожежа не досягла великих розмірів. Системи АПС складаються з пожежних сповіщувачів, ліній зв'язку й прийомних пультів (станцій). Згідно вимог

ДБН В.2.5-56:2014 «Системи протипожежного захисту», в приміщенні обладнаному ПК з ВДТ встановлена система пожежної й охоронної сигналізації «Фотон-ОФІС», яка забезпечує виявлення теплових і димових ознак пожежі і місця виникнення пожежі з точністю до місця розміщення датчика [26].

Оскільки приміщення що обладнане ПК з ВДТ має площу 70 м², тому відповідно до вимог п. 3.8 розділу «Типові норми належності вогнегасників» ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників. Загальні технічні вимоги» [33] для гасіння електроустановок, що знаходяться під напругою, передбачені вуглекислотні вогнегасники типу ВВК-3,5 у кількості

4 штуки (з розрахунку один вогнегасник с величиною заряду вогнегасної

речовини 3 кг. і більше, на 20 м² площі приміщення). Відстань між вогнегасниками та місцями можливих загорянь не перевищує 10 м.

7.5 Заходи щодо забезпечення безпеки в надзвичайних ситуаціях

Центральні та місцеві органи виконавчої влади повинні надавати населенню оперативну і достовірну інформацію про стан захисту населення і території від НС техногенного і природного характеру, про виникнення НС, методи та способи захисту, про вжиті заходи щодо забезпечення захисту.

Оповіщення про загрозу виникнення НС і постійне інформування населення про перебіг подій і зміну обстановки здійснюється за допомогою завчасно створених загальнодержавної і територіальних автоматизованих систем центрального оповіщення, систем оповіщення на об'єктах господарювання, локальних систем оповіщення в зонах можливого катастрофічного затоплення, у районах розміщення радіаційно- і хімічно небезпечних підприємств, інших об'єктів підвищеної небезпеки. У системі оповіщення використовуються технічні засоби загальнодержавної і галузевих систем зв'язку, радіо- і телемереж та ін. засобів передавання інформації.

Забезпечення дій органів управління сил ЦО та з НС та виконання основних заходів ЦО в період загрози виникнення надзвичайних ситуацій виконується у наступному порядку:

- приводяться у готовність і переводяться на цілодобове спостереження пости РХС з доповіддю про інформацію: терміново у разі необхідності; через 4 год.;
- штаб цивільної оборони та з надзвичайних ситуацій об'єкта координує роботу управлінь, відділів і служб об'єкта щодо матеріально-технічного забезпечення сил та заходів цивільної оборони;
- приводяться у готовність всі наявні сховища, протирадіаційні укриття та підвальні приміщення для укриття працівників і службовців, у разі необхідності проводиться додатково їх герметизація;

- виконується прогноз можливої постановки і уточнюються рішення на проведення рятувальних та першочергових невідкладних відновлювальних робіт в можливих осередках ураження;

- проводиться у разі необхідності відселених (або евакуація) працівників і службовців суб'єкта господарської діяльності у разі попадання об'єкта в зони ураження, особливо із зон хімічного і радіоактивного зараження;

- здійснюється (у разі необхідності) висування додаткових сил у райони аварій, катастроф і стихійного лиха, виконуються інші заходи.

У період загрози виникнення надзвичайних ситуацій системи оповіщення приводяться у готовність для здійснення масового інформування працівників, а зв'язок здійснюється з використанням всіх його видів і засобів.

Особливо важливого значення набуває зв'язок в надзвичайних умовах мирного часу, коли виникають великі промислові аварії, катастрофи, стихійні лиха, коли в лічені хвилини необхідно віддати розпорядження на рятування людей, на приведення в готовність сил ЦО, організацію взаємодії та організувати оповіщення населення про загрозу виникнення або виникнення надзвичайних ситуацій.

Зв'язок організовується згідно з рішенням начальника цивільної оборони, вказівками начальника управління (відділу) з питань надзвичайних ситуацій та цивільного захисту населення області (міста обласного підпорядкування, району) і розпорядженням вищого органу управління. Відповідальність за організацію зв'язку несуть начальники управлінь (відділів) з надзвичайних ситуацій і цивільного захисту населення, штабів ЦО та з НС, а за її підготовку і безперервну роботу – відповідні керівники об'єктів зв'язку.

У системах централізованого оповіщення можуть використовуватися апаратура і технічні засоби оповіщення ЦО, канали та засоби зв'язку, мережі радіомовлення і телебачення (канали звукового супроводження) центральних і місцевих органів виконавчої влади, підприємств, установ та організацій, мережі зв'язку яких входять до ЄНСЗ. Ці системи забезпечують оповіщення і подальше інформування:

- чергових служб міністерств та інших центральних органів виконавчої влади по службових телефонах;
- чергових служб місцевих органів виконавчої влади;
- чергових аварійно-рятувальних служб;
- сил цивільної оборони;
- населення, яке знаходиться в зоні можливого ураження [34].

ВИСНОВКИ

У ході виконання дипломної кваліфікаційної роботи магістра було розв'язано актуальне завдання дослідження та реалізації програмно-апаратних засобів вимірювання характеристик передавання даних. Було проведено огляд і аналіз засобів вимірювання характеристик передавання даних. Відібрані матеріали були систематизовані, проаналізовані, узагальнені і представлені в першому, другому і третьому розділах пояснювальної записки..

Складено технічне завдання, в якому зазначена мета розробки та її призначення, основні вимоги до програмно-апаратного комплексу, порядок його функціонування. Розроблено програмний продукт, що має структуру і виконує функції, необхідні для вимірювання характеристик передавання даних.

Наукова новизна роботи полягає у тому, що розроблено модель програмного забезпечення вимірювання характеристик передавання даних, яка характеризується об'єктами Timer, Bucket Manager, DataSender, Connector, Calculation, та дозволяє розробляти програмне забезпечення для вимірювання різних характеристик передавання даних.

Практична значущість отриманих результатів полягає у тому, що розроблено програмне забезпечення для вимірювання характеристик передавання даних.

Проведено декомпозицію керуючої програми відповідно до поставлених вимог. Програма була розділена на 2 підпрограми: та, що вимірює (головна програма) та конфігураційна програма. Кожна програма в свою чергу була розбита на логічні модулі. Розроблено уніфікований протокол, який дозволяє налаштувати програму, як за сожливостями зовнішньої системи, так і telnet сервером.

Визначено, що архітектура головної програми вимірювання характеристик передавання даних складається з таких модулів: частина конфігурацій; керуюча частина; модуль для взаємодії із зовнішніми системами; модуль з відсилання результатів. Імплементовано головний вимірювальний

додаток, який підтримує dns, icmp, udp, rfc2554 тести які можуть працювати одночасно у великій кількості. Реалізовано модуль Plug & Go який реєструє пристрій у зовнішній системі, зчитує сценарії з зовнішніх систем і виконує їх на пристрої, за коштами уніфікованого протоколу. Розроблено модуль з відправки і збереження готових даних, який дозволяє зберігати дані у вигляді csv файлів і передавати дані зовнішньої системі за допомогою http протоколу.

Запропоновано архітектуру конфігураційної програми. Розроблено програму, яка дозволяє створювати, видаляти, редагувати інтерфейси, взаємодіяти з операційною системою, перемикає режими пристрою і сервіси синхронізації часу, взаємодіяти з вимірювальною програмою (створювати, видаляти, редагувати тести). Реалізовано системні помічники, які дозволяють виробляти складні дії відповідаючи на питання системи. Опрацьована гнучка система команд, реалізований їх парсер і виконання.

Проведено функціональне і тестування навантаженням, що дозволило об'єктивно оцінити коефіцієнт похибки вимірювань, відповідність всім функціональним вимогам, а також споживання ресурсів пристрою при проведенні великої кількості одночасних тестів.

В результаті проведених функціонального тестування та тестування навантаженням було виявлено, що розроблений комплекс володіє необхідними вимогами визначеними в технічному завданні. Експериментально доведено, що розроблені алгоритми мають невеликий коефіцієнт похибки при роботі в завантажених лініях зв'язку. Тестування навантаженням довело, що вимірювальна програма при проведенні великої кількості тестів використовує не велику кількість ресурсів, залишаючи великий запас ресурсів для різноманітних призначених для користувача завдань (проведення додаткових тестів, виконання різного призначеного для користувача програмного забезпечення).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Как правильно составить SLA в [Электронный ресурс]. – Режим доступа: http://www.itsmonline.ru/phparticles/show_news_one.php?n_id=281.
2. ETX-203A Демаркационные устройства Carrier Ethernet [Электронный ресурс]. – Режим доступа: <http://www.rad.ru/10/21054/>.
3. Зонды симейства [Электронный ресурс]. – Режим доступа: <http://metrotek.ru/catalog/386/2639/>.
4. Сетевое программное обеспечение (IOS и NX-OS) [Электронный ресурс]. – Режим доступа: http://www.cisco.com/web/RU/products/cisco_ios.html#~three.
5. Стандарты IEEE 802.3 [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/IEEE_802.3.
6. SheevaPlug: мини-сервер в розетке за \$100 [Электронный ресурс]. – Режим доступа: http://www.3dnews.ru/news/sheevaplug_mini_server_v_rozetke_za_100.
7. Raspberry Pi Type B 512 Mb [Электронный ресурс]. – Режим доступа: <http://arduino.net.ua/shop/Raspberry%20Pi/Raspberry%20Pi%20kupit/>.
8. Компьютер Android 4.0 Mini PC МК802 – обзор малютки [Электронный ресурс]. – Режим доступа: http://www.ddriver.ru/kms_catalog+stat+cat_id-15+page-1+nums-452.html.
9. C++ [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/C++>.
10. Bash [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Bash>.
11. Java [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Java>.
12. Perl [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Perl>.

13. Сравнение языков программирования [Электронный ресурс]. – Режим доступа: http://ru.wikipedia.org/wiki/Сравнение_языков_программирования.
14. Библиотека BOOST C++ [Электронный ресурс]. – Режим доступа: http://www.solarix.ru/for_developers/cpp/boost/boost-library.shtml.
15. Boost [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/Boost>.
16. POCO [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/POCO>.
17. Qt [Электронный ресурс]. – Режим доступа: <http://ru.wikipedia.org/wiki/Qt>.
18. MEF 10.2 Ethernet Services Attributes Phase 2 [Электронный ресурс]. – Режим доступа: http://metroethernetforum.org/PDF_Documents/technical-specifications/MEF10.2.pdf.
19. Network Time Protocol Version 4: Protocol and Algorithms Specification [Электронный ресурс]. – Режим доступа: <http://tools.ietf.org/html/rfc5905>.
20. RTP: A Transport Protocol for Real-Time Applications [Электронный ресурс]. – Режим доступа: <http://www.ietf.org/rfc/rfc3550.txt>.
21. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION [Электронный ресурс]. – Режим доступа: <https://tools.ietf.org/html/rfc1035>.
22. Анализатор - генератор трафика Landslide для мобильных сетей 2,5G - 3G - 4G [Электронный ресурс]. – Режим доступа: <http://www.pr-group.ru/landslide/index.html>.
23. Экономика и организация производства программных продуктов [Электронный ресурс]. – Режим доступа: <http://megaobzor.net/e-book/book.html>.
24. Депутат О. П. Цивільна оборона. Навчальний посібник. / О. П. Депутат, І. В. Коваленко, І. С. Мужик. — Львів.: Афіша, 2001. — 336 с.

ДОДАТОК А
Технічне завдання

Вступ

Програма призначена для вимірювання характеристик передавання даних.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну роботу на тему «Дослідження та реалізація програмно-апаратних засобів вимірювання характеристик передавання даних», затверджене наказом по Національному університету «Запорізька політехніка».

A.2 Призначення розробки

Розроблена програма дозволяє виконувати вимірювання характеристик передавання даних.

A.3 Вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має виконувати такі функції:

- можливість одночасного проведення великої кількості (до 100) вимірювальних тестів;
- наявність великої кількості вбудованих типів вимірювальних тестів (udp, dns, rfc2554, icmp);
- управління SLA послуги IP VPN L2 / L3, контроль якості доступу в Інтернет, моніторинг програм L5-L7;
- plug & go автоматичне визначення в системі, завантаження сценаріїв з системи і виконання сценаріїв тестування;

- перешкодозахищеність алгоритми для роботи в високонавантажених каналах;
- гнучка система настройки;
- універсальний протокол для взаємодії із зовнішніми системами;
- різні системи оновлення.

A.3.2 Вимоги до надійності

Програма повинна забезпечувати безперебійну і цілодобову роботу, і виконувати шифрування результатів тестів.

A.3.3 Вимоги до експлуатації

Програма може бути записана як на твердому магнітному диску, так і на флеш носії. Експлуатація програмного продукту здійснюється відповідно до експлуатаційної документації на розроблену програму, що містить інформацію, необхідну для освоєння та експлуатації програмного продукту.

A.3.4 Вимоги до складу й параметрів технічних засобів

Для роботи з програмним продуктом рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 50 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша;
- мережева карта, не менше 100Mbps.

A.3.5 Вимоги до інформаційної й програмної сумісності

Для роботи програми необхідна операційна система.

A.3.6 Вимоги до транспортування і зберігання

Програма повинна зберігатися на жорсткому або на флеш носії і транспортуватися в спеціальних боксах.

A.3.7 Вимоги до програмної документації

Програма повинна поставлятися з технічним завданням, керівництвом оператора, керівництвом програміста.

A.4 Стадії та етапи розробки

Програмний продукт повинен бути виконаний у 16-ти тижневий строк з моменту отримання завдання.

ДОДАТОК Б
Опис програми

Б.1 Загальні відомості

Програма розроблена за допомогою мови програмування C++.

Б.2 Функціональне призначення

Програма призначена для вимірювання характеристик передавання даних.

Б.3 Опис логічної структури

Логічну структуру програмного проєкту можна представити у вигляді схеми, зображеної на рис. Б.1.

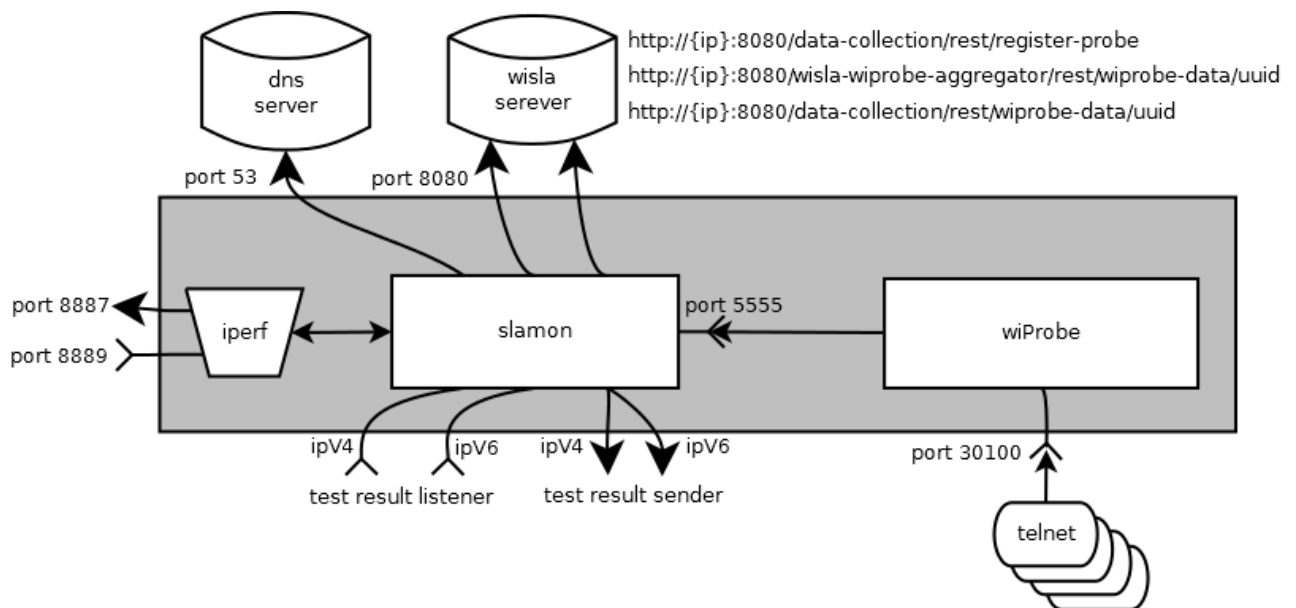


Рисунок Б.1 – Логічна структура програми

Програма складається з таких частин:

- конфігураційної – дозволяє конфігурувати вимірювальну програму;
- вимірювальної – надає можливість проведення icmp, udp, dns тестів;
- iperf – надає проведення навантажувальних RFC2554 тестів;

- wiSLA – зовнішній сервер для прийому і агрегації результатів;
- DNS – зовнішній рекурсивний dns сервер.

Б.4 Використані технічні засоби

Для роботи з програмою рекомендується використовувати наступне апаратне забезпечення:

- ПК під керівництвом операційної системи;
- тактова частота процесора 1 ГГц або більше;
- обсяг оперативної пам'яті не менше 1 Гб;
- обсяг вільного простору на диску – не менше 50 Мб;
- дозвіл екрана монітора не менше 1024x768;
- маніпулятор миша;
- мережева карта, не менше 100Mbps.

Б.5 Виклик і завантаження

Для завантаження програми необхідно скопіювати папки з програмами і запустити виконувані файли.

Б.6 Вхідні дані

Вхідними даними для програми є список тестів в файлі конфігурації.

Б.7 Вихідні дані

Вихідними даними для програми є результати тестів.

ДОДАТОК В
Текст програми

```

//cli.cpp
#include "cli.h"
#include <Poco/Timestamp.h>
#include <Poco/DateTimeFormatter.h>
#include "wrappers.h"
using namespace Poco;

Cli::Cli(Session* s):session(s)
{}

string Leaf::process(deque<string>& params)
{
    string p1=params[0];
    if(name.empty())
    {
        return processChild(p1,params);
    }

    params.pop_front();

    if(name==p1 || alias==p1)
    {
        if(params.size()>0)
        {
            string name=params[0];
            return processChild(name,params);
        }
    }
    parent->currentLeaf=this;
    if(action==NULL)
        return defAction(parent->session,params);
    return (*action)(parent->session,params);
}

string Leaf::processChild(string name, deque<string>params)
{
    for(int q=0; q<childs.size(); q++)
    {
        if(childs[q]->name==name || childs[q]->alias==name)
        {
            return childs[q]->process(params);
        }
    }
    parent->currentLeaf=this;
    if(action==NULL)
        return defAction(parent->session,params);
    return (*action)(parent->session,params);
}

vector<string> Leaf::getAutocompletion(string str)
{
    vector<string> result;
    for(int q=0; q<childs.size(); q++)
    {
        if(childs[q]->name.find(str) == 0 || str.length() == 0)
        {
            result.push_back(childs[q]->name);
        }
    }
    return result;
}

vector<string> Cli::getAutocompletion(string str)
{

```

```

    if(session->master)
        return currentLeaf->getAutocompletion(str);
    return roots[session->mode]->getAutocompletion(str);
}

string Cli::process(deque<string> str)
{
    if(session->master)
        return currentLeaf->directCall(str);
    return roots[session->mode]->process(str);
}

string Leaf::defAction(Session* ctx, deque<string> &params)
{
    return "some error happens";
}

string Leaf::directCall(deque<std::string> &params)
{
    return (*action)(parent->session,params);
}

Cli::~Cli()
{
    cout<<"~Cli()"<<endl;
    reinitAllMasters();
}

void Cli::init()
{
    Leaf* showMode=new Leaf("");
    Leaf* show=new Leaf("show","sh");
    addChild(showMode,show);

    addChild(showMode,new Leaf("exit",ShowMode::exit));
    addChild(showMode,new Leaf("whoami",ShowMode::showCurrentUser));
    addChild(showMode,new Leaf("configure","cfg",ShowMode::configure));
    addChild(showMode,new Leaf("kill",ShowMode::killSession));
    addChild(showMode,new Leaf("ris",ShowMode::slamonReInit));
    addChild(showMode,new Leaf("restart",ShowMode::restartControlProg));
    addChild(showMode,new Leaf("reboot",ShowMode::reboot));
    addChild(showMode,new Leaf("ping",ShowMode::ping));
    addChild(showMode,new Leaf("open","connect",ShowMode::connectToWiProbe));
    addChild(showMode,new Leaf("!>#",ShowMode::execute));
    addChild(showMode,new Leaf("F1",ShowMode::helper));

    addChild(show,new Leaf("sessions","s",ShowMode::showSessions));
    addChild(show,new Leaf("date",ShowMode::showDate));
    addChild(show,new Leaf("slamon",ShowMode::showSlamonSettings));
    addChild(show,new Leaf("services","ss",ShowMode::showServices));
    addChild(show,new Leaf("interface","i",ShowMode::showInterface));
    addChild(show,new Leaf("tests","t",ShowMode::showTest));
    addChild(show,new Leaf("mode",ShowMode::showMode));
    addChild(show,new Leaf("system","sys",ShowMode::showSystem));
    addChild(show,new Leaf("ntp-servers","ns",ShowMode::showNtpServers));
    addChild(show,new Leaf("version","ver",ShowMode::showVersion));
    //addChild(show,new Leaf("url",ShowMode::showUrl));
    addChild(show,new Leaf("ftp-url",ShowMode::showFtpUrl));
    addChild(show,new Leaf("wsla-url",ShowMode::showWslaUrl));
    addChild(show,new Leaf("data-url",SlamonMode::showDataUrl));
    defineRoot(SHOW,showMode);

    Leaf* configMode=new Leaf("");

```

```

addChild(configMode,new Leaf("gen",ShowMode::generate));
addChild(configMode,new Leaf("interface","i",ConfigureMode::interface));
addChild(configMode,new Leaf("test","t",ConfigureMode::test));
addChild(configMode,new Leaf("exit",ConfigureMode::exit));
addChild(configMode,new Leaf("diff",ConfigureMode::diff));
addChild(configMode,new Leaf("save",ConfigureMode::save));
addChild(configMode,new Leaf("set",ConfigureMode::setSystemProperty));
addChild(configMode,new Leaf("slamon",ConfigureMode::editSlamon));
addChild(configMode,new Leaf("ntp-sync",ConfigureMode::ntpSync));
addChild(configMode,show,false);

Leaf* remove=new Leaf("remove","r");
addChild(configMode,remove);
addChild(remove,new Leaf("test","t",ConfigureMode::removeTest));
addChild(remove,new Leaf("service","s",ConfigureMode::removeService));
addChild(remove,new Leaf("ntp-server","ns",ConfigureMode::removeNtpServer));
addChild(remove,new Leaf("interface","i",ConfigureMode::removeInterface));
addChild(remove,new Leaf("user","u",ConfigureMode::removeUser));

Leaf* add=new Leaf("add");
addChild(configMode,add);
addChild(add,new Leaf("test","t",ConfigureMode::addTest));
addChild(add,new Leaf("interface","i",ConfigureMode::addInterface));
addChild(add,new Leaf("service","s",ConfigureMode::addService));
addChild(add,new Leaf("ntp-server","ns",ConfigureMode::addNtpServer));
addChild(add,new Leaf("user","u",ConfigureMode::addUser));

addChild(configMode,new Leaf("passwd",ConfigureMode::changeUserPassword));
addChild(configMode,new Leaf("change-user-role","urc",ConfigureMode::changeUserRole));

Leaf* stop=new Leaf("stop");
addChild(configMode,stop);
addChild(stop,new Leaf("service","ss",ConfigureMode::stopService));

Leaf* start=new Leaf("start");
addChild(configMode,start);
addChild(start,new Leaf("service","ss",ConfigureMode::startService));

Leaf* restart=new Leaf("restart");
addChild(configMode,restart);
addChild(restart,new Leaf("service","ss",ConfigureMode::restartService));

defineRoot(CONFIGURE,configMode);
Leaf* interfaceMode=new Leaf("",InterfaceMode::setParam);
addChild(interfaceMode,new Leaf("show","sh",InterfaceMode::show));
addChild(interfaceMode,new Leaf("done",InterfaceMode::done));
addChild(interfaceMode,new Leaf("dhcp",InterfaceMode::dhcp));
Leaf* addL=new Leaf("add");
addChild(interfaceMode,addL);
addChild(addL,new Leaf("route",InterfaceMode::addRoute));
Leaf* removeL=new Leaf("remove");
addChild(interfaceMode,removeL);
addChild(removeL,new Leaf("route",InterfaceMode::removeRoute));
defineRoot(INTERFACE,interfaceMode);
Leaf *testMode=new Leaf("",TestMode::setParam);
addChild(testMode,new Leaf("show","sh",TestMode::show));
addChild(testMode,new Leaf("done",InterfaceMode::done));
defineRoot(TEST,testMode);
Leaf* slamonMode=new Leaf("",SlamonMode::setParam);
addChild(slamonMode, new Leaf("show","sh",SlamonMode::show));
addChild(slamonMode,new Leaf("done",InterfaceMode::done));
Leaf *addS=new Leaf("add");
Leaf *delS=new Leaf("remove","r");

```

```

    addChild(addS,new Leaf("wisla-url","url",SlamonMode::addWislaUrl));
    addChild(delS,new Leaf("wisla-url","url",SlamonMode::removeWislaUrl));
    addChild(slamonMode,addS);
    addChild(slamonMode,delS);
    defineRoot(SLAMON,slamonMode);
}

//cli.h
#ifndef CLI_H
#define CLI_H

#include<deque>
#include<map>
#include "Session.h"
#include "defs.h"
#include<iostream>
#include "Poco/AutoPtr.h"
#include "Poco/AutoReleasePool.h"

using namespace std;
using namespace Poco;
class Session;
class Cli;

typedef string (*cmd)(Session*,deque<string>&);

class Leaf
{
public:
    Leaf(string _name):name(_name),action(NULL){}
    Leaf(string _name, string shortName):name(_name),alias(shortName),action(NULL){}
    Leaf(string _name, string shortName,cmd cmdAction):name(_name),alias(shortName),action(cmdAction){}
    Leaf(string _name,cmd cmdAction):name(_name),action(cmdAction){}
    void setParent(Cli*p){parent=p;}

    string process(deque<string>&);
    vector<string> getAutocompletion(string str);
    string directCall(deque<string>&);
    void release()
    {
        delete this;
    }

friend class Cli;
private:
    string helpMsg;
    cmd action;
    string name;
    string alias;
    vector<Leaf*>childs;
    Cli* parent;
    string processChild(string,deque<string>);
    void addChild(Leaf* child){childs.push_back(child);child->parent=parent;}
    string defAction(Session* ctx, deque<string>& params);
};

class Cli
{
public:
    Cli(Session*);
    void init();
    Session* session;
    Leaf* currentLeaf;

```

```

    string process(deque<string>params);
    vector<string> getAutocompletion(string str);
    ~Cli();
    void addChild(Leaf* parent,Leaf* child, bool addToReleasePool=true){parent->addChild(child);child-
>parent=this;if(addToReleasePool)pool.add(child);}
    void defineRoot(Modes m,Leaf* root){roots[m]=root;root->parent=this;pool.add(root);}
private:
    map<Modes,Leaf*> roots;
    AutoReleasePool<Leaf>pool;
};

//Session.cpp
#ifdef // CLI_H
#include "Session.h"
#include <string>
#include <iostream>
#include "defs.h"

using namespace std;

void stringToDeque(string str, deque<string>&result)
{
    StringTokenizer parts(str, " ", 1 | 2);
    for (int q = 0; q < parts.count(); q++)
    {
        result.push_back(parts[q]);
    }
}
Session::Session(const StreamSocket& socket) : TCPServerConnection(socket)
{
    cout << "Session()" << endl;
    cli = new Cli(this);
    cli->init();
    isActive = true;
    mode = INIT;
    role = User::UNDEFINE;
    prompt = "wi-probe>";
    master = false;
    isProxy = false;
    maxComandInHistory = 10;
}

Session::~~Session()
{
    delete cli;
    cout << "~Session()" << endl;
}

void Session::run()
{
    SessionManager& mng = SessionManager::instance();
    cout << "begin with id=" << id << endl;
    DialogSocket sck(socket());
    cout << "IP IS" << sck.address().toString() << endl;
    sck.setReceiveTimeout(560 * 1000 * 1000);
    bool status = true;
    mode = SHOW;
    string name, pass, line;
    sck.sendMessage("Login:");
    sck.receiveMessage(name);
    cleanStr(name);
    sck.sendTelnetCommand(DialogSocket::TELNET_WILL,1);
    sck.sendMessage("Password:");

```

```

sck.receiveMessage(pass);
cleanStr(pass);
sck.sendMessage("");
User::Roles role = mng.resolveUser(this, name, pass);
if (role == User::UNDEFINE)
{
    leaveSession("Invalid login or password");
    return;
}
this->role = role;
this->entityName = name;
sck.sendTelnetCommand(DialogSocket::TELNET_WILL, 3);
sck.sendString(prompt);
char ch = getNextSymbol(sck);
int cmdHistoryIndex = -1;
while(status)
{
    line = "";
    string result;
    try
    {
        while (!isEnter(ch))
        {
            if (isSymbol(ch))
            {
                line+=ch;
                sck.sendByte(ch);
            }
            else if(isBackspace(ch) && line.size()>0)
            {
                deleteSymbol(sck);
                line = line.substr(0,line.length()-1);
            }
            else if(isArrow(ch))
            {
                ch = sck.get();
                ch = sck.get();
                if(isUpArrow(ch))
                {
                    deleteLine(sck,line);
                    if(!cmdHistory.empty())
                    {
                        if(cmdHistoryIndex == (cmdHistory.size()-1) )
                        {
                            cmdHistoryIndex--;
                        }
                        cmdHistoryIndex++;
                        line = cmdHistory.at(cmdHistoryIndex);
                    }
                    sck.sendString(line);
                }
                if(isDownArrow(ch))
                {
                    deleteLine(sck,line);
                    if(!cmdHistory.empty() && cmdHistoryIndex > 0)
                    {
                        cmdHistoryIndex--;
                        line = cmdHistory.at(cmdHistoryIndex);
                    }
                    sck.sendString(line);
                }
            }
            else if(isTab(ch))

```

```

        {
            if(line.find(" ") == std::string::npos)
            {
                parintAutocompletion(sck,line);
            }
        }
        ch = sck.get();
    }
    sck.sendMessage("");
    if(!status)
    {
        break;
    }
    if(line.find(8)!=string::npos)
    {
        sck.sendMessage("Wrong symbols detected!!! Please enter once again!");
        line="";
    }
    trimInPlace(line);
    if(!line.empty())
    {
        if(isProxy)
        {
            sendViaProxy(line);
            receiveViaProxy(result);
            sck.sendString(result);
        }
        else
        {
            deque<string>tokens;
            cmdHistoryIndex=-1;
            cmdHistory.push_front(line);
            if( cmdHistory.size() > maxComandInHistory )
            {
                cmdHistory.pop_back();
            }
            stringToDeque(line,tokens);
            result=cli->process(tokens);
            replaceInPlace(result,"\n","\r\n");

            sck.sendMessage(result);
        }
    }
    sck.sendString(prompt);
    ch=getNextSymbol(sck);
}
catch (Poco::Exception&e)
{
    leaveSession("User session was closed after timeout" + e.displayText());
    return;
}
}
leaveSession("Goodbye");
}

void Session::leaveSession(string msg)
{
    SessionManager& mng = SessionManager::instance();
    mng.deleteSession(this);
    DialogSocket sck(socket());
    sck.sendMessage(msg);
    cout << "end" << endl;
    sck.shutdown();
}

```

```

    sck.close();
}

void Session::say(string msg)
{
    DialogSocket sck(socket());
    sck.sendMessage(msg);
}

void Session::connectViaProxy(string host, int port)
{
    string hostM = host + ":" + NumberFormatter::format(port);
    say("Connecting to " + hostM);
    try
    {
        proxy.close();
        SocketAddress addr(host, port);
        proxy.connect(addr);
    }
    catch (Exception& e)
    {
        say("Error during connection to " + hostM);
        say(e.displayText());
    }
    isProxy = true;
    say("Successfully connected to " + hostM);
    say("Type 'close' to close the connection");
    prompt = "@";
    string result;
    if (receiveViaProxy(result))
    {
        say(result);
    }
}

void Session::sendViaProxy(string message)
{
    if (message == "close")
    {
        closeProxy();
    }
    else
    {
        proxy.sendMessage(message);
    }
}

bool Session::receiveViaProxy(string& result)
{
    char buffer[500];
    if (isProxy)
    {
        bool canRead = true;
        bool isData = false;
        proxy.setReceiveTimeout(500 * 1000);
        while (canRead)
        {
            try
            {
                int read = proxy.receiveBytes(buffer, 500);
                if (read)
                {
                    string line(buffer, read);
                    result += line;
                }
            }
        }
    }
}

```

```

        isData = true;
    }
    else
        canRead = false;
    }
    catch (Exception &e)
    {
        canRead = false;
        if (!isData)
        {
            say("Timeout during receiving result from proxy");
            closeProxy();
        }
    }
}
}
if (result.empty())
{
    closeProxy();
    return false;
}
return true;
}
void Session::cleanStr(string& str)
{
    for (int i = 0; i < str.length(); i++)
    {
        if (isdigit(str[i]) || isalpha(str[i]))
        {
            str = str.substr(i, str.length() - i);
            break;
        }
    }
}
bool Session::isSymbol(char ch)
{
    return isalpha(ch) || isdigit(ch) || ch == ' ' || ch == '.'
        || ch == '\\' || ch == ':' || ch == '-' || ch == '_';
}
bool Session::isEnter(char ch)
{
    return (int) ch == 0 ||
        (int) ch == 13;
}
bool Session::isBackspace(char ch)
{
    return (int) ch == 127 ||
        (int) ch == 8;
}
bool Session::isTab(char ch)
{
    return (int) ch == 9;
}

bool Session::isArrow(char ch)
{
    return (int) ch == 27 ||
        (int) ch == 8;
}

bool Session::isUpArrow(char ch)
{
    return (int) ch == 65;
}

```

```

}

bool Session::isDownArrow(char ch)
{
    return (int)ch == 66;
}

void Session::deleteSymbol(DialogSocket& sck)
{
    unsigned char buffer[] = {0x08,0x20,0x08};
    sck.sendBytes(buffer,3,0);
}

void Session::deleteLine(DialogSocket& sck, string& line)
{
    for(int i=0; i<line.length(); i++)
        {deleteSymbol(sck);}
}

void Session::parintAutocompletion(DialogSocket& sck,string& line)
{
    vector<string> cmdList = cli->getAutocompletion(line);
    if(cmdList.size() == 1)
    {
        deleteLine(sck,line);
        line = cmdList[0];
        sck.sendString(line);
    }
    else if (cmdList.size(>1)
    {
        string buff = line+promt;
        deleteLine(sck,buff);
        for(int i=0; i<cmdList.size(); i++)
        {
            sck.sendString(cmdList[i]+" ");
        }
        sck.sendMessage("");
        sck.sendString(promt+line);
    }
}

char Session::getNextSymbol(DialogSocket& sck)
{
    char ch = -1;
    while(!isSymbol(ch) && !isArrow(ch))
    {
        ch = sck.get();
        if(isTab(ch))
        {
            string buff="";
            parintAutocompletion(sck,buff);
        }
    }
    return ch;
}

void Session::closeProxy()

```

ДОДАТОК Г
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та реалізація програмно- апаратних засобів вимірювання характеристик передавання даних

Виконав

ст. гр. КНТ-129м

Р. В. Осташко

Керівник:

к.т.н., доцент

А. О. Олійник

1

Рисунок Г.1 – Слайд № 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес вимірювання якості зв'язку.

Предмет дослідження – засоби вимірювання характеристик передавання даних.

Мета роботи – дослідження та реалізація засобів вимірювання характеристик передавання даних.

2

Рисунок Г.2 – Слайд № 2

Завдання роботи

Для досягнення поставленої мети необхідно було розв'язати такі завдання:

- аналіз та порівняння існуючих засобів вимірювання характеристик передавання даних;
- розробка технічного завдання для реалізації програми;
- вибір середовища розробки програмного забезпечення;
- розробка моделі програмного забезпечення вимірювання характеристик передавання даних;
- конструювання програмного забезпечення відповідно до технічного завдання;
- тестування розробленого програмного забезпечення.

3

Рисунок Г.3 – Слайд № 3

Порівняльний аналіз програмно-апаратних засобів вимірювання характеристик передавання даних

Критерій порівняння	Пристрої		
	EtherNID	Cisco IP Monitor	Carrier ETX-203A
Наявність CLI доступу	+	+	+
Наявність SSH доступу	+	-	-
Performance assurance тест	+	+	+
DNS тест	-	-	-
RFC2554 тест	+	+	+
Максимальна кількість підтримуваних тестів (одного виду)	50	40	50
Загальна кількість одночасно виконуваних тестів	50	40	50
Кількість планованих RFC тестів	1	1	1
Можливість модифікації	неможливо	неможливо	неможливо
Критичні межі вимірювань односторонніх показників (PL – Packet Loss; RTT - Round Trip Time)	до 20% PL; до 300мс RTT	до 30% PL; до 400мс RTT	до 30% PL; до 400мс RTT
Ціна за одиницю (\$)	2000	2500	2300

4

Рисунок Г.4 – Слайд № 4

Аналіз та вибір апаратної платформи для реалізації та використання розроблюваного програмного забезпечення

Прилад	Характеристики								
	Операційна система	Тактова частота	Обсяг накопичувача	Обсяг пам'яті	USB	Ethernet	WiFi	Power supply	Ціна(\$)
SheevaPlug	Linux	1.2 GHz	512 MB	512 MB	2	1	–	embedded	50
Raspberry Pi	Linux	700 MHz	External SD card	512 MB	2	1	–	external	78
MK802	Android	1.5GHz	4 GB	512 MB	1	–	1	external	60

5

Рисунок Г.5 – Слайд № 5

Вибір мови програмування

Можливість	Мова			
	C++	Java	Perl	Bash
Імперативна	+	+	+	–
Об'єктно-орієнтована	+	+	+	–
Функціональна	+/-	-/+	+	–
Статична типізація	+	+	+/-	–
Динамічна типізація	–	–	+	+
Явна типізація	+	+	-/+	–
Неявна типізація	-/+	–	+	+
Open-source компілятор (інтерпретатор)	+	+	+	+
Можливість компіляції	+	+	+	–
Створення об'єктів на стеку	+	–	–	–
Ручне управління пам'яттю	+	–	–	–
Прибирання сміття	-/+	+	+	+
Інтерфейси	+	+	+/-	–
Макроси	+	–	+	–
Шаблони / generics	+	+	–	–
Перевантаження функцій	+	+	–	+

6

Рисунок Г.6 – Слайд № 6

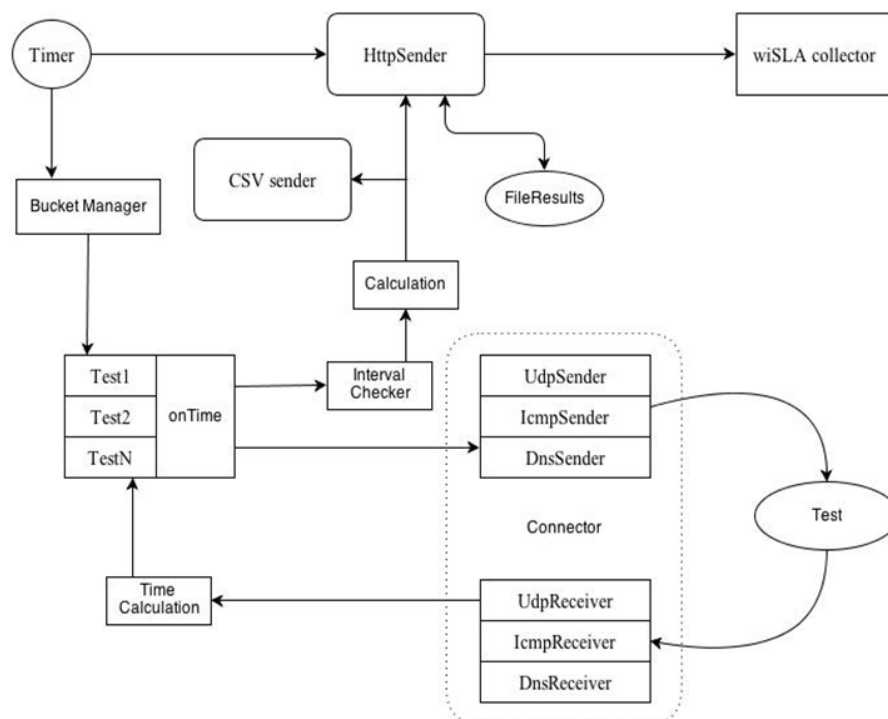
Вибір середовища розробки

Характеристики	QT	POCO	BOOST
Шаблон каркасу застосунку	+	+	-
Шаблон синглтону	-	+	-
Розмір бібліотеки (Мб)	4	1,2	залежить від використовуваних бібліотек
Кросплатформні потоки	+	+	+
Кросплатформні сокети	+	+	+
Покращені контейнери	+	+	+
Розумні показники	+	+	+
Робота з xml	+	+	-
Веб-сервер	-	+	-
Пул потоків	+	+	-
Система подій	+	+	+

7

Рисунок Г.7 – Слайд № 7

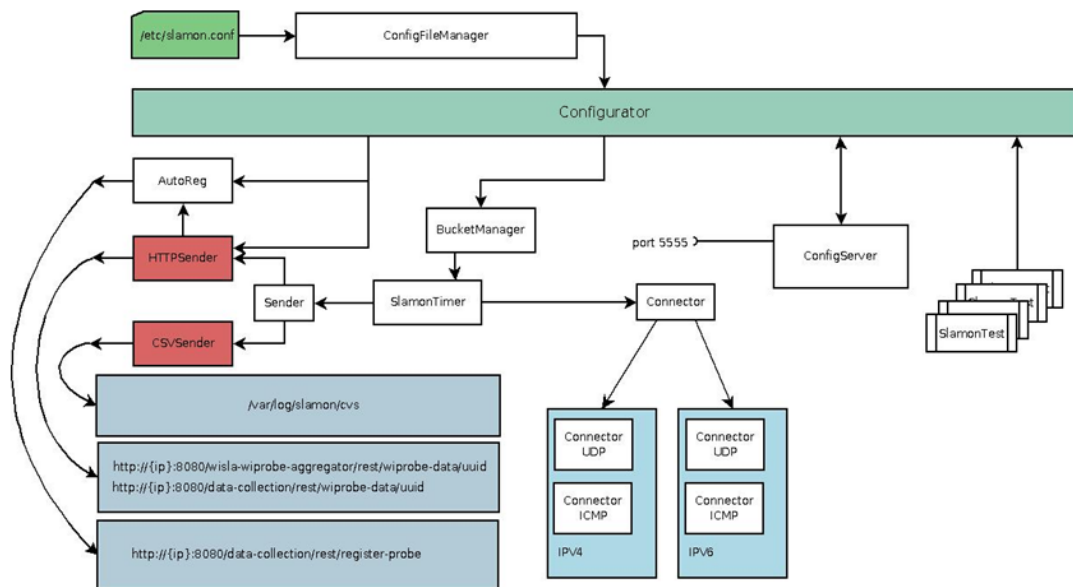
Модель програмного забезпечення вимірювання характеристик передавання даних



8

Рисунок Г.8 – Слайд № 8

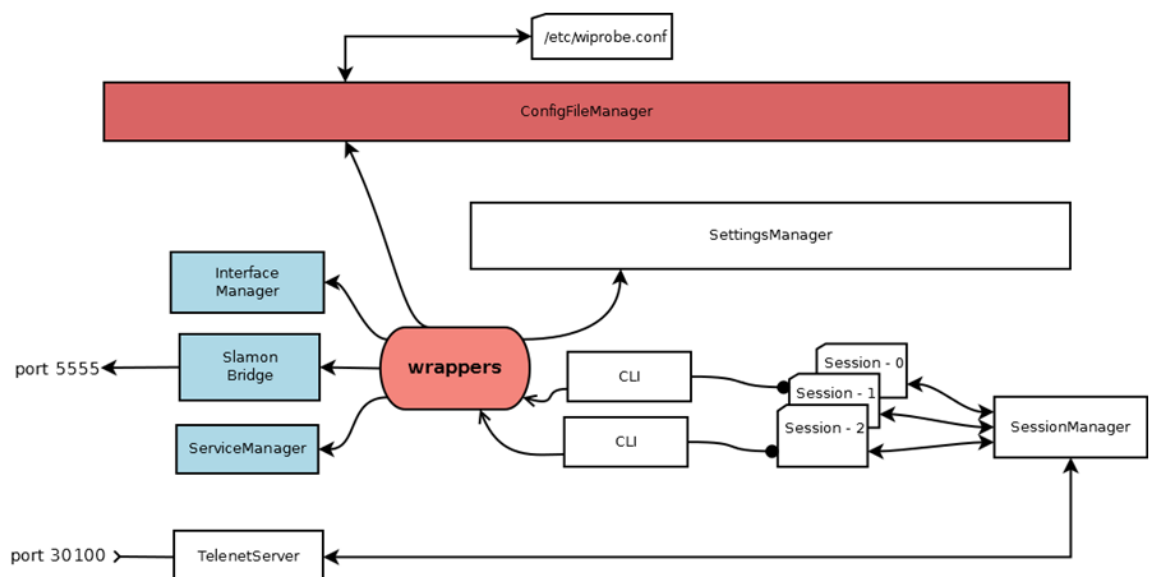
Архітектура головної програми



9

Рисунок Г.9 – Слайд № 9

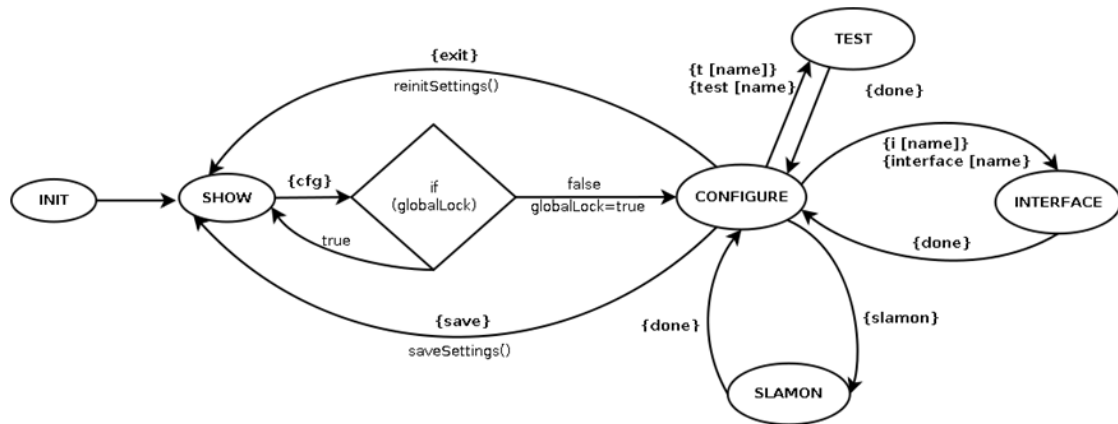
Архітектура конфігураційної програми



10

Рисунок Г.10 – Слайд № 10

Схема взаємодії режимів програми конфігурації



11

Рисунок Г.11 – Слайд № 11

Приклад роботи з програмою

```

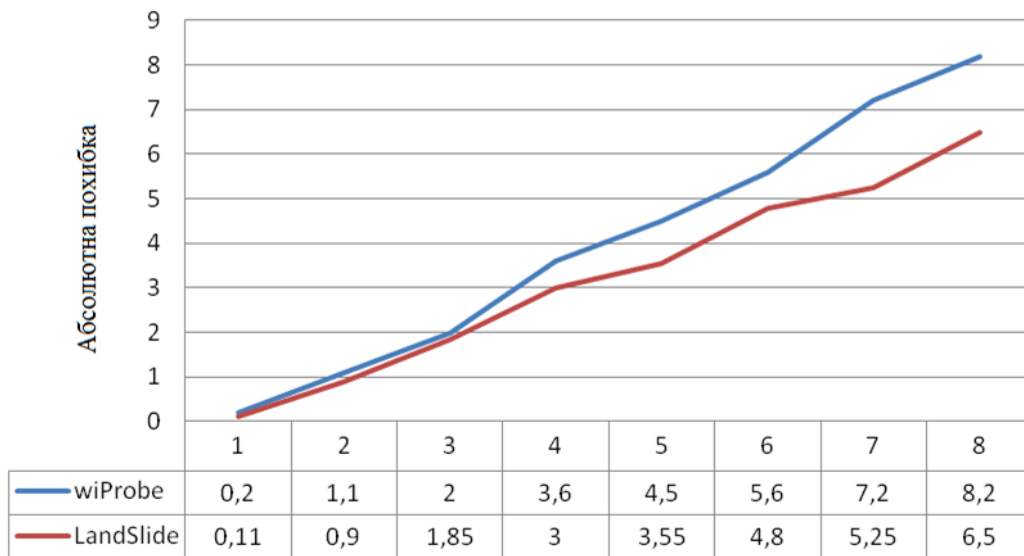
ftp3.metrotekit.zp.ua - PuTTY
login as: root
Authenticating with public key "imported-openssh-key"
Last login: Wed Apr 24 08:41:18 2020 from 192.168.2.53
FreeBSD 9.0-RELEASE (ROUTER_ZP) #0: Fri Mar 15 14:46:07 EET 2020

[80.84.176.10]root@router/root# telnet 192.168.2.93 30100
Trying 192.168.2.93...
Connected to 192.168.2.93.
Escape character is '^]'.
Login:
user
Password:
user
wi-probe>sh ver
21858
wi-probe>sh date
Sat Apr 27 14:00:17 UTC 2020
wi-probe>sh t
wislaTest-7110 wislaTest-9184
wi-probe>
  
```

12

Рисунок Г.12 – Слайд № 12

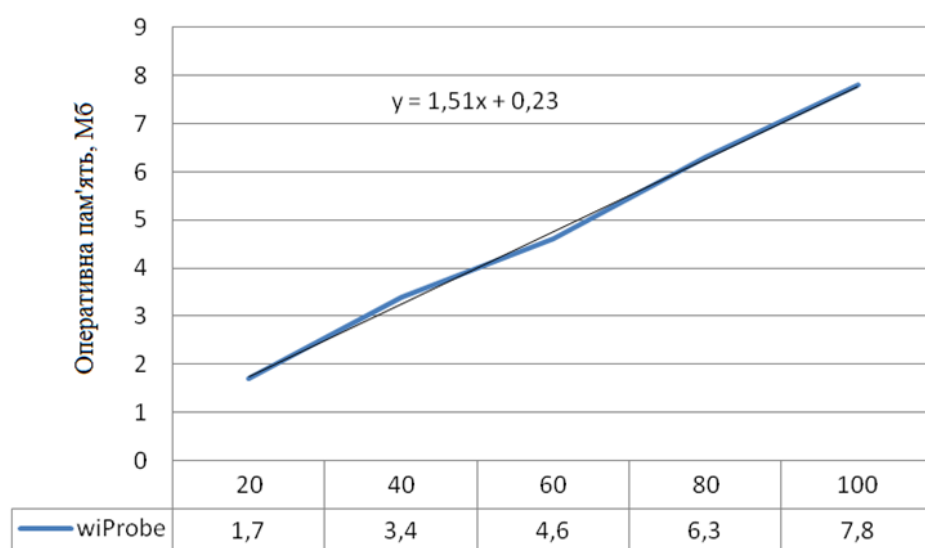
Залежність коефіцієнта похибки від завантаженості каналу



13

Рисунок Г.13 – Слайд № 13

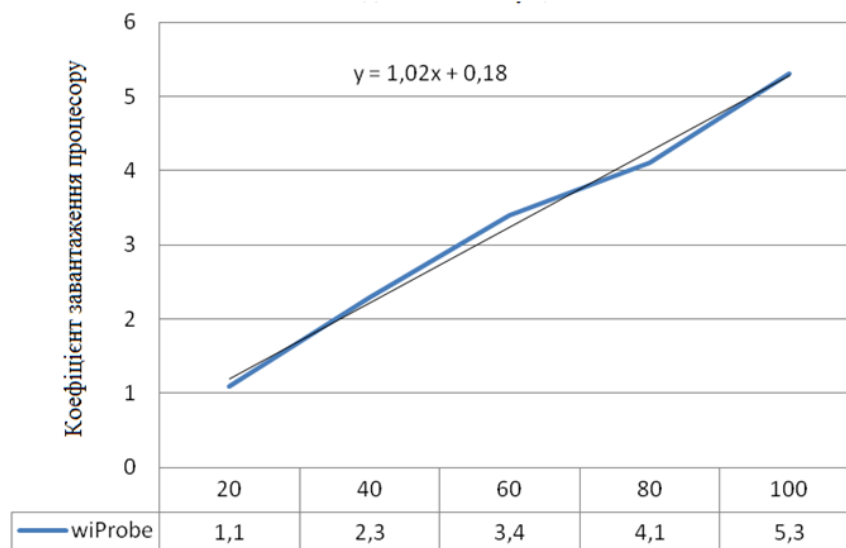
Залежність використаної оперативної пам'яті від кількості тестів



14

Рисунок Г.14 – Слайд № 14

Залежність завантаження центрального процесора від кількості запущених тестів



15

Рисунок Г.15 – Слайд № 15

ВИСНОВКИ

В результаті виконаної роботи було розв'язано актуальне завдання дослідження та реалізації програмно-апаратних засобів вимірювання характеристик передавання даних.

Наукова новизна роботи полягає у тому, що розроблено модель програмного забезпечення вимірювання характеристик передавання даних, яка характеризується об'єктами Timer, Bucket Manager, DataSender, Connector, Calculation, та дозволяє розробляти програмне забезпечення для вимірювання різних характеристик передавання даних.

Практична значущість отриманих результатів полягає у тому, що розроблено програмне забезпечення для вимірювання характеристик передавання даних.

Проведено функціональне і тестування навантаженням, що дозволило об'єктивно оцінити відповідність програми всім функціональним вимогам, а також споживання ресурсів при проведенні великої кількості одночасних тестів.

В результаті проведених функціонального тестування та тестування навантаженням було виявлено, що розроблений комплекс володіє необхідними вимогами визначеними в технічному завданні. Тестування навантаженням довело, що вимірювальна програма при проведенні великої кількості тестів використовує не велику кількість ресурсів, залишаючи великий запас ресурсів для різноманітних призначених для користувача завдань (проведення додаткових тестів, виконання різного призначеного для користувача програмного забезпечення).

16

Рисунок Г.16 – Слайд № 16