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

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи передачі інформації в комп’ютерних мережах систем критичного застосування”

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Почерняєва А.В.
«____» _____ 2025 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*“*Інформаційні технології*”*_____

Спеціальність *123 “*Комп’ютерна інженерія*”*

Освітньо-професійна (освітньо-наукова) програма _*“*Комп’ютерна інженерія*”*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2025 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Почерняєвій Анні Вікторівні
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи передачі інформації в комп’ютерних мережах систем критичного застосування*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 46-02 від 17.01.2025 року

3. Строк подання студентом роботи до захисту *23.05.2025 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи передачі інформації в комп’ютерних мережах систем критичного застосування*

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

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки

6. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	23.05.2025 р.	

Дата видачі завдання
« 17 » січня 2025 р.

Підпис керівника
_____ Усік П.С.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Почерняєва А.В.
(прізвище та ініціали)

АНОТАЦІЯ

Почерняєва А.В. Програмне забезпечення системи передачі інформації в комп'ютерних мережах систем критичного застосування. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи передачі інформації в комп'ютерних мережах систем критичного застосування.

Метою розробки є програмне забезпечення системи передачі інформації в комп'ютерних мережах систем критичного застосування.

Результат роботи – програмна реалізація системи передачі інформації в комп'ютерних мережах систем критичного застосування.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерна інженерія, системи передачі інформації

ABSTRACT

Pochernyayeva A.V. Software for the information transmission system in computer networks of critical application systems. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed, which is intended for the information transmission system in computer networks of critical application systems.

The purpose of the development is software for the information transmission system in computer networks of critical application systems.

The result of the work is the software implementation of the information transmission system in computer networks of critical application systems.

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

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

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

The program was developed in the Python environment.

Keywords: computer engineering, information transmission systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КМ	–	комп'ютерна мережа
РЛІ	–	радіолокаційна інформація
РЛС	–	радіолокаційна станція
СКЗ	–	системи критичного застосування

ВСТУП

Актуальність теми. Критично важлива програма – це програмне забезпечення або набір пов'язаних програм, які мають працювати безперервно, щоб бізнес або бізнес-сегмент були успішними.

Навіть короткий простой критично важливої програми може мати негативні фінансові наслідки. Крім втрати продуктивності, збій у роботі критично важливого додатка також може завдати шкоди репутації компанії.

Приклади критично важливих програм варіюються від галузі до галузі. Наприклад, компанія швидкої допомоги може вважати програму автоматичного пошуку транспортних засобів критично важливою, але сантехнічне підприємство, яке використовує те саме програмне забезпечення, може охарактеризувати її як важливу, але не важливу.

Додаток є критично важливим, якщо він важливий для бізнес-операцій. У критично важливих програмах не повинно виникати простоїв, коли кінцеві користувачі, ймовірно, ними будуть користуватися. Існує багато можливих критично важливих IT-сервісів, і важливість різних систем відрізняється від екосистеми до екосистеми.

Архітектори, розробники, тестувальники, команди IT-операцій та інший персонал, який підтримує критично важливі програми, повинні цінувати стабільність і доступність . Приклади стратегій для забезпечення безперервної роботи включають підтримку резервних копій програми, а також IT-систем та інфраструктури центру обробки даних, на якій вона працює; виконання гарячих резервних копій; а також використання дублюючих проміжних і виробничих середовищ для ретельного тестування.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи передачі інформації в комп'ютерних мережах систем критичного застосування.

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

– Огляд існуючих систем передачі інформації в комп'ютерних мережах систем критичного застосування.

– Дослідження системи передачі інформації в комп'ютерних мережах систем критичного застосування.

– Програмна реалізація системи передачі інформації в комп'ютерних мережах систем критичного застосування.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі передачі інформації в комп'ютерних мережах систем критичного застосування.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи передачі інформації в комп'ютерних мережах систем критичного застосування, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Позначаючи програми та робочі навантаження, IT-команди можуть визначати пріоритетність кожного елемента програми для оновлень, усунення несправностей і обслуговування.

Загальні мітки включають критично важливі для місії, критичні для бізнесу та низький пріоритет, але загальновизнаного списку критичних рівнів немає. Наприклад, підприємство та стартап можуть абсолютно по-різному класифікувати програми. Подібним чином компанії в одній галузевій вертикалі можуть класифікувати більше програм як критично важливих, ніж компанії в іншій.

На відміну від критично важливої програми, критично важлива для бізнесу програма є важливою для діяльності компанії, але її збій не завадить загальній організації функціонувати на базовому рівні. Збій критично важливої для бізнесу програми перешкоджає продуктивності або взаємодії з користувачем, але не такою мірою, як збій критично важливої програми. Користувачі можуть звернутися до альтернатив, поки IT відновить обслуговування.

Тривалий збій критично важливого для бізнесу додатка може спричинити серйозні фінансові втрати для організації. Однак критично важлива для бізнесу система може не працювати на невеликий проміжок часу, наприклад на кілька годин, без серйозної шкоди для бізнесу чи його прибутку.

Некритичні або низькопріоритетні програми можуть залишатися непридатними для використання або мати низьку продуктивність після збою протягом кількох днів або тижнів з лише незначним впливом на IT-середовище чи бізнес. Приклади некритичних програм включають рідко використовувані бізнес-програми та програми, які просто полегшують виконання інших завдань.

1.2 Область застосування

Під час розгортання критично важливого програмного забезпечення IT-адміністратори повинні точно визначити, яка підтримка потрібна, щоб гарантувати, що програма може працювати в неоптимальних умовах. Компанії повинні оцінити свої технічні вимоги до стійкості та доступності додатків, а також свої вимоги до повернення інвестицій і вартості.

Наприклад, сервер, який підтримує повсякденну роботу, повинен мати кілька резервних джерел живлення та резервних копій. Якщо центр обробки даних втрачає електроенергію, резервне джерело живлення може підтримувати всі критично важливі системи в режимі онлайн з мінімальними перебоями.

Залежно від бюджету компанії та фізичної інфраструктури центру обробки даних критично важливі програми вимагають принаймні резервування N+1, де «1» означає додатковий компонент резервного копіювання. Це число збільшується, коли до системи додається більше компонентів резервного копіювання - наприклад, два компоненти резервного копіювання вказують на резервування N+2.

Резервування серверів та інших компонентів центру обробки даних є ключовим фактором стійкості в разі збою або іншої проблеми.

Копії програми можуть забезпечити такий самий вид резервування для критично важливих служб. Наприклад, резервний сервер N+1 має додатковий сервер, налаштований відповідно до тих самих специфікацій, що й звичайна програма, щоб взяти на себе робочі навантаження, якщо програма виходить з ладу.

Часте й автоматизоване резервне копіювання забезпечує збереження конфігурації та оновлень будь-якої критично важливої програми в разі погіршення роботи служби або збою, тим самим захищаючи програми від пошкодження або видалення.

ІТ-адміністратори часто створюють свої плани аварійного відновлення, щоб визначити пріоритети для відновлення критично важливих програм. Адміністратори також можуть вирішити оновлювати критично важливі програми рідше, ніж програми з нижчим пріоритетом, щоб зменшити ризик внесення потенційно проблематичних змін.

Підтримка служби підтримки – 24/7 або 9-to-5 – це ще один спосіб забезпечити доступність критично важливих комп'ютерів і програм. Позначення 24/7 і 9-to-5 стосуються робочого тижня для служби підтримки. Критично важливе апаратне або програмне забезпечення має мати цілодобову підтримку служби підтримки, тоді як для менш важливих послуг може знадобитися обслуговування служби підтримки лише в робочі години.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи передачі інформації в комп'ютерних мережах систем критичного застосування, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Програмне забезпечення для моніторингу мережі відстежує час безперебійної роботи, швидкість передачі, якість обслуговування (QoS) і/або несправність компонентів і записів, документів і сповіщає мережевого адміністратора (через електронну пошту, SMS або інші сигнали тривоги) у разі збоїв або погіршення якості обслуговування. Моніторинг мережі є частиною керування мережею.

Загальні характеристики

- Управління продуктивністю мережі.
- Моніторинг руху.
- Відображення мережі.
- Керування конфігурацією мережі.
- Моніторинг мережевого середовища.
- Мережева аналітика.
- Якість обслуговування мережі.
- Звітність мережі.
- Управління відповідністю мережі.
- Оперативна панель.

Оскільки мережі стають дедалі складнішими завдяки хмарній інтеграції, налаштуванням віддаленої роботи та розгортанню IoT, потреба в надійних рішеннях для моніторингу ніколи не була такою важливою. Організації, які не в змозі запровадити ефективний моніторинг мережі, ризикують простоєм, уразливістю системи безпеки та, зрештою, значним порушенням роботи.

У цьому розділі розглядаються 10 найкращих інструментів моніторингу мережі на 2025 рік, аналізуються їхні функції, сильні сторони й унікальні можливості, щоб допомогти вам прийняти обґрунтоване рішення щодо конкретних потреб вашої організації.

Що таке інструмент моніторингу мережі?

Платформи моніторингу мережі являють собою основні технологічні структури, які систематично спостерігають, оцінюють і документують робочий стан, показники продуктивності та рівень безпеки цифрової інфраструктури організації. Ці продумані рішення для моніторингу дають IT-відділам можливість миттєво аналізувати критичну поведінку мережі – від моделей потоку трафіку та вузьких місць передачі даних до робочих станів пристроїв і потенційних спроб використання вразливостей.

Ці спеціалізовані програмні додатки для моніторингу мережі функціонують шляхом постійного збору телеметрії від мережевого обладнання, включаючи маршрутизатори, комутатори, балансувальники навантаження, брандмауери, сервери та кінцеві точки. Зібрані дані піддаються інтелектуальному аналізу для встановлення базових параметрів продуктивності, виявлення операційних відхилень, прогнозування потенційного погіршення системи та ізоляції проблемних сегментів мережі до того, як вони каскадом призведуть до збоїв, що вплинуть на бізнес.

Сучасні інструменти моніторингу мережі значно розвинулися за межі простих перевірок стану вгору/вниз і включають розширені можливості, такі як:

– Автоматизоване виявлення топології мережі та відображення зв’язків.

– Виявлення аномалій за допомогою штучного інтелекту зі встановленням базового рівня поведінки.

– Налаштовувані інтерфейси візуалізації з інформаційними панелями для певних ролей.

– Складні механізми оповіщення з робочими процесами ескалації.

– Комплексна аналітика продуктивності з історичними трендами.

– Моніторинг використання пропускної здатності з аналізом трафіку.

– Управління конфігурацією та відстеження змін.

– Моніторинг безпеки з інтеграцією виявлення загроз.

Фундаментальна зміна, яку забезпечують ці інструменти, полягає в переході від реактивного пожежогасіння до проактивного керування мережею. Виявляючи зниження продуктивності, незвичайні шаблони трафіку або проблеми з обладнанням до того, як вони виявляться збоями в обслуговуванні, організації можуть підтримувати більш високу надійність обслуговування, одночасно зменшуючи операційні накладні витрати та витрати на підтримку.

Ефективний моніторинг мережі є наріжним каменем сучасних IT-операцій, забезпечуючи безперервну видимість і дієвий інтелект, необхідний для підтримки цифрової інфраструктури, яка забезпечує сучасні бізнес-операції в дедалі складніших гібридних середовищах, що охоплюють локальні центри обробки даних, хмарні платформи та віддалені місця.

Монітор продуктивності мережі SolarWinds

Монітор продуктивності мережі SolarWinds (NPM) залишається лідером галузі завдяки комплексному підходу до видимості мережі та потужним діагностичним можливостям. Це рішення корпоративного рівня забезпечує моніторинг мережі різних постачальників з інтуїтивно зрозумілими інструментами візуалізації для швидкого усунення несправностей.

Основні характеристики:

– Технології NetPath™ і PerfStack™ для покращеного усунення несправностей.

– Розширена візуалізація мережі з відображенням топології.

– Інтелектуальне сповіщення з налаштовуваними пороговими значеннями.

– Комплексна підтримка пристроїв від постачальників.

– Автоматичне виявлення та відображення мережі.

– Моніторинг і аналіз бездротової мережі.

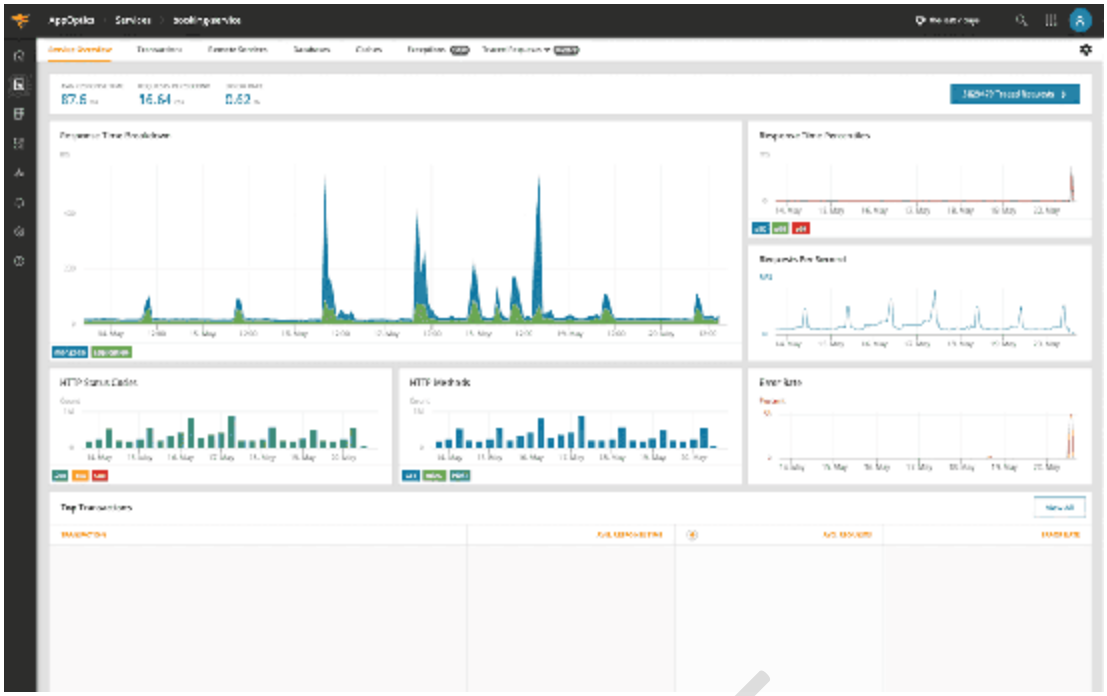

Рисунок 2.1 – Монітор продуктивності мережі SolarWinds

Унікальна функція: технологія NetPath™ забезпечує покроковий аналіз у гібридних середовищах, візуалізуючи весь мережевий шлях для критичних програм і служб.

Переваги:

– Виняткова глибина можливостей моніторингу.
– Інтуїтивно зрозумілий інтерфейс, який можна налаштувати.
– Чудова масштабованість для корпоративних мереж.
– Надійна підтримка пристроїв незалежно від постачальника.

Недоліки:

– Вища ціна, ніж у багатьох конкурентів.
– Може вимагати значних ресурсів для більшого розгортання.
– Більш крутий курс навчання для нових адміністраторів.

Безкоштовна пробна версія протягом 30 днів.

Мережевий монітор Paessler PRTG

PRTG пропонує уніфікований підхід до моніторингу, який виходить за межі мереж і включає сервери, програми тощо. Її модель ліцензування на основі датчиків забезпечує гнучкість для організацій будь-якого розміру.

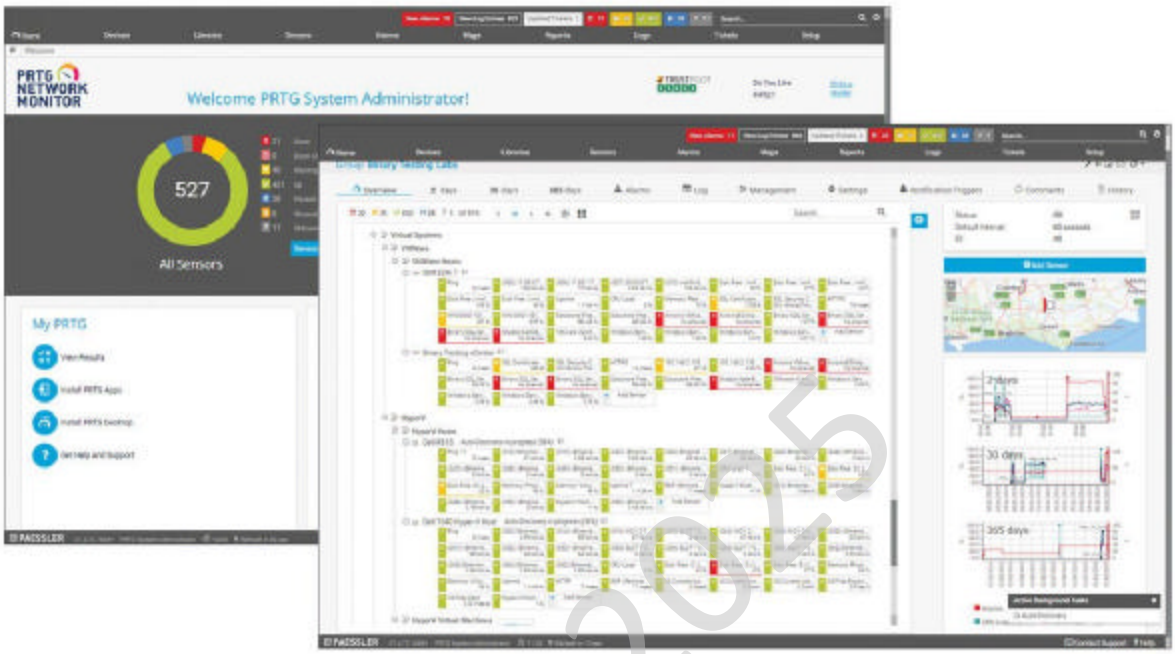

Рисунок 2.2 – Мережевий монітор Paessler PRTG

Основні характеристики:

– Понад 250 попередньо налаштованих типів датчиків.
– Автоматичне виявлення мережевих пристроїв.
– Гнучкі параметри сповіщень і сповіщень.
– Налаштовувані інформаційні панелі та звітність.
– Варіанти локального або розміщеного розгортання.
– Мобільні програми для iOS та Android.

Унікальна функція: «сенсорний» підхід PRTG до ліцензування забезпечує виняткову гнучкість, дозволяючи організаціям точно розподіляти ресурси моніторингу там, де вони найбільше потрібні.

Переваги:

– Прозоре ліцензування на основі датчиків.

– Комплексний моніторинг за межами мереж.

– Зручний інтерфейс з мінімальною конфігурацією.

– Чудові мобільні додатки.

Недоліки:

– Може стати дорогим із збільшенням кількості датчиків.

– Обмежені налаштування порівняно з деякими корпоративними рішеннями.

– Можливості звітування менш надійні, ніж спеціалізовані інструменти.

Безкоштовна пробна версія протягом 30 днів.

Nagios Core

Піонер моніторингу з відкритим кодом, Nagios Core продовжує пропонувати потужну структуру для моніторингу мережі з неперевершеною гнучкістю та можливостями налаштування.

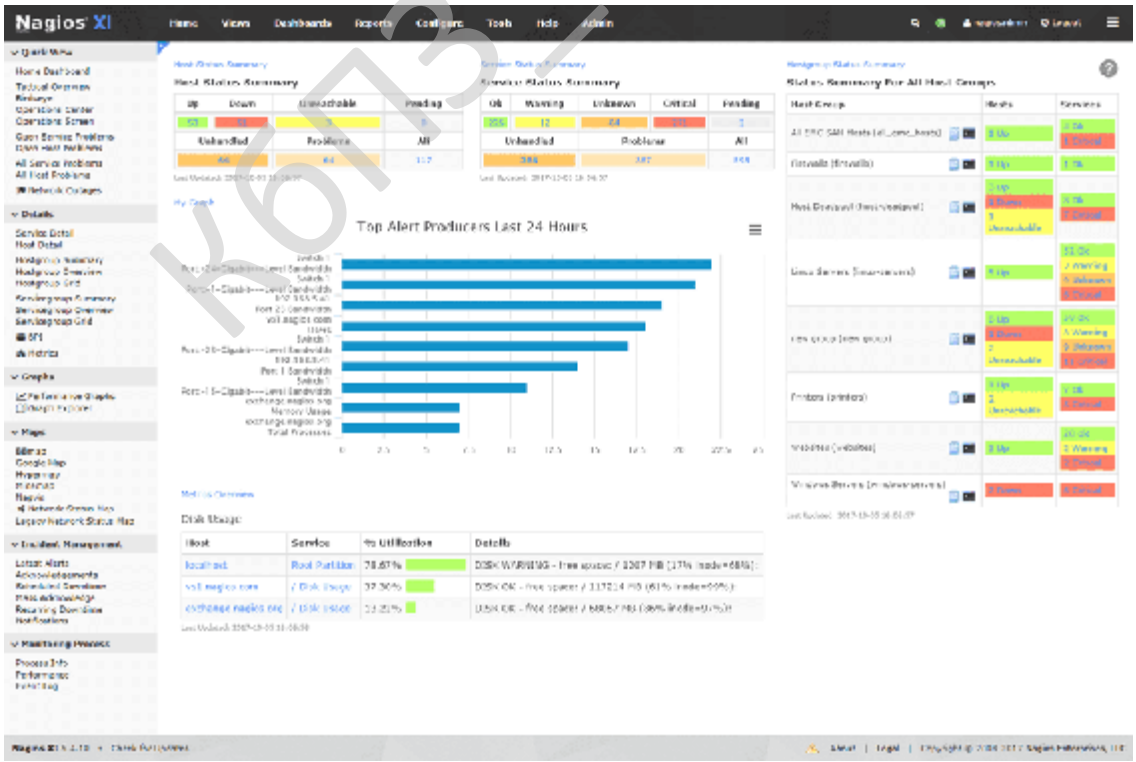

Рисунок 2.3 – Nagios Core

Основні характеристики:

– Велика екосистема плагінів.

– Потужна система оповіщення та оповіщення.

– Обробники подій для автоматичного виправлення.

– Можливість налаштування практично для будь-яких потреб моніторингу.

– Сильна підтримка спільноти та документація.

– Можливості роботи з кількома клієнтами.

Унікальна функція: широка екосистема Nagios Core із понад 5000 плагінів, розроблених спільнотою, дає змогу відстежувати практично будь-який пристрій, послугу чи показник.

Переваги:

– Повністю безкоштовне рішення з відкритим кодом.

– Необмежений потенціал налаштування.

– Ніяких витрат на ліцензування незалежно від розміру мережі.

– Зріла, стабільна платформа з обширною документацією.

Недоліки:

– Крута крива навчання, ніж комерційні рішення.

– Конфігурація переважно через текстові файли.

– Менш досконалий інтерфейс, ніж комерційні альтернативи.

– Потребує більше ручного налаштування та обслуговування.

Zabbix

Zabbix надає можливості моніторингу корпоративного рівня в пакеті з відкритим кодом, пропонуючи складні функції моніторингу, візуалізації та автоматизації без витрат на ліцензування.

Рисунок 2.4 – Zabbix

Основні характеристики:

– Варіанти моніторингу на основі та без агентів.
– Автоматичне виявлення мережевих пристроїв.
– Архітектура розподіленого моніторингу.
– Налаштовувані інформаційні панелі та звітність.
– Потужні можливості кореляції подій.
– Конфігурація на основі шаблону.

Унікальна функція: потужний механізм кореляції подій Zabbix може аналізувати численні сповіщення, щоб визначити основні причини, значно зменшуючи шум сповіщень і прискорюючи вирішення проблеми.

Переваги:

– Безкоштовне рішення для моніторингу корпоративного рівня.
– Чудова масштабованість для великих мереж.
– Сильна інтеграція бази даних і продуктивність.
– Велика бібліотека шаблонів.

Недоліки:

– Складна початкова конфігурація.

– Крута крива навчання, ніж комерційні рішення.

– Веб-інтерфейс менш інтуїтивно зрозумілий, ніж нові інструменти.

– Вимагає ресурсів для дуже великих розгортань.

ManageEngine OpManager

OpManager забезпечує комплексний моніторинг мережі з акцентом на зручності використання та швидкому розгортанні, що робить його особливо придатним для малих і середніх організацій з обмеженими IT-ресурсами.

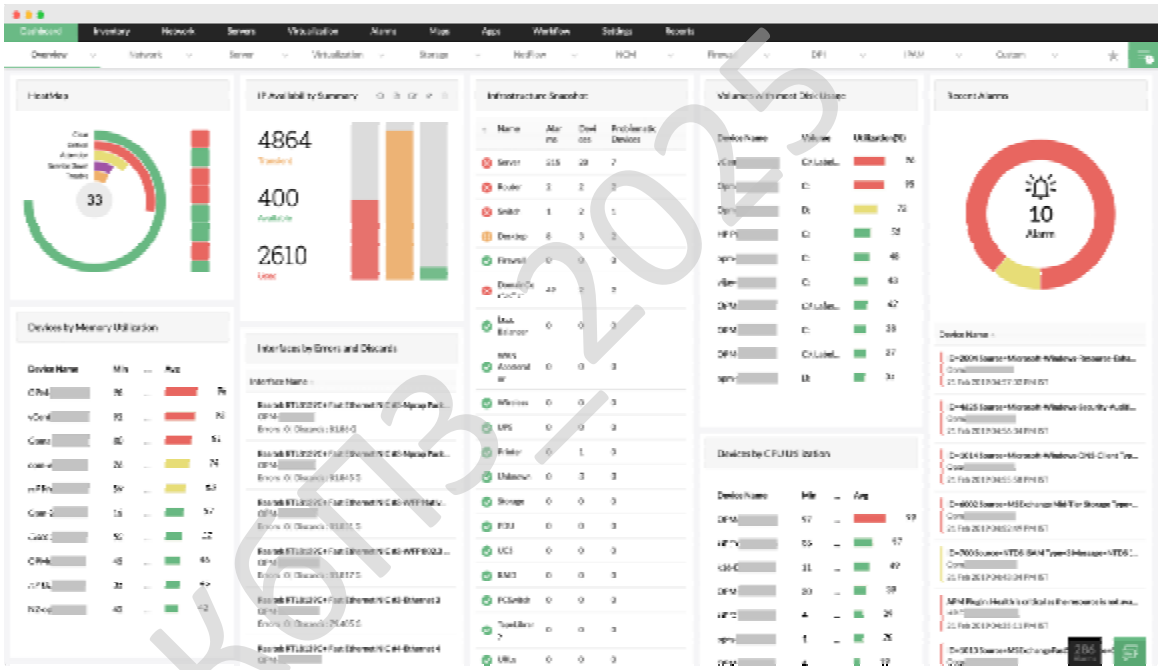

Рисунок 2.5 – ManageEngine OpManager

Основні характеристики:

– Моніторинг і сповіщення мережі в реальному часі.

– Вбудоване управління конфігурацією.

– Аналіз мережевого трафіку та моніторинг пропускної здатності.

– Автоматизоване виявлення та відображення пристроїв.

– Інтегроване керування IP-адресами.

– Автоматизація робочого процесу для звичайних завдань. Унікальна функція: інтегровані можливості OpManager для керування конфігурацією мережі дозволяють IT-командам керувати, створювати резервні копії та відстежувати зміни конфігурацій мережевих пристроїв за допомогою того самого інтерфейсу, який використовується для моніторингу продуктивності..

Переваги:

– Швидке розгортання з мінімальною конфігурацією.

– Інтуїтивно зрозумілий інтерфейс, що вимагає менше технічних знань.

– Відмінне співвідношення ціни та якості.

– Сильна інтеграція з іншими продуктами ManageEngine.

Недоліки:

– Менша глибина розширених функцій моніторингу.

– Більш обмежені налаштування, ніж корпоративні рішення.

– Звітність менш гнучка, ніж спеціалізовані інструменти.

– Деякі розширені функції залежать від плагінів.

Безкоштовна пробна версія протягом 30 днів.

Dynatrace

Dynatrace використовує моніторинг на основі штучного інтелекту, щоб надати виняткову інформацію про продуктивність мережі як частину уніфікованої платформи спостереження для сучасних хмарних і гібридних середовищ.

Основні характеристики:

– Аналіз першопричин за допомогою ШІ.

– Моніторинг повного стека за межами мереж.

– Автоматизоване виявлення та відображення залежностей.

– Візуалізація топології в реальному часі.

– Потужна аналітика та звітність.

– Унікальна функція хмарної архітектури: ШІ Davis® від Dynatrace автоматично визначає першопричину проблем із продуктивністю в складних мережах, значно скорочуючи середній час вирішення.

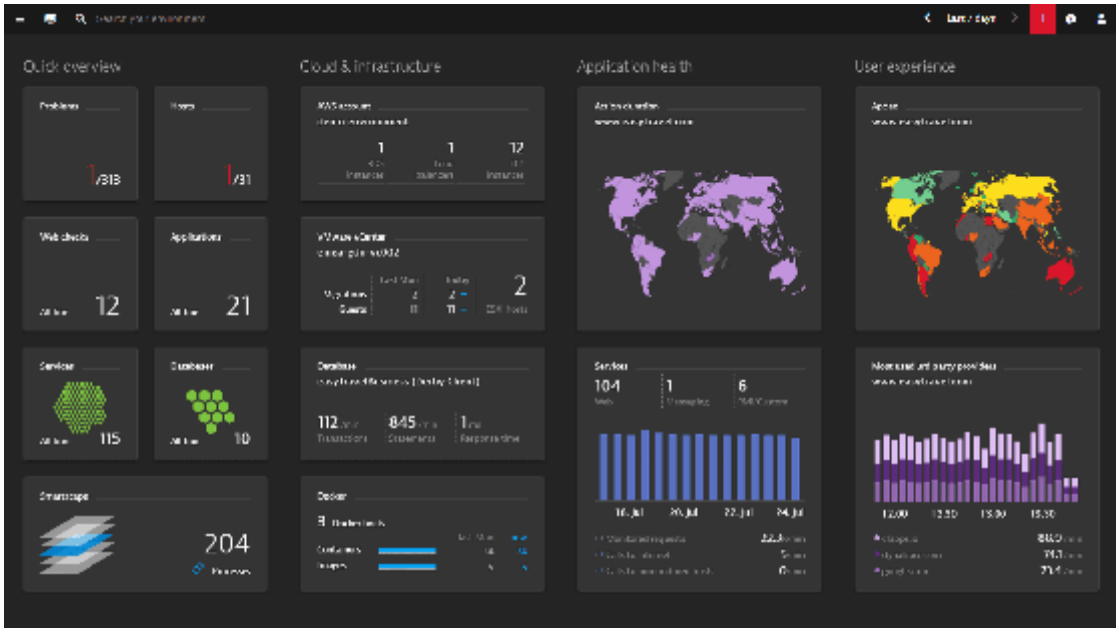

Рисунок 2.6 – Dynatrace

Переваги:

– Потужна аналітика та усунення несправностей на основі ШІ.
– Виняткова видимість у гібридних середовищах.
– Автоматизоване виявлення та відображення залежностей.
– Чудово підходить для сучасних хмарних архітектур.

Недоліки:

– Вища ціна, ніж традиційні засоби моніторингу.
– Більш складні, ніж чисті рішення для моніторингу мережі.
– Може вимагати значної початкової конфігурації.
– Потенційно вражає своєю широтою можливостей.

Безкоштовна пробна версія на 15 днів.

WhatsUp Gold

WhatsUp Gold пропонує доступний мережевий моніторинг із потужними можливостями візуалізації, що робить його особливо придатним для організацій середнього розміру, які потребують швидкого розгортання та мінімальної складності.

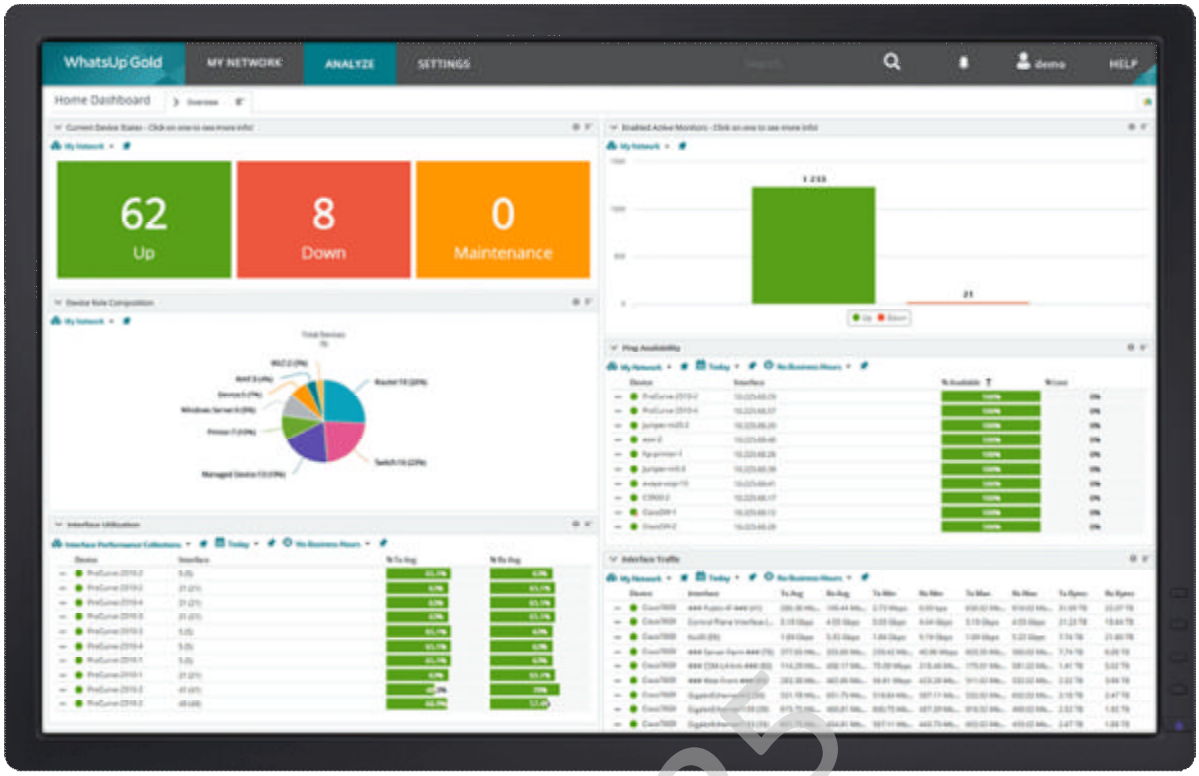

Рисунок 2.7 – WhatsUp Gold

Основні характеристики:

– Інтерактивне відображення мережі.
– Виявлення мережі рівня 2/3.
– Моніторинг бездротової мережі.
– Автоматичне сповіщення та сповіщення.
– Хмарний моніторинг ресурсів.
– Інтеграція керування конфігурацією.

Унікальна функція: інтерактивні карти мережі WhatsUp Gold забезпечують інтуїтивно зрозумілу візуалізацію топології мережі з індикаторами стану в реальному часі, що дозволяє адміністраторам швидко виявляти та усувати проблеми за допомогою візуального інтерфейсу..

Переваги:

– Інтуїтивно зрозумілий, зручний інтерфейс.
– Швидке розгортання та налаштування.

– Сильні можливості візуалізації.

– Ефективний моніторинг без надзвичайної складності.

Недоліки:

– Менш масштабована для дуже великих підприємств.

– Менше додаткових функцій, ніж корпоративні рішення.

– Менш розширюваний, ніж альтернативи з відкритим кодом.

– Обмежене налаштування для спеціалізованих середовищ.

OpenNMS

OpenNMS забезпечує мережевий моніторинг корпоративного рівня в пакеті з відкритим вихідним кодом, з особливо сильними перевагами у великомасштабних середовищах, які вимагають широких можливостей налаштування та інтеграції.

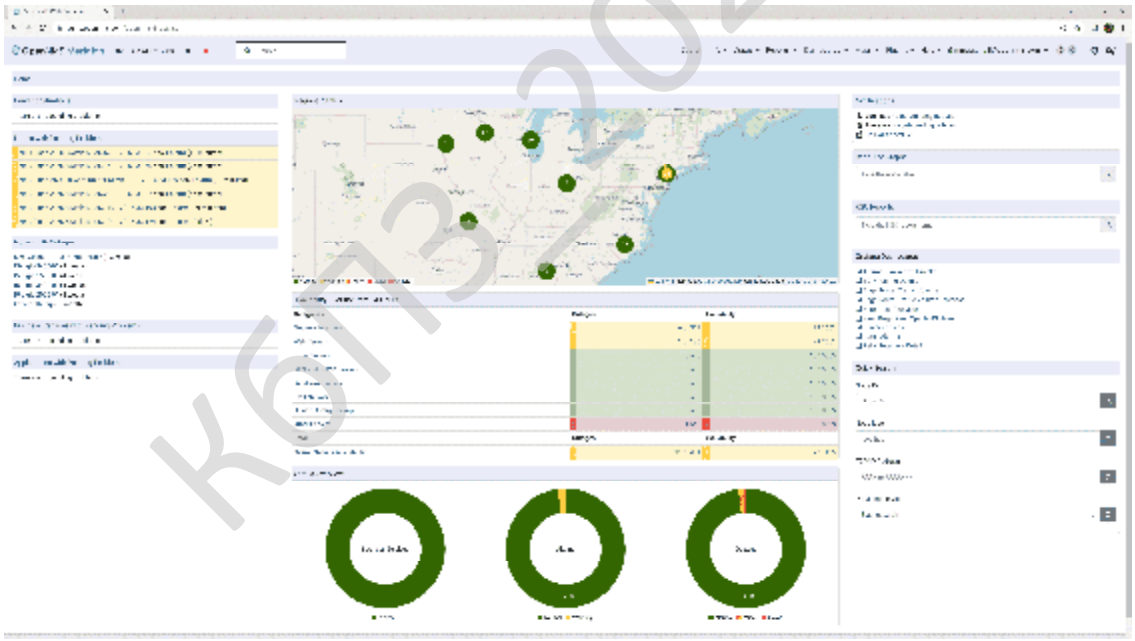

Рисунок 2.8 – OpenNMS

Основні характеристики:

– Розподілена архітектура з високою масштабованістю.

– Продумана система організації заходів.

– Автоматизоване виявлення та надання.

– Збір даних про ефективність часових рядів.

– Оповіщення на основі порогового значення.

– Моніторинг бізнес-послуг

Унікальна функція: OpenNMS Meridian, комерційно підтримуваний дистрибутив, забезпечує стабільність підприємства, зберігаючи при цьому гнучкість і економічні переваги програмного забезпечення з відкритим кодом..

Переваги:

– Безкоштовне рішення з відкритим вихідним кодом із корпоративними можливостями.

– Виняткова масштабованість для великих мереж.

– Потужна архітектура на основі Java.

– Спільнота активного розвитку.

Недоліки:

– Крута крива навчання для впровадження.

– Складна початкова конфігурація.

– Менш інтуїтивно зрозумілий інтерфейс, ніж комерційні альтернативи.

– Для максимізації потрібні значні технічні знання.

Icinga

Створений як форк Nagios, Icinga поєднує в собі стабільність свого попередника з сучасними інтерфейсами, покращеним підключенням до бази даних і розширеними можливостями звітування.

Основні характеристики:

– Сучасний адаптивний веб-інтерфейс.

– Архітектура розподіленого моніторингу.

– Гнучка система сповіщень.

– REST API для інтеграції.

– Моніторинг бізнес-процесів.

– Конфігурація за допомогою веб-інтерфейсу користувача або файлів.

Унікальна функція: модульна архітектура Icinga дозволяє організаціям вибирати лише ті компоненти, які їм потрібні, створюючи компактне та ефективне рішення для моніторингу, адаптоване до конкретних вимог.

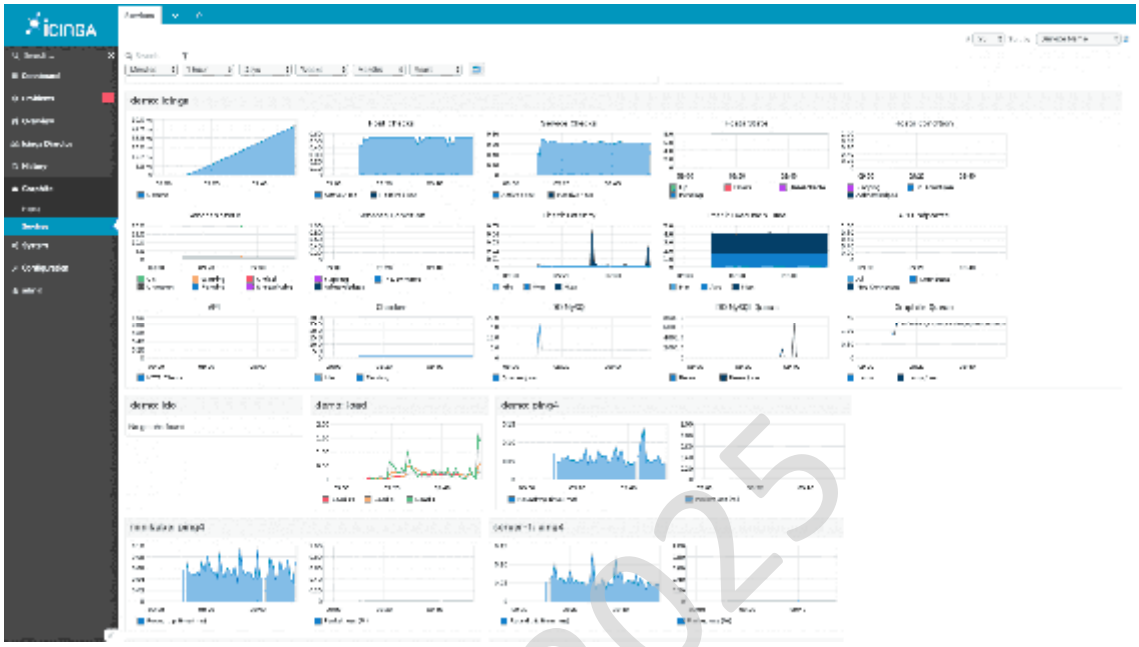

Рисунок 2.9 – Icinga

Переваги:

– Безкоштовне рішення з відкритим кодом.
– Більш сучасний інтерфейс, ніж Nagios.
– Краща інтеграція з базою даних, ніж його попередник.
– Потужний API для інтеграції та автоматизації.

Недоліки:

– Усе ще складно налаштувати порівняно з комерційними інструментами.
– Для правильного налаштування потрібні технічні знання.
– Менш повна документація, ніж Nagios.
– Менша екосистема плагінів, ніж більш усталені інструменти.

Netdata

Netdata робить революцію в моніторингу мережі завдяки своєму підходу до нульової конфігурації та збору показників у реальному часі, надаючи виняткову інформацію про продуктивність з мінімальними витратами на налаштування.

Рисунок 2.10 – Netdata

Основні характеристики:

– Посекундний моніторинг у реальному часі.
– Розгортання без конфігурації.
– Мінімальний слід ресурсів.
– Тисячі показників збираються автоматично.
– Інтерактивні інформаційні панелі з можливістю налаштування.
– Вбудований веб-сервер для миттєвого доступу.

Унікальна функція: унікальна архітектура Netdata в режимі реального часу збирає та візуалізує тисячі показників за секунду з незначним впливом на систему, забезпечуючи безпрецедентну деталізацію для усунення несправностей..

Переваги:

– Надзвичайно легке розгортання.
– Мінімальні вимоги до конфігурації.
– Надзвичайно легке використання ресурсів.
– Неперевершена видимість у реальному часі.

Недоліки:

– Менш зрілі, ніж усталені корпоративні інструменти.
– Менше розширених функцій для складних середовищ.
– Обмежене зберігання історичних даних у базовій версії.
– Менш повна підтримка пристроїв, ніж спеціалізовані інструменти.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це потужна мова програмування, яка проста у вивченні. Він має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис і динамічна типізація Python разом з його інтерпретованим характером роблять його ідеальною мовою для створення сценаріїв і швидкої розробки додатків у багатьох сферах на більшості платформ.

Інтерпретатор Python і обширна стандартна бібліотека доступні у вихідному або двійковому вигляді для всіх основних платформ на веб-сайті Python https://www.python.org/ і можуть вільно поширюватися. Цей же сайт також містить дистрибутиви та вказівники на багато безкоштовних сторонніх модулів Python, програм і інструментів, а також додаткову документацію.

Інтерпретатор Python легко розширюється за допомогою нових функцій і типів даних, реалізованих у C або C++ (або інших мовах, які можна викликати з C). Python також підходить як мова розширення для налаштовуваних програм.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи передачі інформації в комп'ютерних мережах систем критичного застосування.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

ІТ-індустрія втягнута в дискусію про переваги та небезпеки розміщення критично важливих програм у загальнодоступній хмарі . Незважаючи на те, що багато компаній використовують хмарне сховище для резервування своїх критично важливих програм, питання про те, чи розміщувати критично важливі програми в хмарі, залишається суперечливим .

Це рішення має прийматися в кожному конкретному випадку залежно від розміру організації, передбачуваних ризиків і особливостей застосування. Вибір залежить від багатьох змінних, включаючи відповідність нормативним вимогам, безпеку, продуктивність і доступність. Регуляторні фактори, які слід враховувати, включають державні зобов’язання або закони, які обмежують місця розміщення та зберігання програм і даних.

Технологія загальнодоступної хмари досягла зрілості в сферах безпеки, продуктивності та доступності з моменту свого створення. Оскільки постачальники публічних хмарних послуг (CSP), такі як Amazon Web Services (AWS), Google Cloud і Microsoft Azure, зростають і розширюють свої можливості, безпека хмарного хостингу стає менш занепокоєною.

У деяких випадках організація може заощадити гроші, покладаючись на послуги безпеки CSP, а не інвестуючи в спеціалізовані інструменти та персонал. Деякі організації вважають за краще контролювати всю ІТ-інфраструктуру для критично важливих додатків, щоб забезпечити доступність ресурсів для оптимальної продуктивності. Інші звертаються до CSP, вказуючи необхідну потужність і масштабованість, залишаючи CSP керувати інфраструктурою та ресурсами.

Іншим важливим фактором для організацій, які думають про перенесення

своїх критично важливих програм у хмару, є доступність, яка стосується здатності CSP підтримувати послуги в робочому стані. Великі загальнодоступні CSP обіцяють високу доступність – понад 99% часу безвідмовної роботи – що може перевищувати можливості окремих IT-організацій, які керують центрами обробки даних. Однак, якщо служби загальнодоступного CSP стають недоступними, організації користувачів не можуть вирішити цю проблему.

3.2 Розробка структурної схеми

Ви керуєте бізнесом, який використовує комп'ютерні мережі? Ви знаєте, що таке критична програма? Якщо ні, то ви в правильному місці. У сучасному світі функціонування бізнесу залежить від комп'ютерних мереж. Ці мережі відповідають за все, від спілкування електронною поштою до фінансових операцій. Однак не всі програми однакові. Деякі програми важливіші за інші, і вони відомі як критичні програми.

Критична програма – це програма, яка є важливою для функціонування бізнесу чи організації. Ці програми зазвичай використовуються для критично важливих завдань, таких як фінансові операції, обслуговування клієнтів і управління запасами. Якщо критично важлива програма виходить з ладу, це може мати серйозні наслідки для бізнесу, включаючи втрату доходу, зниження продуктивності та шкоду репутації компанії.

Деякі поширені приклади критичних програм включають:

– Системи фінансових операцій

– Програмне забезпечення для управління взаємовідносинами з клієнтами (CRM).

– Системи планування ресурсів підприємства (ERP).

– Системи управління запасами

– Системи електронних медичних записів (EHR).

Ці програми мають важливе значення для функціонування бізнесу чи організації та зазвичай використовуються для критично важливих завдань, таких як фінансові операції, обслуговування клієнтів і управління запасами. Якщо будь-яка з цих критично важливих програм виходить з ладу, це може мати серйозні наслідки для бізнесу, зокрема втрату прибутку, зниження продуктивності та шкоду репутації компанії.

Критичні програми важливі, оскільки вони є джерелом життя бізнесу. Вони забезпечують безперебійну роботу бізнесу та гарантують, що все працює належним чином. Без цих типів додатків підприємства не зможуть належним чином функціонувати, і це може мати серйозні наслідки для прибутку.

Виявлення критичних програм є важливою частиною керування комп'ютерною мережею. Існує кілька факторів, які слід враховувати під час визначення критичних програм, зокрема важливість програми для бізнесу, її вплив на інші програми та можливість простою.

Виявлення критичних програм передбачає оцінку важливості кожної програми для бізнесу, її впливу на інші програми та потенціал простою. Щоб визначити важливі програми, ви можете почати зі створення списку всіх програм, які використовуються у вашому бізнесі. Потім ви можете оцінити кожну заявку за такими критеріями:

1. Важливість для бізнесу: подумайте, наскільки важливою є програма для функціонування бізнесу. Програми, які використовуються для критично важливих завдань, таких як фінансові операції або обслуговування клієнтів, швидше за все, будуть важливішими, ніж програми, які використовуються для менш критичних завдань.

2. Вплив на інші програми: подумайте про те, як програма взаємодіє з іншими програмами у вашій мережі. Програми, тісно інтегровані з іншими програмами, можуть мати більший вплив на загальне функціонування вашої мережі.

3. Можливість простою: подумайте, скільки часу простою може витримати програма, перш ніж це почне мати негативний вплив на ваш бізнес. Програми, які більш чутливі до простою, можуть бути більш критичними, ніж програми, які можуть терпіти тривалі періоди простою.

Оцінюючи кожну програму на основі цих критеріїв, ви можете визначити, які програми мають вирішальне значення для функціонування вашого бізнесу, і відповідно розставити їм пріоритети. Це може допомогти вам більш ефективно розподіляти ресурси та гарантувати, що ваші критично важливі програми захищені від можливого простою або збою.

Захист критично важливих програм має важливе значення для забезпечення їхньої належної роботи. Є кілька кроків, які компанії можуть зробити, щоб захистити свої критично важливі програми, включаючи впровадження заходів із резервування, моніторинг показників продуктивності та регулярне резервне копіювання.

1. Щоб захистити критично важливі програми, компанії можуть вжити кількох заходів, щоб переконатися, що вони продовжують функціонувати належним чином і бути захищеними від можливого простою або збою. Ось деякі заходи, які бізнес може впровадити:

2. Запровадження заходів із резервування: це передбачає налаштування систем резервного копіювання або резервного обладнання, щоб гарантувати, що програми можуть продовжувати працювати, навіть якщо одна система виходить з ладу.

3. Відстежуйте показники продуктивності. Відстежуючи показники продуктивності, такі як час відгуку та пропускна здатність, компанії можуть виявити потенційні проблеми, перш ніж вони стануть критичними, і вжити заходів для запобігання простою.

4. Здійснюйте регулярне резервне копіювання: регулярне резервне копіювання критично важливих даних і програм може допомогти забезпечити їх швидке відновлення у разі збою або збою.

5. Застосуйте заходи безпеки: критичні програми часто містять конфіденційні дані, тому важливо впровадити заходи безпеки, такі як брандмауери, системи виявлення вторгнень і засоби контролю доступу, щоб захистити від несанкціонованого доступу або порушень даних. *Див* .: Поширені загрози безпеці

6. Навчання співробітників. Співробітники відіграють важливу роль у захисті критично важливих додатків, тому важливо забезпечити навчання найкращим практикам безпеки та захисту даних.

Впроваджуючи ці заходи, компанії можуть допомогти забезпечити належну роботу своїх критично важливих програм і захист від потенційних простоїв або збоїв.

Підсумовуючи, критично важливі програми є невід'ємною частиною будь-якого бізнесу, який покладається на комп'ютерні мережі. Вони забезпечують безперебійну роботу бізнесу та гарантують, що все працює належним чином. Розуміючи, що це за додатки та як їх захистити, компанії можуть гарантувати, що вони продовжуватимуть працювати належним чином, і уникнути будь-яких можливих простоїв або втрати прибутку.

Структурна схема методу забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування (КМ СКЗ) наведена на рисунку 3.1.

При рішенні задачі визначення множини $\aleph_{баз}$ шляхів передачі інформації в КМ СКЗ для вузлів зв'язку (ВЗ) «*i*» та «*j*» з множини $\Re$ вузлів зв'язку спочатку необхідно знайти найкоротшу «відстань» (мінімальний час передачі інформаційних пакетів) $T_{i,j\ min}$ від джерела «*i*» до адресата «*j*» і множини $S_j^{(i)}$ вузлів, найближчих ВЗ «*i*» за напрямом руху потоку до «*j*» (множина «вузлів-наступників») у порядку рівнів ієрархії дерева допустимих маршрутів множини U .

Комп’ютерна мережа системи критичного застосування

Блок реалізації методу забезпечення своєчасності передачі інформації в комп’ютерних мережах систем критичного застосування

Блок визначення множини шляхів передачі інформації в комп’ютерній мережі системи критичного застосування

Блок знаходження оптимальної множини маршрутів в комп’ютерній мережі системи критичного застосування

Блок розподілу інформаційних пакетів по знайденій множині шляхів передачі інформації в комп’ютерній мережі системи критичного застосування

Користувач системи

Рисунок 3.1 – Структурна схема системи

При рішенні поставленої задачі відомими алгоритмами пошуку найкоротших шляхів в більшості практичних випадків маємо проблему «зациклення» при передачі інформації в знайдених шляхах. Це призводить до збільшення часу передачі інформаційних пакетів, а деколи і до їх втрати. Уникнути «зациклення» при передачі інформації пропонується шляхом додання обмежень (умова постійної відсутності циклів), які надані у вигляді виразів:

$$T_{k,j} \le T_{i,j\,min};$$

$$T_{k,j\,min} \le T_{i,k,j},\ k \subset R,$$

де $T_{k,j\,min}$ – найкоротша «відстань» (мінімальний час передачі інформаційних пакетів) від вузла «*k*» до адресата «*j*»; $T_{i,k,j}$ – «відстань» (час передачі інформаційних пакетів) від вузла «*i*» до адресата «*j*» через вузол «*k*».

Рішення задачі пошуку множини шляхів, що виключають «цикли», складається з двох етапів: визначення найкоротшої «відстані» $T_{i,j\,min}$ від джерела «*i*» до адресата «*j*»; знаходження множини $S_j^{(i)}$ «вузлів-наступників» на маршрутах, що виключають «циклічність», для довільних джерел «*i*» та адресатів «*j*» за порядком множини U рівнів ієрархії дерева вибору допустимих маршрутів.

В подальшому, для передачі інформації про повітряну обстановку та забезпечення «збалансування» загрузки при флуктуаціях трафіку виконується процедура розподілу інформаційних пакетів за оптимальною множиною маршрутів в КМ СКЗ.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Так як функціональна схема є більш детальним описом структурної схеми, то логіка роботи основних функціональних блоків описана в пункті 3.2.

Рисунок 3.2 – Функціональна схема системи

Проведемо оцінки ефективності методу забезпечення своєчасності передачі інформації в КМ СКЗ при передачі інформації про повітряну обстановку і вірогідності отриманих результатів. В якості рекомендацій щодо практичного застосування методу забезпечення своєчасності передачі інформації в КМ СКЗ при управлінні повітряним рухом розроблена структура процесу передачі інформації про повітряну обстановку в режимі реального часу. Врахована необхідність здійснювати комплекс заходів (адаптивне кодування, зменшення

статистичної і психовізуальної надмірності, адаптивна маршрутизація та ін.), направлених на управління швидкістю передачі інформації про повітряну обстановку і зниження її інтенсивності. Для компенсації втрачених при передачі в каналах зв'язку інформаційних пакетів і забезпечення безперервності відтворення інформації про повітряну обстановку розроблено алгоритм компенсації втрачених інформаційних пакетів. Це дозволить використовувати для управління передачею цифрової інформації про повітряну обстановку протоколи транспортного рівня (RTP і UDP).

Оцінка ефективності роботи методу забезпечення своєчасності передачі інформації в КМ СКЗ по відношенню до відомих методів показала ряд його переваг (ймовірність доведення інформації до одержувача за час, що не перевищує допустиме значення, вище за аналогічну ймовірність відомих методів до 3 разів, середній час доставки інформаційних пакетів менше аналогічної характеристики відомих методів до 15 разів).

Для обґрунтування достовірності отриманих результатів математичного моделювання проведено імітаційне моделювання передачі інформації в повнозв'язній КМ в умовах застосування бездротових каналів зв'язку (середня пропускна спроможність $\rho_z = 19$Кбіт/с) при різних інтенсивностях вхідного потоку (у діапазоні $\lambda = [5,...,30]$Кбіт/с). За критерієм згоди Персона з рівнем значущості $\alpha = 0{,}01$ показано, що число прийнятих за час $T_{доп}$ інформаційних пакетів а також ймовірність $Q_{експ}^{(сч)}$ можна вважати розподіленими за нормальним законом.

Одержані довірчі інтервали для оцінок математичного очікування $Q_{експ}^{(сч)}$, в які потрапляють «розрахункові» значення $Q_{\lambda}^{(сч)}$ з довірчою ймовірністю 0,95. Високий ступінь збігу результатів імітаційного і математичного моделювання підтверджує достовірність математичної моделі, використаної для розрахунку ймовірності $Q^{(сч)}$ доставки інформаційних пакетів за час, що не перевищує допустиме значення.

3.4 Розробка діаграми процесів

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

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

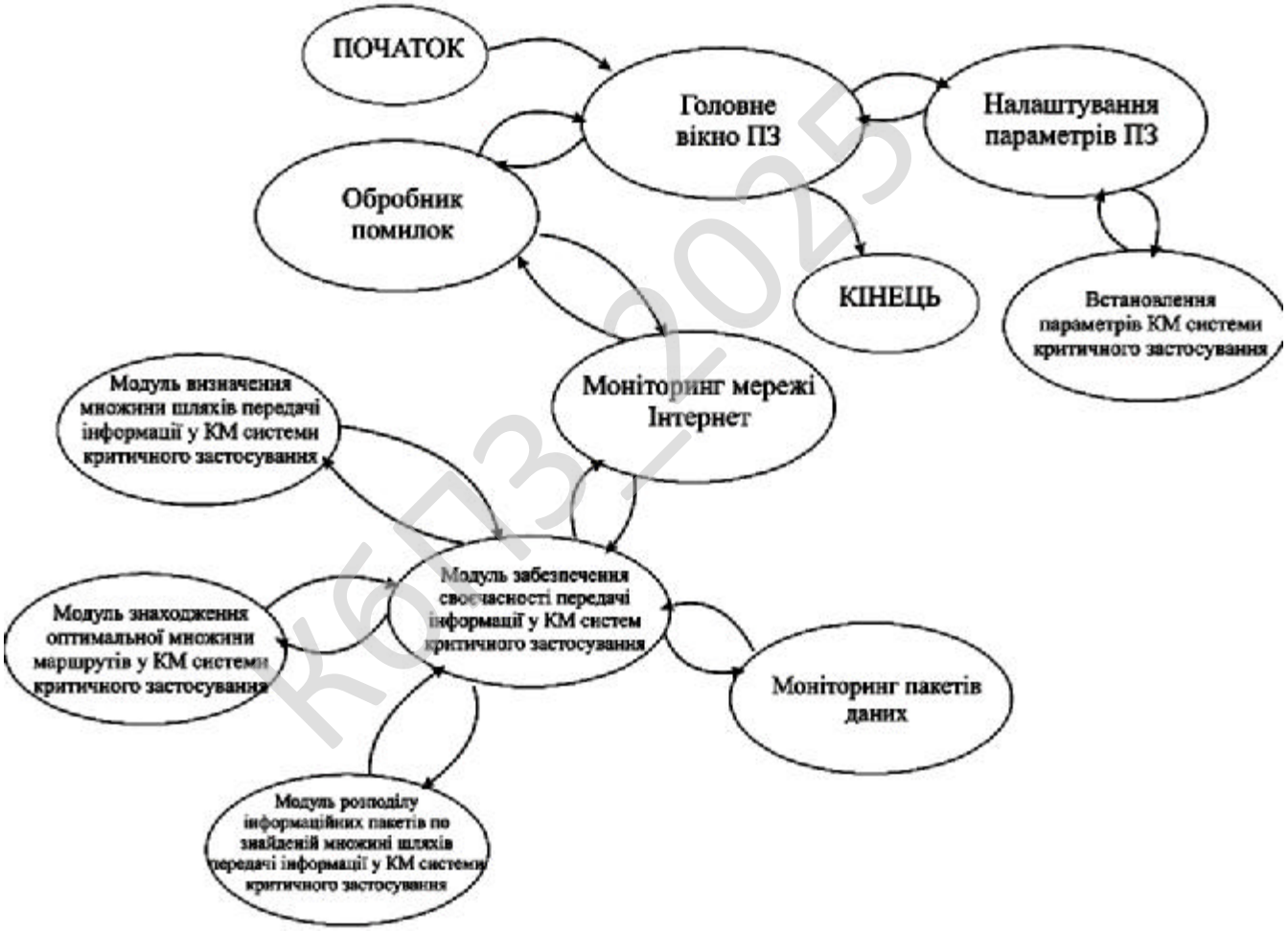

Рисунок 3.3 – Діаграма взаємодії процесів

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

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем.

На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

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

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми. При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю забезпечення своєчасності передачі інформації в комп’ютерних мережах систем критичного застосування. При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

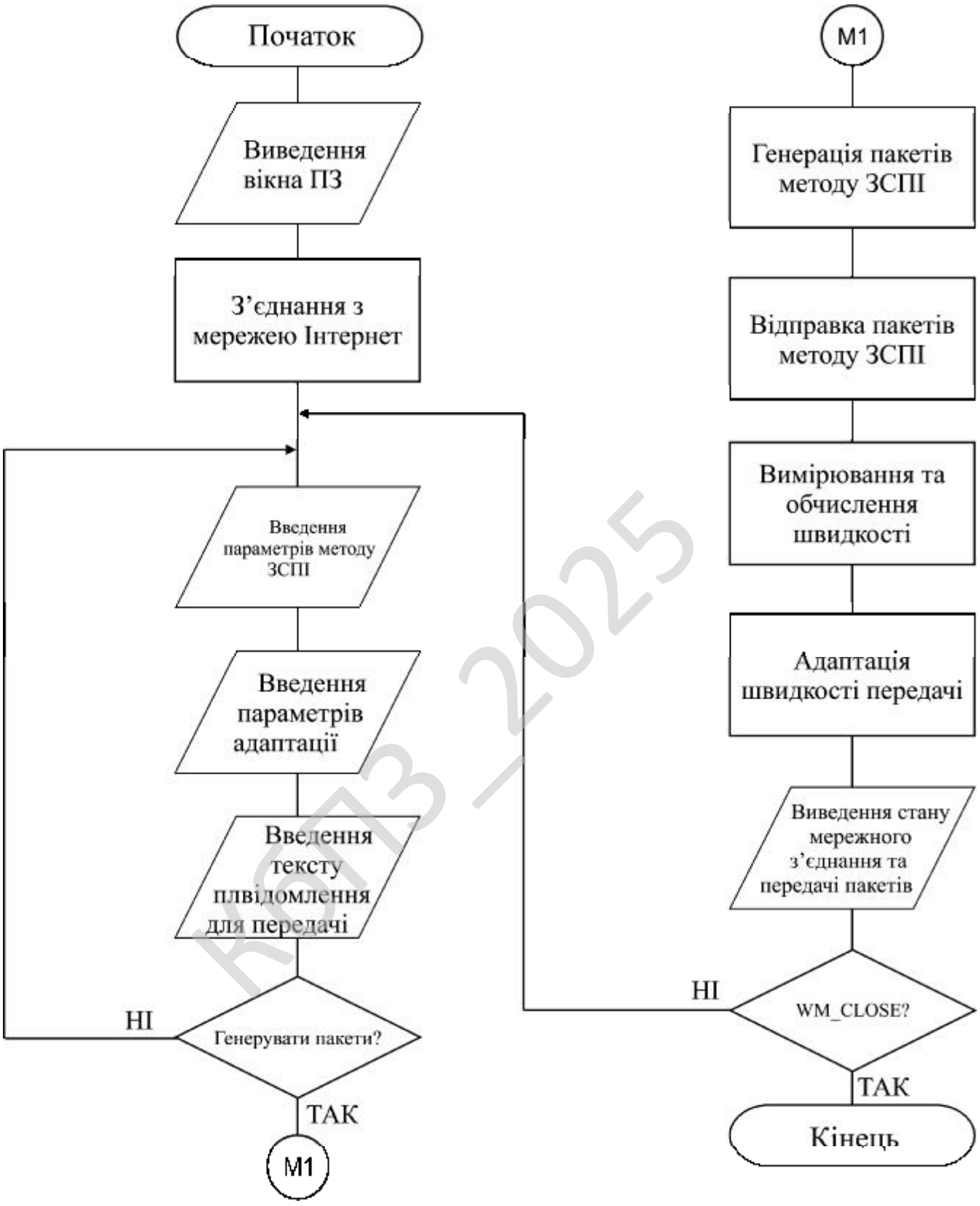

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування.

Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки.

Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють

значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

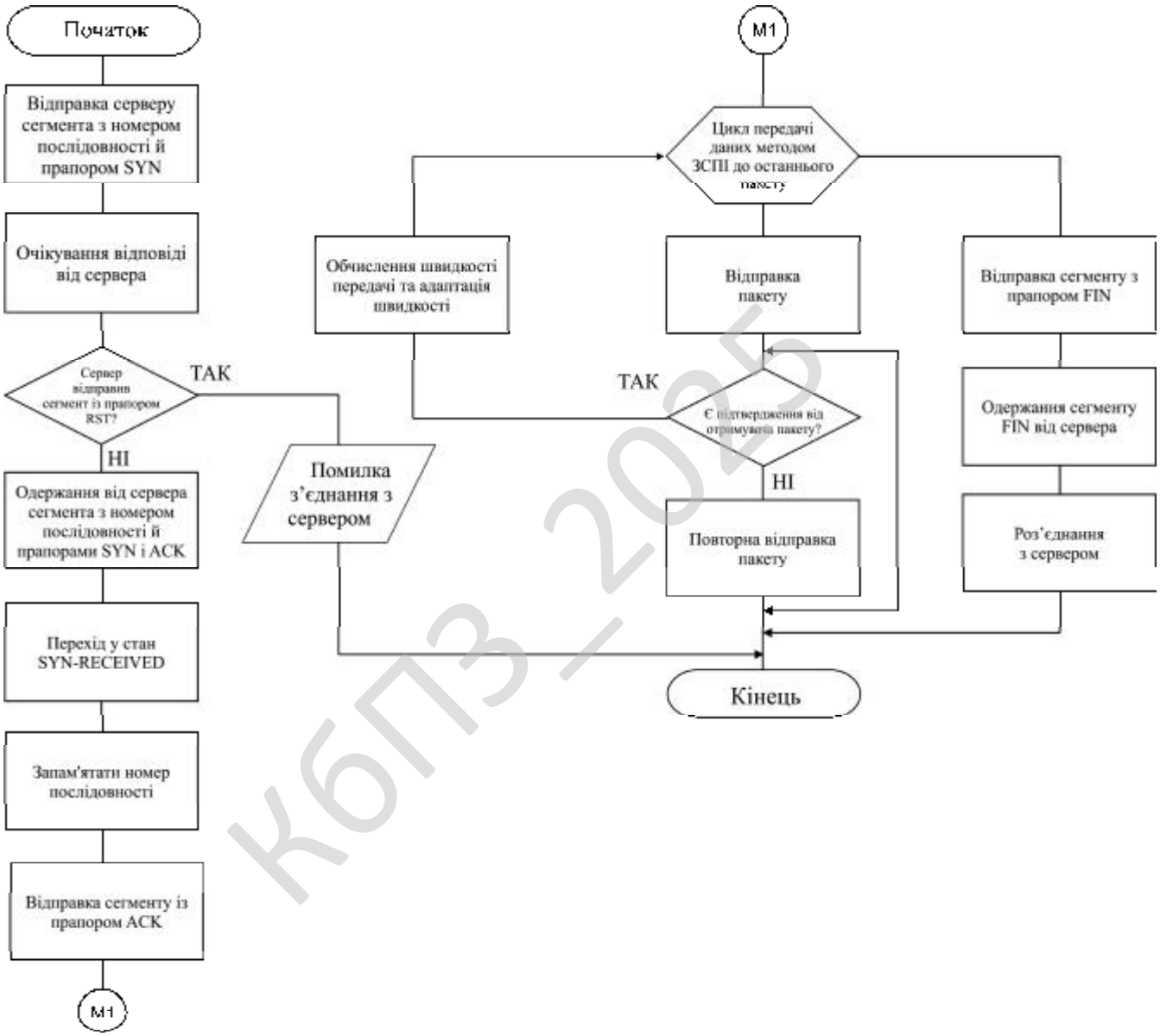

Рисунок 4.2 – Блок-схема роботи підпрограми

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Нові сегменти створюються усередині об'єктів протоколів ARTCP або CBR. Всі об'єкти топології передають один одному покажчик на область пам'яті, що містить відповідний сегмент, замість копіювання реальних даних. Крім того, розміри заголовка сегмента й поля корисного навантаження задаються в самій структурі даних, і всі об'єкти визначають розмір сегмента, інтерпретуючи значення цих полів.

Оскільки посилання на об'єкт може одночасно втримуватися у всіх об'єктах, де реалізована буферізація: ARTCP, link, interface, то щоб уникнути конфліктів лічильник посилань зберігає кількість об'єктів, у буферах, в яких перебуває посилання на даний сегмент.

Звільнення області пам'яті, що містить сегмент, дозволено тільки у випадку обнуління лічильника. За винятком полів "TI" і "PS" і службових полів link, id, всі інші поля сегмента відповідають полям стандартного TCP заголовка.

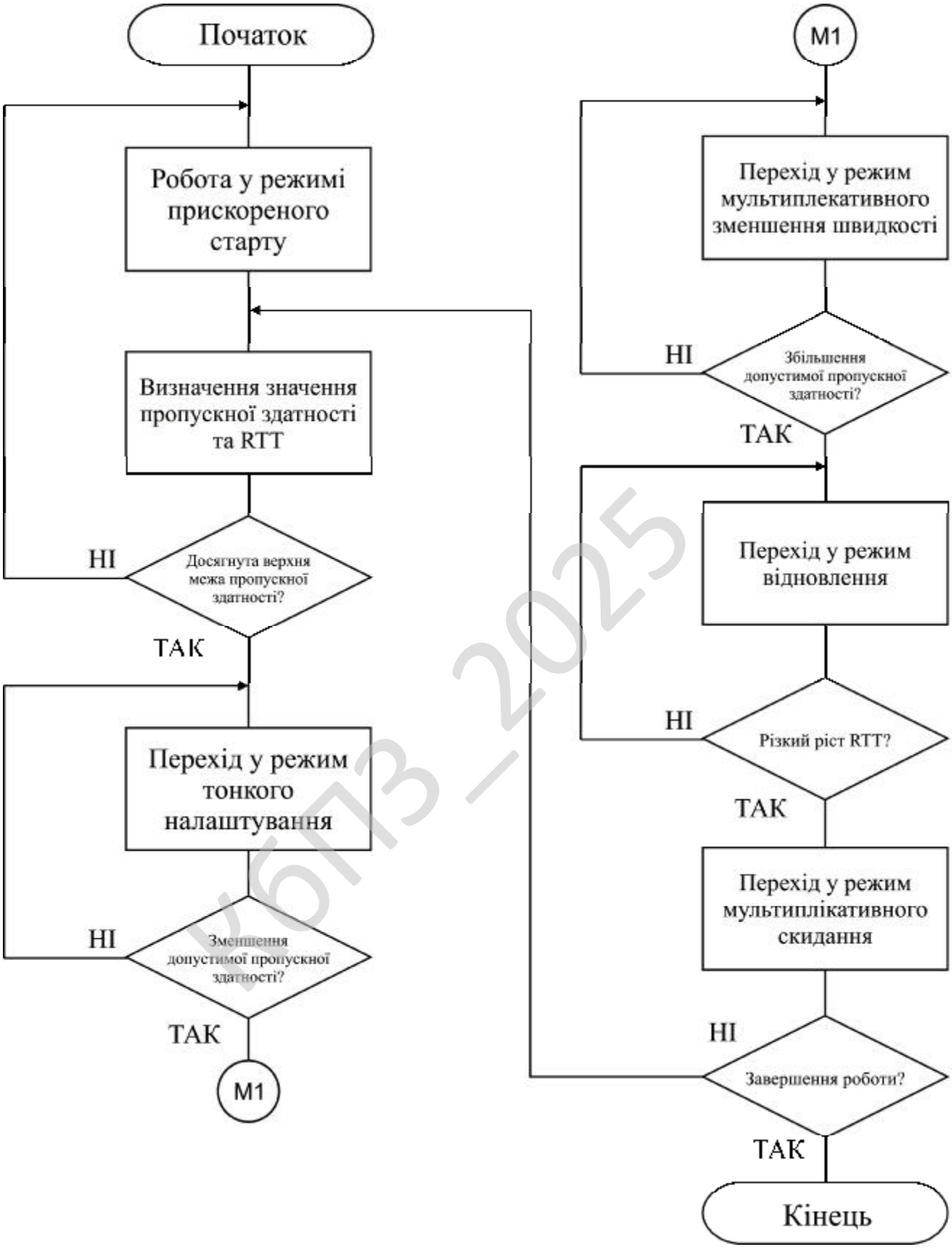

Рисунок 4.3 – Блок-схема роботи підпрограми

Поле заголовка ARTCP у моделі не містить перевірочної суми, ушкодження сегментів у транзиті можуть моделюватися шляхом реалізації випадкового відкидання сегментів на мережних вузлах або в кінцевій системі з імовірністю, що відповідає ймовірності бітових помилок середовища передачі, яку необхідно змоделювати.

Всі класи, що описують топологічні елементи мережі: кінцева система (host), канал (link), маршрутизатор (router), інтерфейс (interface), є спадкоємцями базового класу element.

Інтерфейс кожного із цих класів визначається трьома найважливішими компонентами:

– прийом сегмента;

– запит сервісу;

– обробка переривання.

Ці методи є перевизначенням віртуальних функцій базового класу. Крім того, кожний похідний клас розширений додатковими специфічними для нього методами. За допомогою цих елементів інтерфейсу моделюється передача даних у системі. Кожний зі спадкоємців класу element посилається на елемент, з яким він зв'язаний топологічно, по покажчику на базовий клас.

Екземпляр кожного з наведених класів використовує виклик топологічно прив'язаного до нього об'єкта для передачі йому сегмента. Залежно від наявності ресурсів метод викликуваного об'єкта може прийняти або не прийняти сегмент на обслуговування.

Метод кожного екземпляра викликається з основної програми для обробки черги й ухвалення рішення про обслуговування чергового сегмента.

Об'єднання елементів мережі в топологічну схему здійснюється за допомогою виклику спеціального методу для кожного екземпляра всіх класів.

Розглянемо клас link (аспекти реалізації). Клас, що моделює канал, одержує значення пропускної здатності й затримки передачі при ініціалізації.

Структура даних класу реалізується однозв'язною динамічною чергою, у яку містяться сегменти, прийняті на обслуговування.

Після початку обслуговування сегмента канал блокує наступні запити протягом часу $t=s/R$, де s – розмір сегмента, а R – швидкість каналу, тобто часу прийому сегмента розміру s у канал. У черзі каналу сегмент затримується протягом установленого часу затримки передачі. Після закінчення цього часу, об'єкт каналу викликає метод наступного об'єкта в топологічній схемі мережі. Якщо наступний об'єкт відмовляє в обслуговуванні сегмента, то цей сегмент відкидається. Для моделювання дуплексного зв'язку застосовуються 2 екземпляри каналу. Параметри каналів можуть бути різними для моделювання асиметричних систем.

Метод всіх екземплярів класу link викликається з головного циклу. Обробка переривання переводить внутрішній лічильник часу й викликає передачу готового сегмента з каналу наступному об'єкту топології.

Для моделювання помилок середовища передачі екземпляр класу link може бути ініційований ініціалі затором, що перевантажується, який крім параметрів швидкості й затримки каналу одержує також параметр відповідному резидентному значенню помилки передачі *BER*. З імовірністю $1-(1-BER)^s$ сегмент відкидається замість передачі наступному елементу топології. Таким чином, можна моделювати супутникові, радіо, стільникові канали.

Розглянемо клас router (аспекти реалізації). Структура класу router є складною. У його склад входять кілька екземплярів класу interface. При ініціалізації класу router йому передаються два параметри: кількість інтерфейсів і об'єм буферного простору (максимальна довжина черги), однаковий для всіх інтерфейсів. Об'єкт маршрутизатора створює задана кількість екземплярів класу interface. Покажчики на кожний інтерфейс розташовуються в спеціальному масиві, проіндексованому один по одному створення інтерфейсів. Після цього на конкретний інтерфейс можна посилатися як router.interface(X), де X – номер потрібного інтерфейсу.

Кожний інтерфейс при ініціалізації одержує покажчик на його об'єкт, що створив, класу router, по якому він може посилатися на методи класу router для передачі сегментів у матрицю комутації (позначена в схемах як "поле комутації").

При прийманні сегмента інтерфейс негайно передає його об'єкту маршрутизатора. Блокування при цьому відбутися не може, тобто моделюється не комутаційна матриця, що блокує. Об'єкт маршрутизатора вносить запис, що складається із двох полів – номер інтерфейсу та адреса відправника у таблицю маршрутизації.

Після цього зіставлення адреси призначення сегмента із другим полем таблиці маршрутизації дає номер вихідного інтерфейсу для даного сегмента.

При відсутності збігу із записами таблиці маршрутизації, сегмент передається для розсилання всім інтерфейсам маршрутизатора, крім того, з якого він був отриманий. Отриманий від комутуючої матриці, сегмент міститься у вихідну чергу інтерфейсу, якщо вільний простір у ній $Q_{max} - Q \geq s$, у противному випадку сегмент відкидається.

Черга моделюється списком подвійної зв’язності, вишиковується динамічно в пам'яті. Черга обслуговується зі швидкістю каналу, до якого підключений інтерфейс.

Метод обробки переривання кожного з інтерфейсів викликається методом обробки переривання об'єкта маршрутизатора. Також об'єкти маршрутизатора й інтерфейсу постачені методами для видачі звіту по поточних параметрах трафіку:

– середня швидкість;

– лічильники пропущених/відкинутих сегментів;

– таблиця маршрутизації;

– середня довжина черги.

Таким чином, об'єкт маршрутизатора моделює сучасний міжмережний пристрій з не комутаційною матрицею, що блокує, і буферизацією на виході. Можливо також розширення класу маршрутизатора алгоритмами RED і WFQ або CBQ.

Поводження запропонованої моделі маршрутизатора щодо побудови таблиці маршрутизації несуттєво й збігається зі схемою функціонування комутатора ЛОМ. Таблиці маршрутизації заповнюються на підставі спостереження за трафіком, а не в результаті роботи алгоритмів маршрутизації, тому не потрібно ніякої конфігурації маршрутизатора, у процесі роботи він навчається топології мережі.

Розглянемо клас host (аспекти реалізації). Клас host має у своєму складі екземпляр класу ARTCP, що моделює сам протокол і екземпляр класу CBR, що моделює протокол передачі даних без підтверджень і керування потоком з постійної бітової швидкості. У кожний момент часу тільки один із протоколів може бути в активному стані.

При відправленні сегмента метод екземпляра класу host викликається одним із протоколів. У випадку прийому сегмента на обслуговування об'єктом каналу передачі, позитивне підтвердження вертається об'єкту протоколу, що викликав метод, і покажчик на область пам'яті, що зберігає сегмент передається об'єкту канального рівня.

У випадку відсутності можливості передати сегмент на канальному рівні, метод об'єкта host повертає негативне підтвердження. При ініціалізації екземпляра класу host він одержує мережну адресу. Окремо встановлюється мережна адреса призначення.

У випадку прийому сегмента від об'єкта каналу, метод передає покажчик протоколу обумовленому за значенням поля port заголовка сегмента. Сегменти, чия адреса призначення не збігається з адресою даного екземпляра вузла, відкидаються.

Метод класу host використовується для передачі запиту на обробку переривання об'єкту активного протоколу.

Розглянемо об'єкт протоколу CBR. Протокол постійної швидкості передачі прямо відповідає реальному режиму передачі даних з постійною бітовою швидкістю, наприклад CBR в ATM або (Circuit emulation services) CES в IP

мережах. Коректна взаємодія з таким режимом передачі даних є актуальною задачею будь-якого адаптивного транспортного протоколу в сучасних мережах з інтеграцією сервісів.

Саме тому протокол CBR включений у модель середовища виконання ARTCP. Протокол CBR генерує сегменти й відправляє їх у мережу із передвстановленою швидкістю R_{CBR} , тобто часові проміжки між моментами початку послідовних трансляцій становлять $\tau_{CBR} = s / R_{CBR}$. Підтвердження й, отже, ретрансляція сегментів і контроль швидкості не реалізується протоколом CBR.

Розглянемо об'єкт протоколу ARTCP. Модель ARTCP реалізує:

– керування швидкістю відправлення сегментів за допомогою механізму ковзного вікна й алгоритму ARTCP;

– корекцію помилок передачі не пов'язану з управлінням передачі;

– швидку ретрансляцію сегментів і стандартну ретрансляцію по спрацьовуванню ТПП.

Основні призначення методів класу ARTCP такі:

– застосовуються для початкової синхронізації з'єднання;

– обчислюють значення області компенсації;

– запитують результат відправлення сегмента вузлом. З них три останні є відкритими й становлять інтерфейс класу.

До складу класу ARTCP входять два екземпляри класу Queue, які реалізують прийомний і передавальний буфер і методи для маніпуляції з ними.

Розглянемо клас Queue. Даний клас реалізує схему стандартного керування потоком по методу ковзного вікна.

Клас містить динамічний список подвійної зв’язності, у який записуються покажчики на сегменти, поставлені в чергу разом з екземпляром класу таймера, що реалізує ТПП.

Методи класу Queue дозволяють здійснювати вставку сегмента із сортуванням, сканування списку, знаходження чергового сегмента готового до

передачі. Ретрансляція реалізована за допомогою зміни черговості готових до відправлення сегментів.

Розглянемо реалізацію обробки прийому. Сегмент, прийнятий вузлом, передається об'єкту ARTCP. Метод приймає сегмент і обробляє його як підтвердження, якщо сегмент містить поле підтвердження.

Параметри ARTCP обчислюються, якщо встановлено поле "TI" сегмента. Якщо номер підтверджуваного сегмента утворить безперервну послідовність із порядковими номерами сегментів, що перебувають у черзі передачі, то вся послідовність, крім підтверджуваного сегмента віддаляється із черги.

Наприклад, сегменти в черзі передачі: {*k*, *k*+1, *k*+2, *k*+3, *k*+4, ...} і вузол одержує підтвердження {k+3}, із черги віддаляються сегменти *k*, *k*+1, *k*+2.

Якщо сегмент містить дані, то він міститься в прийомну чергу, після чого прийомна черга сканується, з її убираються (передаються користувачеві) сегменти, отримані в безперервній послідовності порядкових номерів.

Потім генерується підтвердження наступного очікуваного одержувачем сегмента, у поля якого заносяться наступні значення: ack – кумулятивне підтвердження, ack-trig – номер останнього отриманого сегмента, advwnd – вільний простір у черзі прийому.

Наприклад, у черзі прийому перебувають сегменти {*i*+1, *i*+2, *i*+3}, після одержання *i-го* сегмента і його запису в чергу, з її віддаляються сегменти і, i+1, *i*+2, *i*+3 і значення кумулятивного підтвердження стає рівним *i*+4. Прототип контрольного сегмента з підтвердженням ставиться на чергу до передачі.

Розглянемо обробку переривання. Метод активного протоколу викликається з методу утримуючого його екземпляра класу вузла.

Даний метод обновляє значення внутрішнього лічильника часу екземпляра об'єкта протоколу, обчислює поточну швидкість передачі потоку в стані SS; у стані REC; у стані FT.

Обчисливши значення міжсегментного інтервалу, метод класу протоколу ARTCP визначає, чи пройшов цей час із моменту відправлення попереднього сегмента.

Якщо так, то запитується екземпляр черги передачі на предмет наявності в ній готового до відправлення сегмента.

Сегмент для відправлення береться із черги або створюється новий (у випадку відсутності готового сегмента в черзі), після чого в поля ack, acktrig, advwnd заносяться значення із прототипу контрольного сегмента, й сегмент передається об'єкту вузла для передачі.

Якщо передача успішна, то вузол, викликаючи метод, повідомляє про це об'єкт протоколу, що відзначає відповідний сегмент як відправлений і запускає ТПП для нього.

Розглянемо прискорення ретрансляції. Підтримка прискореної ретрансляції сегментів також включена в ПМ. Для цього об'єкт протоколу ARTCP відслідковує надходження дублюючих підтверджень.

Якщо послідовно приходять два підтвердження того самого і-го сегмента, то ARTCP здійснює ретрансляцію даного сегмента поза залежністю від значення його ТПП.

Після здійснення ретрансляції алгоритм переходить у стан, у якому прискорена ретрансляція не дозволена й залишається в цьому стані поки не прийде хоча б одне підтвердження наступні: (i+1)-го сегмента.

Розглянемо початкову синхронізацію. Початкове значення RTT необхідно для обчислення початкової (мінімальної) швидкості роботи з'єднання, рівної s байт за час RTT. Для здійснення цього виміру при відкритті нового з'єднання об'єкт протоколу ARTCP реалізує обмін сегментом спеціального типу із протилежною стороною.

При активації об'єкт ARTCP попадає в "не синхронізований" стан. Сегмент, позначений як SYN (по наявності відповідного прапора), відправляється від ініціюючої сторони обміну до одержувача.

Сегмент SYN несе порядковий номер, що не впливає на порядкові номери при обміні даними.

Одержувач сегмента SYN реагує відправленням сегмента із установленими прапорами SYN, ACK, де поле ack несе значення порядкового номера SYN сегмента.

Одержання SYN, ACK сегмента дає можливість відправникові виміряти час RTT і переводить з'єднання в стан "синхронізації", після чого починається обмін даними. Наявність ідентифікуючого SYN сегмент порядкового номера дає можливість розрізнити їхні можливі копії й правильно виміряти час RTT.

Синхронізація з'єднання може бути ініційована одночасно обома сторонами.

Для протоколу TCP, наприклад початкова синхронізація має місце при відкритті з'єднання й має на меті встановити ідентичні початкові значення змінних у контрольних блоках обох сторін з'єднання.

Розглянемо головний цикл. Головний цикл програми викликає метод обробки переривання всіх елементів топології моделі (host, link, router). У результаті емулирується хід внутрішніх годиників кожного з об'єктів, і моделюються процеси передачі даних. Також з головного циклу з конфигуруємою періодичністю викликаються процедури, що генерують звіти.

Дуплексний режим. Протокол ARTCP, як і TCP здатний здійснювати симетричний обмін інформацією з одного з'єднання.

У такому режимі контрольна інформація одержувача транслюється не окремими сегментами, а записується у відповідні поля сегментів з даними, що випливають у протилежному напрямку. Програмна модель передбачає наявність буфера під прототип заголовка контрольного сегмента.

Цей заголовок, що містить номер підтвердження, розмір вікна й іншу керуючу інформацію, формується після одержання чергового сегмента з даними. Однак замість негайного відправлення порожнього сегмента з підтвердженням, створеним із прототипу, модель перевіряє наявність сегмента з даними чекаючого

передачі і якщо сегмент такий сегмент є, то підтвердження передається разом з ним.

Інакше із прототипу контрольного сегмента створюється окремий сегмент не несучий даних, а лише передавальне підтвердження.

Трасування моделі. Кожний з об'єктів топології, таких як вузол, канал і інтерфейс маршрутизатора, при кожній операції із сегментом виводить запис у файл звіту.

Формат цього запису такий:

"{ + |-|d|s|r}" time elem src"->"dst "{-|D}{-|A}{-|s}" size seq"/"ack " ... " psn

psk ack trig adv wnd link seq/syn ack id

де:

– дія: + прийом до черги, – із черги, d відкидання, s відправлення протоколом, r прийом протоколом;

– time – час у секундах;

– elem – ідентифікатор об'єкта створившего дію;

– src – адреса відправника сегмента;

– dst – адреса одержувача сегмента;

– прапори: D дані, A підтвердження, s синхронізація;

– size – розмір сегмента в байтах;

– seq – порядковий номер;

– ack – номер кумулятивного підтвердження;

– psn – поле "PS";

– psk – поле "TI";

– ack_trig – номер сегмента викликавшого відправлення підтвердження;

– adv_wnd – розмір вікна одержувача в байтах;

– link – лічильник посилань на область пам'яті;

– synack – номер підтверджуваного SYN сегмента;

– id – унікальний ідентифікатор кожного сегмента.

За допомогою інформації звіту можна простежити шлях кожного сегмента, тобто до якого вузла він дійшов, чи був він підтверджений, загублений і ретрансльований.

Дослідження системи що моделюється виробляється за допомогою візуалізації інформації звіту.

Крім подій, що відбуваються із сегментом, у звіт записуються також значення змінних протоколу ARTCP і характеристики, що знімаються з мережних пристроїв (з певних інтерфейсів).

На відміну від подій із сегментами, змінні ARTCP і дані з інтерфейсів маршрутизаторів знімаються із заданою періодичністю.

По кожному з подій звіт може виводитися як у короткій формі (наведеній вище для операцій із сегментами) так і в розгорнутому виді з метою налагодження.

Розглянемо розроблену систему візуалізації даних. Для візуалізації отриманих при моделюванні даних застосовувалася програма gnuplot версії 3.7 для ОС UNIX.

Як правило, візуалізація роботи протоколу TCP виробляється за допомогою графіків залежності розміру вікна CWND від часу, залежності послідовності передачі сегментів від часу, що дозволяє візуально визначити різні фази роботи протоколу. У переважній більшості робіт в області транспортних протоколів застосовується саме така схема.

У деяких роботах приводяться більш складні системи візуалізації. У цій роботі результати моделювання найбільше наочно представляються такими способами:

– залежність швидкості потоку;

– порядкового номера переданого сегмента;

– часу RTT, середньої довжини черги від часу.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм PRESENT – окремий випадок SP-мережі й складається з 31 раунду. Довжина блоку становить 64 біта, а ключі підтримуються в 2 варіантах, 80- і 128-бітні. Такого рівня захисту повинно цілком вистачати для низькозахищених додатків, звичайно використовуваних для розгортання на основі тегів, а крім того, що важливіше, PRESENT багато в чому збігається своїми конструктивними особливостями з потоковими шифрами проекту estream, заточеними на ефективну реалізацію в залозі, що дозволяє нам адекватно порівнювати їх.

Кожний з 31 раундів складається з операції XOR, щоб увести ключ K_i для $1 \leq i \leq 32$, де K_{32} використовується для «відбілювання» ключа, лінійної побітової перестановки й нелінійного шару заміщення (або, попросту говорячи, збільшення стійкості шифрування. Нелінійний шар використовує роздільні 4-бітні S-блоки, які застосовуються паралельно 16 раз на кожному раунді. Шифр, описаний псевдо-кодом представлено на рисунку 4.4.

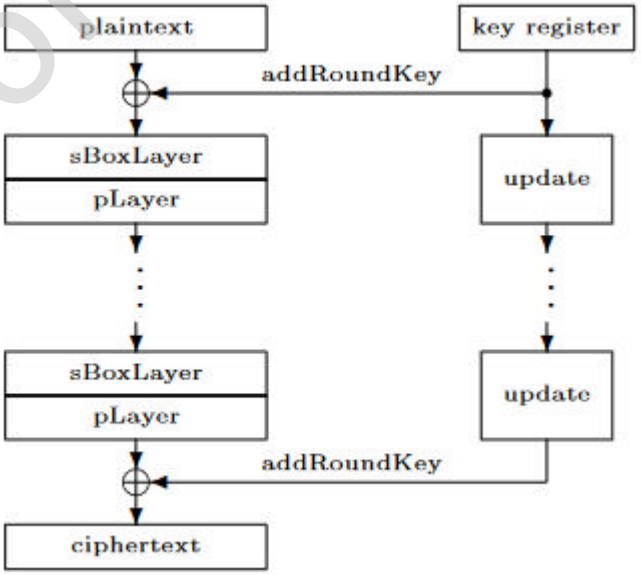

Рисунок 4.4 – Шифр PRESENT

Тепер кожна стадія визначається по черзі. Обґрунтування конструкції наведені нижче, а біти всюди нумеруються з нуля, починаючи із правого в блоці або слові.

Додавання раундового ключа (addRoundKey)

Заданий раундовий ключ $K_i = k^i_{63}\ldots k^i_0$, де $1 \le i \le 32$, а так само поточний стан $b_{63}\ldots b_0$. Додавання раундового ключа до поточного стану відбувається за модулем 2 ($b_j = b_j \oplus k^i_j$, де $0 \le j \le 63$).

Шар S-блоків (sBoxlayer)

Використовувані в PRESENT S-блоки відображають 4-бітні блоки в 4-бітні блоки. Дія цього блоку в шістнадцятковій системі числення наведене в наступній таблиці.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Рисунок 4.5 – Дія блоку в шістнадцятковій системі числення

Для шару S-блоків поточний стан $b_{63}\ldots b_0$ представляє із себе 16 4-бітних слів $w_{15}\ldots w_0$, де $w_i = b_{4*i+3} \,||\, b_{4*i+2} \,||\, b_{4*i+1} \,||\, b_{4*i}$ для $0 \le i \le 15$. Вихід рамки $S[w_i]$ видає оновлені значення станів очевидним образом.

Шар перестановки (player)

Побітова перестановка, використовувана в PRESENT задається наступною таблицею (біт і стану зміщається на позицію P(i).

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Рисунок 4.5 – Побітова перестановка, використовувана в PRESENT

Перетворення ключа (The key schedule)

PRESENT може використовувати 80- і 128-бітні ключі, однак зосередимося на 80-бітній версії. Надаваний користувачем ключ зберігається в регістрі ключів K, представлений у вигляді $k_{79}k_{78}\ldots k_0$. На i-ом раунді 64-бітний раундовий ключ $Ki = k_{63}k_{62}\ldots k_0$, що полягає з 64 лівих битов поточного вмісту регістру K. Таким чином, на i-ом раунді маємо:

$K_i = k_{63}k_{62}\ldots k_0 = k_{79}k_{78}\ldots k_{16}$.

Після розпакування раундового ключа K_i регістр ключа $K = k_{79}k_{78}\ldots k_0$ обновляється в такий спосіб:

1. $[k_{79}k_{78}\ldots k_1k_0] = [k_{18}k_{17}\ldots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus$ round_counter

Отже, регістр ключа зрушується на 61 позицію вліво, 4 крайніх лівих біта, що пройшли через S-блок і round_counter значення і складається за модулем 2 з бітами $k_{19}k_{18}k_{17}k_{16}k_{15}$ з K з найменшим значущим бітом з round_counter праворуч.

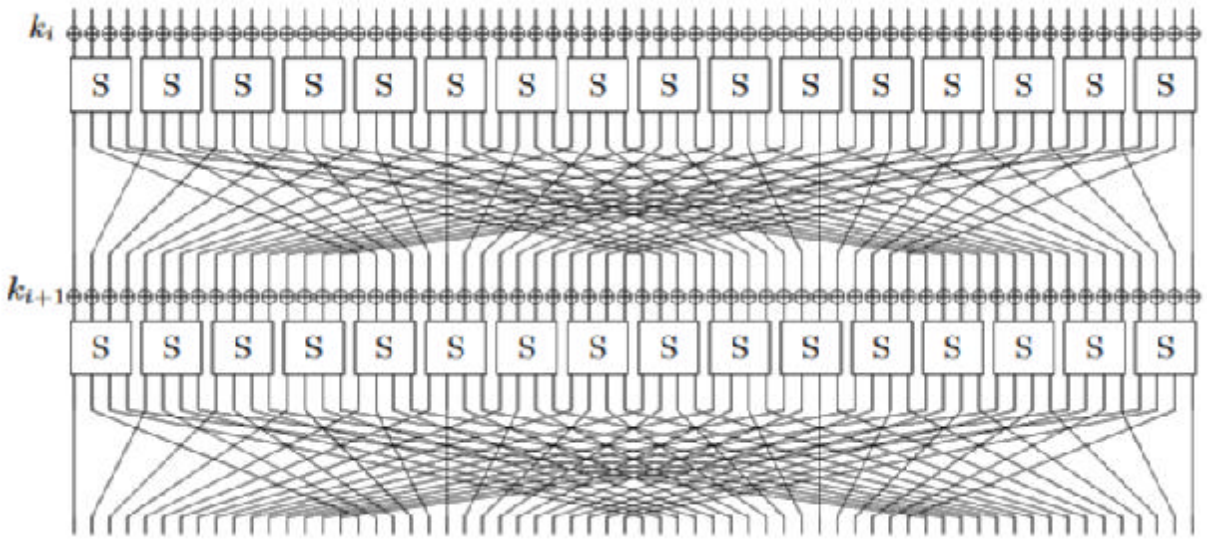

Рисунок 4.7 – Перетворення ключа (The key schedule)

Перетворення ключа для 128-бітного алгоритму можна знайти в додатку до оригінальної статті.

Конструктивні особливості PRESENT

Крім безпеки й ефективної реалізації, основне досягнення PRESENT – його простота. по цьому не дивно, що схожі проекти були прийняті в інших обставинах, і навіть були використані як навчальний посібник для студентів. У даній секції ми обґрунтуємо рішення, прийняті нами при проектуванні PRESENT. Однак, у першу чергу, опишемо очікувані прикладні вимоги.

Цілі й середовище застосування

При проектуванні блокового шифру, застосовного в жорстко обмежених оточеннях, важливо зрозуміти, що ми не створюємо блоковий шифр, який, неодмінно, буде застосовний у багатьох ситуаціях – для цього існує AES. Навпаки, ми націлені на досить специфічне застосування, для якого AES не підходить. Вищесказане визначає нам наступні характеристики:

– Шифр буде реалізований «у залізі»

– Додатки будуть вимагатися лише для регулювання рівня безпеки. Отже, 80-бітний ключ буде здоровим розв'язком. Відзначимо, що такої ж позиції дотримуються розроблювачі потокових шифрів проекту eSTREAM.

– Додатки не припускають шифровки великої кількості даних. Таким чином, реалізація може бути оптимізована для продуктивності або простору без внесення занадто великих змін.

– У деяких застосуваннях можлива ситуація, що ключ буде зафіксований при виробництві. У такому випадку, не треба буде змінювати ключ пристрою (що може вилитися в атаки з маніпуляцією ключем.

– Фізичний обсяг пристрою буде першим пріоритетом, після безпеки, що спричинить обмеження на пікові й середні споживання енергії, і, отже, зрушить швидкодія в область низькопріоритетних параметрів.

– У пристроях, що вимагають найбільш ефективного використання фізичного простору, блоковий шифр найчастіше зможе лише шифрувати дані (encryption-only mode). Таким чином, він зможе бути використаний у запит-відповідь (challenge-response) протоколах авторизації, і, при дотриманні контролю стану, може бути використаний для шифровки й дешифрування переговорів із пристроєм, використовуючи режим лічильника.

Виходячи з таких міркувань, розв'язали створити PRESENT як 64-бітний блоковий шифр із 80-бітним ключем. Шифровка й дешифрування, у цьому випадку, мають приблизно схожі фізичні вимоги. Маючи можливість підтримувати як шифрацію, так і дешифрацію, PRESENT буде компактніше, чим підтримуючий лише шифрацію AES. А у випадку encryption-only виконання, наш шифр виявиться й зовсім понад-легко. Суб-ключі що шифрують будуть обчислюватися на ходу.

У літературі є безліч прикладів атак компромісу між часом, датою й пам'яттю, або атак з використанням парадокса днів народження при шифровці великих обсягів даних. Однак, дані атаки залежать тільки від параметрів шифру й не використовують внутрішню структуру. Наша мета полягає в тому, щоб ці

атаки були кращим, що можуть застосувати проти нас. Атаки стороннього каналу й атаки з безпосереднім зломом чипа загрожують PRESENT тією самою мірою, як і іншим криптографічним примітивам. Однак для ймовірних застосувань, помірні вимоги безпеки роблять вигоду, одержувану зловмисником на практиці, досить обмеженої. В оцінці ризиків, подібні погрози не сприймаються як істотний фактор.

Перестановочний шар

При виборі шару змішування ключа, наша увага до апаратної ефективності вимагає наявності лінійного шару, який може бути реалізований з мінімальною кількістю керуючих елементів (наприклад, транзисторів. це приводить до побітової перестановки. Приділяючи увагу простоті, ми вибрали регулярну бітову перестановку, що допомагає провести прозорий аналіз безпеки.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської роботи. Розроблене програмне забезпечення своєчасності передачі інформації в комп'ютерних мережах систем критичного застосування складається з наступних функціональних блоків:

– Навігаційне меню: Файл; Дані; Додатки; Налаштування; Довідка.

– Функціонального підрозділу визначення множини шляхів передачі інформації в КМ системи критичного застосування.

– Функціонального підрозділу знаходження оптимальної множини.

– Функціонального підрозділу розподілу ІП по знайденій множині.

– Навігаційного меню.

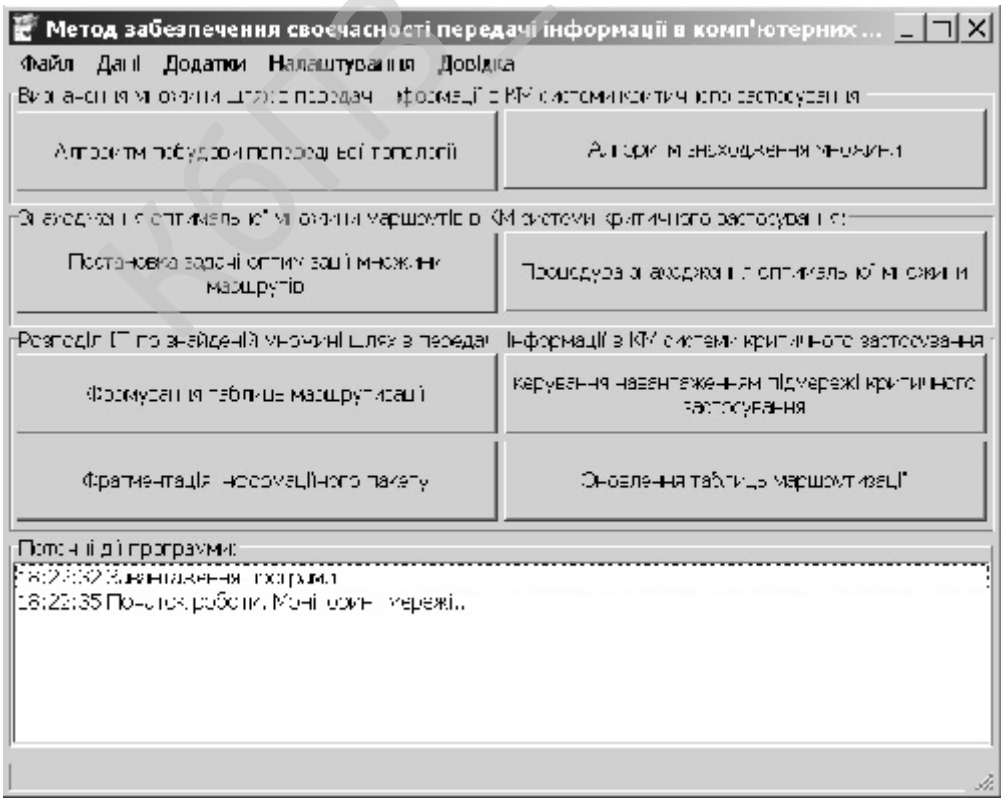

Рисунок 5.1 – Головне вікно розробленого ПЗ

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

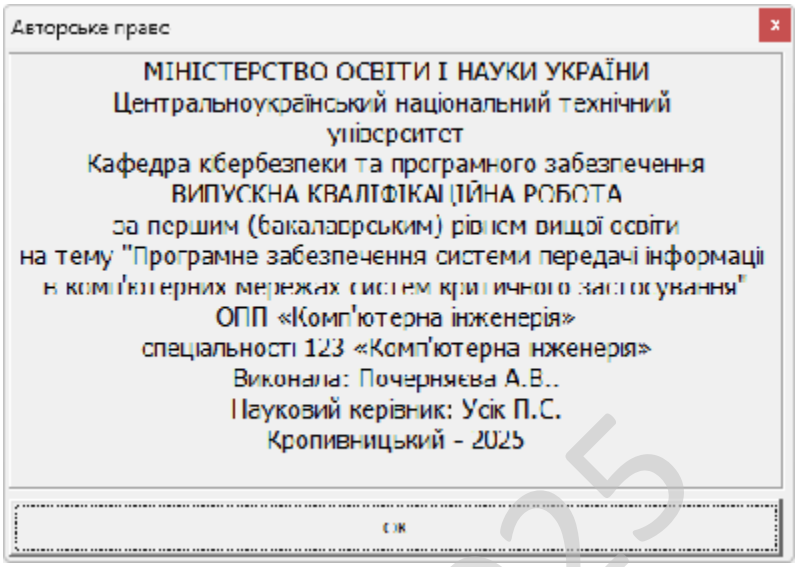

Рисунок 5.2 – Вікно розробника ПЗ

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі

можливі тести з урахуванням наявного часу та ресурсів. Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки.

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок. Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми. Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ІТс).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його ОТ).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ. Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Freeware.

Це власницьке програмне забезпечення, котре можна Безоплатно використовувати протягом необмеженого терміну без обмежень у функціональності, і поширюване без сирцевих кодів. Автори такого програмного забезпечення, як правило, хочуть «дати щось спільноті», але хочуть також контролювати його подальшу розробку. Іноді, коли програмісти вирішують припинити розробку, вони передають сирцевий код іншим програмістам, або ж спільності як вільне програмне забезпечення. Дуже часто плутають поняття «безплатне програмне забезпечення» та «вільне програмне забезпечення», хоча вони суттєво відрізняються. Безплатне програмне забезпечення можна безоплатно встановлювати та використовувати (іноді з певними обмеженнями, як, наприклад, «безплатне для домашнього або некомерційного вжитку»), в той час як вільне програмне забезпечення можна продавати за будь-яку суму, але при тому, у користувача, котрий його отримує, повинні бути права на вивчення, модифікацію та поширення сирцевих кодів одержаної програми.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи передачі інформації в комп'ютерних мережах систем критичного застосування.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем передачі інформації в комп'ютерних мережах систем критичного застосування.

– Досліджена система передачі інформації в комп'ютерних мережах систем критичного застосування.

– На основі отриманих результатів досліджень створена програмна реалізація системи передачі інформації в комп'ютерних мережах систем критичного застосування.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання передачі інформації в комп'ютерних мережах систем критичного застосування.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об'єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для

системи передачі інформації в комп'ютерних мережах систем критичного застосування. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм PRESENT.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

30. Смірнова Т.В., Конопліцька-Слободенюк О.К., Буравченко К.О., Смірнов С.А., Кравчук О.В., Козірова Н.Л., Смірнов О.А. «Дослідження технологій забезпечення кібербезпеки хмарних сервісів IaaS, PaaS та SaaS». *Кібербезпека: освіта, наука, техніка.* 2024. №4(24), С. 6-27.

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

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

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

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

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

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

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

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

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

40. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В., Поліщук Л.І. Інформаційна безпека в комп'ютерних мережах. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2020. – 294 с.

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

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

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

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

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

46. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для

моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

47. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

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

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

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

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

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

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи передачі інформації в комп’ютерних мережах систем критичного застосування.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 46-02 від 17.01.2025 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи передачі інформації в комп’ютерних мережах систем критичного застосування.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи передачі інформації в комп'ютерних мережах систем критичного застосування;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

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

5.4 Вимоги до архітектури

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

5.5 Вимоги до надійності

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

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

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

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

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

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 71 аркуш.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2025 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 6.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Усік П.С.

Програмне забезпечення системи передачі інформації в комп’ютерних мережах систем критичного застосування

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2025 року

Основна програма

```
import socket
import threading
import time
import hashlib
import json
import os
import sys
import ssl
import logging
import random
import base64
import queue

# Налаштування логування
logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s -
%(message)s")

class SecureCommunication:
    def __init__(self, certfile="server.crt", keyfile="server.key"):
        self.context = ssl.create_default_context(ssl.Purpose.CLIENT_AUTH)
        self.context.load_cert_chain(certfile, keyfile)

    def wrap_socket(self, sock, server_side=True):
        return self.context.wrap_socket(sock, server_side=server_side)

class DataIntegrity:
    def __init__(self):
        pass

    def compute_hash(self, data):
        return hashlib.sha256(data.encode()).hexdigest()

    def verify_hash(self, data, expected_hash):
        return self.compute_hash(data) == expected_hash

class CriticalNetworkSystem:
    def __init__(self, host="127.0.0.1", port=9000):
        self.host = host
        self.port = port
        self.server_socket = None
        self.clients = []
        self.integrity_checker = DataIntegrity()
        self.secure_comms = SecureCommunication()

    def start_server(self):
        try:
            self.server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
            self.server_socket.bind((self.host, self.port))
            self.server_socket.listen(5)
            self.server_socket =
self.secure_comms.wrap_socket(self.server_socket)
            logging.info(f"Сервер запущено на {self.host}:{self.port}")

            while True:
                client_socket, client_address = self.server_socket.accept()
                logging.info(f"Новий підключений клієнт: {client_address}")
                self.clients.append(client_socket)
                threading.Thread(target=self.handle_client,
args=(client_socket,)).start()
        except Exception as e:
```

```
            logging.error(f"Помилка сервера: {e}")
        finally:
            self.stop_server()

    def handle_client(self, client_socket):
        try:
            while True:
                data = client_socket.recv(1024).decode()
                if not data:
                    break
                received_hash = client_socket.recv(1024).decode()
                if self.integrity_checker.verify_hash(data, received_hash):
                    logging.info(f"Отримано коректні дані: {data}")
                    response = f"Дані отримані успішно: {data}"
                else:
                    logging.warning("Помилка цілісності даних!")
                    response = "Помилка перевірки цілісності!"
                client_socket.send(response.encode())
        except Exception as e:
            logging.error(f"Помилка при обробці клієнта: {e}")
        finally:
            client_socket.close()

    def stop_server(self):
        if self.server_socket:
            self.server_socket.close()
        logging.info("Сервер зупинено")

class Client:
    def __init__(self, host="127.0.0.1", port=9000):
        self.host = host
        self.port = port
        self.integrity_checker = DataIntegrity()
        self.secure_comms = SecureCommunication()

    def send_data(self, message):
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock = self.secure_comms.wrap_socket(sock, server_side=False)
            sock.connect((self.host, self.port))
            data_hash = self.integrity_checker.compute_hash(message)
            sock.send(message.encode())
            sock.send(data_hash.encode())
            response = sock.recv(1024).decode()
            logging.info(f"Отримана відповідь: {response}")
        except Exception as e:
            logging.error(f"Помилка клієнта: {e}")
        finally:
            sock.close()

class AuthenticationSystem:
    def __init__(self):
        self.users = {}
        self.load_users()

    def load_users(self):
        if os.path.exists("users.json"):
            with open("users.json", "r") as file:
                self.users = json.load(file)

    def save_users(self):
        with open("users.json", "w") as file:
            json.dump(self.users, file)
```

```
    def register_user(self, username, password):
        if username in self.users:
            return False
        hashed_password = hashlib.sha256(password.encode()).hexdigest()
        self.users[username] = hashed_password
        self.save_users()
        return True

    def authenticate_user(self, username, password):
        hashed_password = hashlib.sha256(password.encode()).hexdigest()
        return self.users.get(username) == hashed_password

class SecurityManager:
    def __init__(self):
        self.failed_attempts = {}

    def log_failed_attempt(self, username):
        if username not in self.failed_attempts:
            self.failed_attempts[username] = 0
        self.failed_attempts[username] += 1
        if self.failed_attempts[username] > 3:
            logging.warning(f"Блокування облікового запису: {username}")

    def is_account_locked(self, username):
        return self.failed_attempts.get(username, 0) > 3

class DataQueueManager:
    def __init__(self):
        self.queue = queue.Queue()

    def add_to_queue(self, data):
        self.queue.put(data)

    def get_from_queue(self):
        if not self.queue.empty():
            return self.queue.get()
        return None

class MonitoringSystem:
    def __init__(self):
        self.logs = []

    def log_event(self, event):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        self.logs.append(f"{timestamp} - {event}")

    def show_logs(self):
        for log in self.logs:
            print(log)

class DataEncryption:
    def encrypt_data(self, data, key):
        encoded = base64.b64encode(data.encode()).decode()
        return encoded[::-1]

    def decrypt_data(self, encrypted_data, key):
        decoded = base64.b64decode(encrypted_data[::-1].encode()).decode()
        return decoded

if __name__ == "__main__":
    auth_system = AuthenticationSystem()
    security_manager = SecurityManager()
```

```
monitoring_system = MonitoringSystem()
encryption_system = DataEncryption()

if auth_system.register_user("admin", "securepassword"):
    logging.info("Користувач admin зареєстрований")

if auth_system.authenticate_user("admin", "securepassword"):
    logging.info("Аутентифікація успішна")

server = CriticalNetworkSystem()
threading.Thread(target=server.start_server).start()

client = Client()
message = "Передача конфіденційних даних"
encrypted_message = encryption_system.encrypt_data(message, "key123")
client.send_data(encrypted_message)
```

Файл SecureFileServe.py

```
import socket
import threading
import time
import hashlib
import json
import os
import ssl
import logging
import sqlite3
import random
import base64
import queue
import rsa
import http.server
import socketserver
import urllib.parse

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s -
%(message)s")

class SecureFileServer:
    def __init__(self, directory="files", port=8000):
        self.directory = directory
        self.port = port
        os.makedirs(directory, exist_ok=True)

    def start_server(self):
        handler = http.server.SimpleHTTPRequestHandler
        os.chdir(self.directory)
        with socketserver.TCPServer(("0.0.0.0", self.port), handler) as httpd:
            httpd.serve_forever()

class DatabaseLogger:
    def __init__(self, db_name="logs.db"):
        self.conn = sqlite3.connect(db_name)
        self.cursor = self.conn.cursor()
        self.cursor.execute('''CREATE TABLE IF NOT EXISTS logs (timestamp TEXT,
event TEXT)''')
        self.conn.commit()

    def log_event(self, event):
        timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
        self.cursor.execute("INSERT INTO logs VALUES (?, ?)", (timestamp,
event))
        self.conn.commit()

    def get_logs(self):
        self.cursor.execute("SELECT * FROM logs")
        return self.cursor.fetchall()

class RSASecurity:
    def __init__(self):
        self.public_key, self.private_key = rsa.newkeys(2048)

    def encrypt(self, message):
        return rsa.encrypt(message.encode(), self.public_key)

    def decrypt(self, encrypted_message):
        return rsa.decrypt(encrypted_message, self.private_key).decode()

class WebInterface(http.server.SimpleHTTPRequestHandler):
```

```
    def do_GET(self):
        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(b"<html><body><h1>Secure Server</h1></body></html>")

class TwoFactorAuth:
    def __init__(self):
        self.pending_codes = {}

    def generate_code(self, user):
        code = random.randint(100000, 999999)
        self.pending_codes[user] = code
        return code

    def verify_code(self, user, code):
        return self.pending_codes.get(user) == code

if __name__ == "__main__":
    file_server = SecureFileServer()
    threading.Thread(target=file_server.start_server).start()

    db_logger = DatabaseLogger()
    db_logger.log_event("Server started")

    rsa_sec = RSASecurity()
    encrypted_msg = rsa_sec.encrypt("Test Message")
    decrypted_msg = rsa_sec.decrypt(encrypted_msg)

    two_factor = TwoFactorAuth()
    auth_code = two_factor.generate_code("admin")

    server = socketserver.TCPServer(("0.0.0.0", 8080), WebInterface)
    threading.Thread(target=server.serve_forever).start()
```

Файл RateLimiter.py

```
import socket
import threading
import time
import hashlib
import json
import os
import ssl
import logging
import sqlite3
import random
import base64
import queue
import hmac

logging.basicConfig(level=logging.INFO, format="%(asctime)s - %(levelname)s -
%(message)s")

class RateLimiter:
    def __init__(self, max_requests=5, period=60):
        self.requests = {}
        self.max_requests = max_requests
        self.period = period

    def is_allowed(self, ip):
        current_time = time.time()
        if ip not in self.requests:
            self.requests[ip] = []
        self.requests[ip] = [req for req in self.requests[ip] if req >
current_time - self.period]
        if len(self.requests[ip]) < self.max_requests:
            self.requests[ip].append(current_time)
            return True
        return False

class AccessControl:
    def __init__(self):
        self.roles = {"admin": ["read", "write", "delete"], "user": ["read"],
"guest": []}

    def check_access(self, role, action):
        return action in self.roles.get(role, [])

class DoSProtection:
    def __init__(self, threshold=10):
        self.requests = {}
        self.threshold = threshold

    def register_request(self, ip):
        current_time = time.time()
        if ip not in self.requests:
            self.requests[ip] = []
        self.requests[ip].append(current_time)
        self.requests[ip] = [req for req in self.requests[ip] if req >
current_time - 10]
        return len(self.requests[ip]) > self.threshold

class MQTTHandler:
    def __init__(self, broker="localhost", port=1883):
        self.broker = broker
        self.port = port
        self.topics = {}
```

```
    def publish(self, topic, message):
        if topic not in self.topics:
            self.topics[topic] = []
        self.topics[topic].append(message)

    def subscribe(self, topic):
        return self.topics.get(topic, [])

class SIEMIntegration:
    def __init__(self, log_file="siem_logs.txt"):
        self.log_file = log_file

    def send_log(self, event):
        with open(self.log_file, "a") as f:
            f.write(f"{time.strftime('%Y-%m-%d %H:%M:%S')} - {event}\n")

if __name__ == "__main__":
    rate_limiter = RateLimiter()
    access_control = AccessControl()
    dos_protection = DoSProtection()
    mqtt_handler = MQTTHandler()
    siem_integration = SIEMIntegration()

    user_ip = "192.168.1.1"
    if rate_limiter.is_allowed(user_ip):
        siem_integration.send_log("Access granted to " + user_ip)
    else:
        siem_integration.send_log("Rate limit exceeded for " + user_ip)

    if access_control.check_access("admin", "write"):
        siem_integration.send_log("Admin access granted")

    if dos_protection.register_request(user_ip):
        siem_integration.send_log("DoS attempt detected from " + user_ip)

    mqtt_handler.publish("alerts", "Intrusion detected")
    siem_integration.send_log("MQTT message published")
```

Файл scapy.py

```
from scapy.interfaces import resolve_iface
from scapy.utils import atol, ltoa, itom, pretty_list

from typing import (
    Any,
    Dict,
    List,
    Optional,
    Tuple,
    Union,
)


##############################
#  Routing/Interfaces stuff  #
##############################

class Route:
    def __init__(self):
        # type: () -> None
        self.routes = []  # type: List[Tuple[int, int, str, str, str, int]]
        self.invalidate_cache()
        if conf.route_autoload:
            self.resync()

    def invalidate_cache(self):
        # type: () -> None
        self.cache = {}  # type: Dict[Tuple[str, Optional[str]], Tuple[str, str,
str]]

    def resync(self):
        # type: () -> None
        from scapy.arch import read_routes
        self.invalidate_cache()
        self.routes = read_routes()

    def __repr__(self):
        # type: () -> str
        rtlst = []  # type: List[Tuple[Union[str, List[str]], ...]]
        for net, msk, gw, iface, addr, metric in self.routes:
            if_repr = resolve_iface(iface).description
            rtlst.append((ltoa(net),
                          ltoa(msk),
                          gw,
                          if_repr,
                          addr,
                          str(metric)))

        return pretty_list(rtlst,
                           [("Network", "Netmask", "Gateway", "Iface", "Output
IP", "Metric")])  # noqa: E501

    def make_route(self,
                   host=None,  # type: Optional[str]
                   net=None,  # type: Optional[str]
                   gw=None,  # type: Optional[str]
                   dev=None,  # type: Optional[str]
                   metric=1,  # type: int
                   ):
        # type: (...) -> Tuple[int, int, str, str, str, int]
        if host is not None:
```

```
            thenet, msk = host, 32
        elif net is not None:
            thenet, msk_b = net.split("/")
            msk = int(msk_b)
        else:
            raise Scapy_Exception("make_route: Incorrect parameters. You should
specify a host or a net")  # noqa: E501
        if gw is None:
            gw = "0.0.0.0"
        if dev is None:
            if gw:
                nhop = gw
            else:
                nhop = thenet
            dev, ifaddr, _ = self.route(nhop)
        else:
            ifaddr = "0.0.0.0"  # acts as a 'via' in `ip addr add`
        return (atol(thenet), itom(msk), gw, dev, ifaddr, metric)

    def add(self, *args, **kargs):
        # type: (*Any, **Any) -> None
        """Add a route to Scapy's IPv4 routing table.
        add(host|net, gw|dev)

        :param host: single IP to consider (/32)
        :param net: range to consider
        :param gw: gateway
        :param dev: force the interface to use
        :param metric: route metric

        Examples:

        - `ip route add 192.168.1.0/24 via 192.168.0.254`::
            >>> conf.route.add(net="192.168.1.0/24", gw="192.168.0.254")

        - `ip route add 192.168.1.0/24 dev eth0`::
            >>> conf.route.add(net="192.168.1.0/24", dev="eth0")

        - `ip route add 192.168.1.0/24 via 192.168.0.254 metric 1`::
            >>> conf.route.add(net="192.168.1.0/24", gw="192.168.0.254",
metric=1)
        """
        self.invalidate_cache()
        self.routes.append(self.make_route(*args, **kargs))

    def delt(self, *args, **kargs):
        # type: (*Any, **Any) -> None
        """Remove a route from Scapy's IPv4 routing table.
        delt(host|net, gw|dev)

        Same syntax as add()
        """
        self.invalidate_cache()
        route = self.make_route(*args, **kargs)
        try:
            i = self.routes.index(route)
            del self.routes[i]
        except ValueError:
            raise ValueError("No matching route found!")

    def ifchange(self, iff, addr):
        # type: (str, str) -> None
        self.invalidate_cache()
```

```
        the_addr, the_msk_b = (addr.split("/") + ["32"])[:2]
        the_msk = itom(int(the_msk_b))
        the_rawaddr = atol(the_addr)
        the_net = the_rawaddr & the_msk

        for i, route in enumerate(self.routes):
            net, msk, gw, iface, addr, metric = route
            if iff != iface:
                continue
            if gw == '0.0.0.0':
                self.routes[i] = (the_net, the_msk, gw, iface, the_addr, metric)
# noqa: E501
            else:
                self.routes[i] = (net, msk, gw, iface, the_addr, metric)
        conf.netcache.flush()

    def ifdel(self, iff):
        # type: (str) -> None
        self.invalidate_cache()
        new_routes = []
        for rt in self.routes:
            if iff == rt[3]:
                continue
            new_routes.append(rt)
        self.routes = new_routes

    def ifadd(self, iff, addr):
        # type: (str, str) -> None
        self.invalidate_cache()
        the_addr, the_msk_b = (addr.split("/") + ["32"])[:2]
        the_msk = itom(int(the_msk_b))
        the_rawaddr = atol(the_addr)
        the_net = the_rawaddr & the_msk
        self.routes.append((the_net, the_msk, '0.0.0.0', iff, the_addr, 1))

    def route(self, dst=None, dev=None, verbose=conf.verb, _internal=False):
        # type: (Optional[str], Optional[str], int, bool) -> Tuple[str, str,
str]
        """Returns the IPv4 routes to a host.

        :param dst: the IPv4 of the destination host
        :param dev: (optional) filtering is performed to limit search to route
                    associated to that interface.

        :returns: tuple (iface, output_ip, gateway_ip) where
            - ``iface``: the interface used to connect to the host
            - ``output_ip``: the outgoing IP that will be used
            - ``gateway_ip``: the gateway IP that will be used
        """
        dst = dst or "0.0.0.0"  # Enable route(None) to return default route
        if isinstance(dst, bytes):
            try:
                dst = plain_str(dst)
            except UnicodeDecodeError:
                raise TypeError("Unknown IP address input (bytes)")
        if (dst, dev) in self.cache:
            return self.cache[(dst, dev)]
        # Transform "192.168.*.1-5" to one IP of the set
        _dst = dst.split("/")[0].replace("*", "0")
        while True:
            idx = _dst.find("-")
            if idx < 0:
                break
```

```
            m = (_dst[idx:] + ".").find(".")
            _dst = _dst[:idx] + _dst[idx + m:]

        atol_dst = atol(_dst)
        paths = []
        for d, m, gw, i, a, me in self.routes:
            if not a:  # some interfaces may not currently be connected
                continue
            if dev is not None and i != dev:
                continue
            aa = atol(a)
            if aa == atol_dst:
                paths.append(
                    (0xffffffff, 1, (conf.loopback_name, a, "0.0.0.0"))  # noqa:
E501
                )
            if (atol_dst & m) == (d & m):
                paths.append((m, me, (i, a, gw)))

        if not paths:
            if verbose:
                warning("No route found for IPv4 destination %s "
                        "(no default route?)", dst)
            return (dev or conf.loopback_name, "0.0.0.0", "0.0.0.0")
        # Choose the more specific route
        # Sort by greatest netmask and use metrics as a tie-breaker
        paths.sort(key=lambda x: (-x[0], x[1]))
        # Return interface
        ret = paths[0][2]
        # Check if source is 0.0.0.0. This is a 'via' route with no src.
        if ret[1] == "0.0.0.0" and not _internal:
            # Then get the source from route(gw)
            ret = (ret[0], self.route(ret[2], _internal=True)[1], ret[2])
        self.cache[(dst, dev)] = ret
        return ret

    def get_if_bcast(self, iff):
        # type: (str) -> List[str]
        bcast_list = []
        for net, msk, gw, iface, addr, metric in self.routes:
            if net == 0:
                continue    # Ignore default route "0.0.0.0"
            elif msk == 0xffffffff:
                continue    # Ignore host-specific routes
            if iff != iface:
                continue
            bcast = net | (~msk & 0xffffffff)
            bcast_list.append(ltoa(bcast))
        if not bcast_list:
            warning("No broadcast address found for iface %s\n", iff)
        return bcast_list


conf.route = Route()

# Update conf.iface
conf.ifaces.load_confiface()
```

Файл sessions.py

```
from collections import defaultdict
import socket
import struct

from scapy.compat import orb
from scapy.config import conf
from scapy.packet import Packet
from scapy.pton_ntop import inet_pton

# Typing imports
from typing import (
    Any,
    Callable,
    DefaultDict,
    Dict,
    Iterator,
    List,
    Optional,
    Tuple,
    Type,
    cast,
    TYPE_CHECKING,
)
from scapy.compat import Self
if TYPE_CHECKING:
    from scapy.supersocket import SuperSocket


class DefaultSession(object):
    """Default session: no stream decoding"""

    def __init__(self, supersession: Optional[Self] = None):
        if supersession and not isinstance(supersession, DefaultSession):
            supersession = supersession()
        self.supersession = supersession

    def process(self, pkt: Packet) -> Optional[Packet]:
        """
        Called to pre-process the packet
        """
        # Optionally handle supersession
        if self.supersession:
            return self.supersession.process(pkt)
        return pkt

    def recv(self, sock: 'SuperSocket') -> Iterator[Packet]:
        """
        Will be called by sniff() to ask for a packet
        """
        pkt = sock.recv()
        if not pkt:
            return
        pkt = self.process(pkt)
        if pkt:
            yield pkt


class IPSession(DefaultSession):
    """Defragment IP packets 'on-the-flow'.

    Usage:
```

```
    >>> sniff(session=IPSession)
    """

    def __init__(self, *args, **kwargs):
        # type: (*Any, **Any) -> None
        DefaultSession.__init__(self, *args, **kwargs)
        self.fragments = defaultdict(list)  # type: DefaultDict[Tuple[Any, ...],
List[Packet]]  # noqa: E501

    def process(self, packet: Packet) -> Optional[Packet]:
        from scapy.layers.inet import IP, _defrag_ip_pkt
        if not packet:
            return None
        if IP not in packet:
            return packet
        return _defrag_ip_pkt(packet, self.fragments)[1]  # type: ignore


class StringBuffer(object):
    """StringBuffer is an object used to re-order data received during
    a TCP transmission.

    Each TCP fragment contains a sequence number, which marks
    (relatively to the first sequence number) the index of the data contained
    in the fragment.

    If a TCP fragment is missed, this class will fill the missing space with
    zeros.
    """

    def __init__(self):
        # type: () -> None
        self.content = bytearray(b"")
        self.content_len = 0
        self.noff = 0  # negative offset
        self.incomplete = []  # type: List[Tuple[int, int]]

    def append(self, data: bytes, seq: Optional[int] = None) -> None:
        if not data:
            return
        data_len = len(data)
        if seq is None:
            seq = self.content_len
        seq = seq - 1 - self.noff
        if seq < 0:
            # Data is located before the start of the current buffer
            # (e.g. the first fragment was missing)
            self.content = bytearray(b"\x00" * (-seq)) + self.content
            self.content_len += (-seq)
            self.noff += seq
            seq = 0
        if seq + data_len > self.content_len:
            # Data is located after the end of the current buffer
            self.content += b"\x00" * (seq - self.content_len + data_len)
            # As data was missing, mark it.
            # self.incomplete.append((self.content_len, seq))
            self.content_len = seq + data_len
            assert len(self.content) == self.content_len
        # XXX removes empty space marker.
        # for ifrag in self.incomplete:
        #     if [???]:
        #         self.incomplete.remove([???])
        memoryview(self.content)[seq:seq + data_len] = data
```

```
    def shiftleft(self, i: int) -> None:
        self.content = self.content[i:]
        self.content_len -= i

    def full(self):
        # type: () -> bool
        # Should only be true when all missing data was filled up,
        # (or there never was missing data)
        return bool(self)

    def clear(self):
        # type: () -> None
        self.__init__()  # type: ignore

    def __bool__(self):
        # type: () -> bool
        return bool(self.content_len)
    __nonzero__ = __bool__

    def __len__(self):
        # type: () -> int
        return self.content_len

    def __bytes__(self):
        # type: () -> bytes
        return bytes(self.content)

    def __str__(self):
        # type: () -> str
        return cast(str, self.__bytes__())


def streamcls(cls: Type[Packet]) -> Callable[
    [bytes, Dict[str, Any], Dict[str, Any]],
    Optional[Packet],
]:
    """
    Wraps a class for use when dissecting streams.
    """
    if hasattr(cls, "tcp_reassemble"):
        return cls.tcp_reassemble  # type: ignore
    else:
        # There is no tcp_reassemble. Just dissect the packet
        return lambda data, *_: data and cls(data)


class TCPSession(IPSession):
    """A Session that reconstructs TCP streams"""

    def __init__(self, app=False, *args, **kwargs):
        # type: (bool, *Any, **Any) -> None
        super(TCPSession, self).__init__(*args, **kwargs)
        self.app = app
        if app:
            self.data = StringBuffer()
            self.metadata = {}  # type: Dict[str, Any]
            self.session = {}  # type: Dict[str, Any]
        else:
            # The StringBuffer() is used to build a global
            # string from fragments and their seq nulber
            self.tcp_frags = defaultdict(
                lambda: (StringBuffer(), {})
```

```
            )  # type: DefaultDict[bytes, Tuple[StringBuffer, Dict[str, Any]]]
            self.tcp_sessions = defaultdict(
                dict
            )  # type: DefaultDict[bytes, Dict[str, Any]]
        # Setup stopping dissection condition
        from scapy.layers.inet import TCP
        self.stop_dissection_after = TCP

    def _get_ident(self, pkt, session=False):
        # type: (Packet, bool) -> bytes
        underlayer = pkt["TCP"].underlayer
        af = socket.AF_INET6 if "IPv6" in pkt else socket.AF_INET
        src = underlayer and inet_pton(af, underlayer.src) or b""
        dst = underlayer and inet_pton(af, underlayer.dst) or b""
        if session:
            # Bidirectional
            def xor(x, y):
                # type: (bytes, bytes) -> bytes
                return bytes(orb(a) ^ orb(b) for a, b in zip(x, y))
            return struct.pack("!4sH", xor(src, dst), pkt.dport ^ pkt.sport)
        else:
            # Uni-directional
            return src + dst + struct.pack("!HH", pkt.dport, pkt.sport)

    def _strip_padding(self, pkt: Packet) -> Optional[bytes]:
        """Strip the packet of any padding, and return the padding.
        """
        if isinstance(pkt, conf.padding_layer):
            return cast(bytes, pkt.load)
        pad = pkt.getlayer(conf.padding_layer)
        if pad is not None and pad.underlayer is not None:
            # strip padding
            del pad.underlayer.payload
            return cast(bytes, pad.load)
        return None

    def process(self,
                pkt: Packet,
                cls: Optional[Type[Packet]] = None) -> Optional[Packet]:
        """Process each packet: matches the TCP seq/ack numbers
        to follow the TCP streams, and orders the fragments.
        """
        packet = None  # type: Optional[Packet]
        if self.app:
            # Special mode: Application layer. Use on top of TCP
            self.data.append(bytes(pkt))
            if cls is None and not isinstance(pkt, bytes):
                cls = pkt.__class__
            if "tcp_reassemble" in self.metadata:
                tcp_reassemble = self.metadata["tcp_reassemble"]
            elif cls is not None:
                self.metadata["tcp_reassemble"] = tcp_reassemble =
streamcls(cls)
            else:
                return None
            if self.data.full():
                packet = tcp_reassemble(
                    bytes(self.data),
                    self.metadata,
                    self.session,
                )
            if packet:
                padding = self._strip_padding(packet)
```

```
                if padding:
                    # There is remaining data for the next payload.
                    self.data.shiftleft(len(self.data) - len(padding))
                    # Skip full-padding
                    if isinstance(packet, conf.padding_layer):
                        return None
                else:
                    # No padding (data) left. Clear
                    self.data.clear()
                self.metadata.clear()
                return packet
            return None

        _pkt = super(TCPSession, self).process(pkt)
        if _pkt is None:
            return None
        else:  # Python 3.8 := would be nice
            pkt = _pkt

        from scapy.layers.inet import IP, TCP
        if not pkt:
            return None
        if TCP not in pkt:
            return pkt
        pay = pkt[TCP].payload
        new_data = pay.original
        # Match packets by a unique TCP identifier
        ident = self._get_ident(pkt)
        data, metadata = self.tcp_frags[ident]
        tcp_session = self.tcp_sessions[self._get_ident(pkt, True)]
        # Handle TCP sequence numbers
        seq = pkt[TCP].seq
        if "seq" not in metadata:
            metadata["seq"] = seq
        if "next_seq" in metadata and seq < metadata["next_seq"]:
            # Retransmitted data (that we already returned)
            new_data = new_data[metadata["next_seq"] - seq:]
            if not new_data:
                return None
            seq = metadata["next_seq"]
        # Let's guess which class is going to be used
        if "pay_class" not in metadata:
            metadata["pay_class"] = pay_class =
pkt[TCP].guess_payload_class(new_data)
            metadata["tcp_reassemble"] = tcp_reassemble = streamcls(pay_class)
        else:
            tcp_reassemble = metadata["tcp_reassemble"]

        if pay:
            # Get a relative sequence number for a storage purpose
            relative_seq = metadata.get("relative_seq", None)
            if relative_seq is None:
                relative_seq = metadata["relative_seq"] = seq - 1
            seq = seq - relative_seq
            # Add the data to the buffer
            data.append(new_data, seq)

        # Check TCP FIN or TCP RESET
        if pkt[TCP].flags.F or pkt[TCP].flags.R:
            metadata["tcp_end"] = True
        elif not pay:
            # If there's no payload and the stream isn't ending, ignore.
            return pkt
```

```
        # In case any app layer protocol requires it,
        # allow the parser to inspect TCP PSH flag
        if pkt[TCP].flags.P:
            metadata["tcp_psh"] = True
        # XXX TODO: check that no empty space is missing in the buffer.
        # XXX Currently, if a TCP fragment was missing, we won't notice it.
        if data.full():
            # Reassemble using all previous packets
            metadata["original"] = pkt
            metadata["ident"] = ident
            packet = tcp_reassemble(
                bytes(data),
                metadata,
                tcp_session
            )
        # Stack the result on top of the previous frames
        if packet:
            if "seq" in metadata:
                pkt[TCP].seq = metadata["seq"]
            # Clear TCP reassembly metadata
            metadata.clear()
            # Check for padding
            padding = self._strip_padding(packet)
            while padding:
                # There is remaining data for the next payload.
                full_length = data.content_len - len(padding)
                metadata["relative_seq"] = relative_seq + full_length
                data.shiftleft(full_length)
                # There might be a sub-payload hidden in the padding
                sub_packet = tcp_reassemble(
                    bytes(data),
                    metadata,
                    tcp_session
                )
                if sub_packet:
                    packet /= sub_packet
                    padding = self._strip_padding(sub_packet)
                else:
                    break
            else:
                # No padding (data) left. Clear
                data.clear()
                del self.tcp_frags[ident]
            # Minimum next seq
            metadata["next_seq"] = pkt[TCP].seq + len(new_data)
            # Skip full-padding
            if isinstance(packet, conf.padding_layer):
                return None
            # Rebuild resulting packet
            if pay:
                pay.underlayer.remove_payload()
            if IP in pkt:
                pkt[IP].len = None
                pkt[IP].chksum = None
            pkt = pkt / packet
            pkt.wirelen = None
            return pkt
        return None

    def recv(self, sock: 'SuperSocket') -> Iterator[Packet]:
        """
        Will be called by sniff() to ask for a packet
```

```
"""
pkt = sock.recv(stop_dissection_after=self.stop_dissection_after)
# Now handle TCP reassembly
if self.app:
    while pkt is not None:
        pkt = self.process(pkt)
        if pkt:
            yield pkt
            # keep calling process as there might be more
            pkt = b""  # type: ignore
else:
    pkt = self.process(pkt)  # type: ignore
    if pkt:
        yield pkt
return None
```