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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації”

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

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 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

Бідненку Павлу Максимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*
(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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації*

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
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2025 р.

Підпис здобувача
_____ Бідненко П.М.
(прізвище та ініціали)

АНОТАЦІЯ

Бідненко П.М. Програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації. 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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

Метою розробки є програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

Результат роботи – програмна реалізація системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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, оперативність передачі даних

ABSTRACT

Bidnenko P.M. Software for a system for increasing data transmission efficiency by reducing graphic information traffic. 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 a system for increasing data transmission efficiency by reducing graphic information traffic.

The purpose of the development is software for a system for increasing data transmission efficiency by reducing graphic information traffic.

The result of the work is a software implementation of a system for increasing data transmission efficiency by reducing graphic information traffic.

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, data transfer efficiency

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ГА	–	генетичний алгоритм
ДПФ	–	дискретне перетворення Фур'є
УБК	–	метод усіченого блокового кодування

ВСТУП

Актуальність теми. На сучасному рівні розвитку телекомунікаційних мереж і систем виникає ряд проблем, зв'язаних безпосередньо із забезпеченням необхідного рівня якості обслуговування за критеріями QoS у відповідності стандартам Міжнародного союзу електрозв'язку, які приведуть до необхідності й актуальності проведення досліджень і створення перспективних методів і алгоритмів обробки й передачі інформації [2, 4]. Забезпечення якості обслуговування зв'язку за критеріями QoS має на увазі використання кодування, стиск, шифрування, кращий розподіл навантаження каналів передачі, розподіл навантаження на обчислювальні ресурси, технологій цифрової обробки сигналів, і інші можливості керування й оптимізації. Одним із пріоритетних напрямків є поліпшення використання існуючих апаратних рішень, що надає підвищення якості обслуговування зв'язку за критеріями QoS або збільшення пропускної здатності каналу зв'язку. Високий рівень розвитку телекомунікаційних технологій і прагнення сучасного соціуму до інформаційного об'єднання, створює умови створення й розвитку глобальної телекомунікаційної структури, що забезпечувала б вільний доступ до інформаційних ресурсів для кожного бажаючого. На жаль, відомі технологічні й протокольні рішення в цій області, в умовах підвищеної інтенсивності інформаційного трафіку, не забезпечують заданих вимог. Тому безліч передових концепцій і технологій (Traffic Engineering (TE), DiffServ, FastReRouting і ін.) не можуть повною мірою реалізувати функції, які пов'язані з підвищенням якості обслуговування й масштабованості в телекомунікаційних системах (ТКС).

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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

– Огляд існуючих систем підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

– Дослідження системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

– Програмна реалізація системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації, 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

Сучасний прогрес у галузі телекомунікаційних технологій привів до розробки безлічі методів, призначених для забезпечення якості обслуговування передачі інформації. Однак на додаток до мультисервісних ТКС ці методи часто не мають достатньої теоретичної основи, тому опис властивостей, переваг і недоліків обґрунтовується переважно на практичному досвіді використання, що не гарантує їхньої успішності в умовах широкого спектра телекомунікаційних послуг. Внаслідок цього стає актуальним диференціальний підхід до оптимізації використання апаратних ресурсів у телекомунікаційних системах і мережах. Частковим випадком ТКС є корпоративні мережі, де спостерігаються значні варіації вимог ефективності передачі інформації й окремо виділяються системи зі значним обсягом часто затребуваних даних.

1.2 Область застосування

У теорії інформації й передачі даних накопичені значний теоретичний матеріал і практичний досвід. Однак, динамічний розвиток телекомунікаційних пристроїв і засобів керування, і також розмаїтість програмних технологічних рішень сприяють тому, що постановка завдань підвищення якості обслуговування передачі даних істотно видозмінюється через необхідність обліку нових телекомунікаційних послуг.

Внаслідок позначеного, одержують підвищену актуальність питання розробки методів підвищення ефективності передачі інформації. При цьому ключовою особливістю таких методів і засобів є адаптація засобів керування й протоколів до вимог і особливостей поводження окремих видів

телекомунікаційних послуг. Зокрема, динамічне керування обсягом графічного й відеоконтенту в процесі його доставки кінцевим одержувачам. Облік цих факторів і особливостей телекомунікаційного трафіку виходить за рамки існуючих методів підвищення ефективності передачі інформації й вимагає як їхньої модифікації, так і повторного розгляду.

Проблема підвищення пропускної здатності існуючих каналів зв'язку без змін апаратного забезпечення виникла при бурхливому росту кількості клієнтів глобальної мережі, при якому обновити апаратну частину стало важко не тільки у фінансовому ракурсі, але й фізично, через недостачу людино-годин. Основними шляхами зменшення завантаженості каналів зв'язку, у такому випадку, стають більше затребувані методи стиску інформації й більше затребувані методи маршрутизації. У наш час основним потоком інформації в телекомунікаційних мережах і системах залишається графічна інформація. Тому навіть незначне поліпшення ступеня стиску графічної інформації може значно збільшити якість обслуговування.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації, 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

Програма Adobe Photoshop

Дана програма є однією із затребуваних серед конвертерів. У цю програму убудовані практично всі необхідні інструменти для пакетного перетворення зображень. На даний момент актуальної є версія – CS6, поки ж розглядаємо версію CS5. Обробка файлів виробляється за допомогою екшнів, операції Batch або скрипта Image Processor.

Перший спосіб обробки зображень

Для обробки графічних зображень необхідно створити набір (Set). Для цього краще взяти тестовий зразок. Надалі цей набір буде застосований до всіх обраних файлів. Набір створюється через палітру Actions, шляхом записування необхідних дій. На даному етапі доступні тільки засоби Photoshop. Надалі список можна відкоригувати, додаючи або видаляючи певні дії. Через меню «File – Automate – Batch…»заходимо в групу налаштувань «Play», вибираємо потрібний екшн, указуємо джерело й папку для збереження оброблених файлів. Ця дія приводить до масового застосування набору.

Другий метод обробки зображень

Другий спосіб обробки є більше прийнятним, тому що не вимагає створення екшна. Метод ґрунтується на використанні скрипта Image Processor. Зміна формату або розміру зображення відбувається через меню «File – Scripts – Image Processor…»... Інші налаштування можна зробити першим способом.

Зберігаються оброблені зображення у форматах JPEG, PSD і TIFF.

Програма Adobe Lightroom

Робота програми Adobe Lightroom заснована на масовому перетворенні експортованих зображень або за допомогою модуля «Library». Для роботи із зображеннями використовуються наступні налаштування, розмічені в меню «File – Export…»:

– ExportTo-вибір експорту зображення. Як варіанти: жорсткий диск, e-mail або запис CD або DVD. Також модуль Library дозволяє експортувати зображення в Інтернет: Facebook, Flickr, Adobe Revel і SmugMug;

– Export Location – папка для збереження зображень;

– File naming – вибір ім'я файлів по масці. У програмі є великий перелік змінних. Так само змінні можуть бути взяті з метаданих зображення;

– File Settings – тут вибирається формат збереження зображень. Можна залишити вихідний формат або вибрати один з наступних: JPEG, PSD, TIFF, DNG

– Image Sizing – вказуються розміри й дозвіл збереженого зображення;

– Output Sharpening – визначення різкості зображення;

– Metadata – запит по збереженню метаданих файлу;

– Watermarking – опція накладення водяного знака.

Якщо вас не влаштовують стандартні передустановки, надані в лівій бічній панелі («Preset»), ви маєте можливість додати власні.

Програма Image Tuner

Найбільш проста, не ускладнена налаштуваннями й фільтрами, програма для базової обробки зображень. Перелік її можливостей обмежується зміною відтінків, розмірів, орієнтації зображення, додаванням водяного знака. Програма працює в режимі «одного вікна»: у ліву половину завантажуються файли, що піддаються обробці. У правій вказуються параметри конвертації.

Переваги цієї програми:

– простота використання;

– підтримує такі популярні формати, як JPEG, BMP, PNG, GIF, TIFF, RAW, NEF і інші;

– є функція передперегляду. Спрацьовує при кліку на зображення.

Недоліки:

– надмірна простота фільтрів. Фільтри настільки прості, що не мають налаштувань. Перебувають у розділі меню «Змінити розмір»;

– невелика кількість форматів для збереження готових ескізів: усього 5;

– недоробки в оформленні програми: частина налаштувань на англійському, що залишилася частина переведена на російську.

Таким чином, ця програма призначена для елементарної базової обробки зображень.

Програма IrfanView

Основними функціями цієї програми є перегляд зображень.

Переваги:

– доступність. Програма безкоштовна для завантаження;

– функціональність. Виконує функції перегляду й конвертації. Як конвертор працює через меню «Batch Conversion/Rename…»... Підтримує три варіанти режиму: пакетне перейменування, перетворення й змішаний режим;

– компактність (невеликий розмір);

– значна кількість форматів (близько 20);

– тестовий режим, доступний для пакетного перейменування файлів.

Недоліки:

– не всі параметри доступні для повного списку форматів;

– деякі перетворення відбуваються тільки при активації Use advanced options…», по натисканню кнопки «Advanced». В іншому випадку доступні лише стандартні для переглядача перетворення: зміна розмірів, кадрування, горизонтальне/вертикальне відбиття, водяний знак.

– передперегляд передбачається тільки для вихідного зображення. Якщо, наприклад, розміри ми можемо вказати попіксельно, то зміна колірних параметрів, яскравості, балансу являє собою проблему, тому що змінюються вони

методом уведення цифр. А результати перетворень ми зможемо побачити тільки після закінчення конвертацій і виходу з «Advanced».

Таким чином, якщо вас улаштовують базові перетворення, що не вимагають передперегляду, дана програма задовольнить ваші вимоги. Але внести її в список найбільш зручних конвертерів не можна.

Програма AVS Image Converter

AVS Image Converter є програмою-конвертером. Вона є однією з набору програм AVS4You, призначених для роботи із зображеннями. Особливістю цієї програми є допоміжні утиліти Software Navigator і Update Manager. До речі, це пояснює великий розмір програми: 27 Мб.

Переваги:

– простота у використанні. Дуже простий інтерфейс, зі спрощеними функціями або передустановками, що дозволяє з легкістю працювати із програмою навіть недосвідченим користувачам;

– кількість підтримуваних форматів: 8 для запису, більше 20 для читання;

– зв'язок з Інтернетом: можна обробляти фото, імпортоване з аккаунтів Flickr або Facebook, а в самій програмі доступні передналаштування для фотознімків, призначених для публікації в Інтернеті;

– окремо розташована вкладка «Водяной знак». Застосовується для накладки тексту або зображення на фото.

Недоліки:

– мало налаштувань перетворення. В основному переважають налаштування, спрямовані на колір: яскравість, контрастність, колірна температура. Є присутнім ефект розмиття/різкості, додавання тексту. З усім переліком ефектів можна познайомитися у вкладці «Корекція».

Програма FastStone Photo Resizer

По функціональності ця програма нагадує IrfanView, але, на відміну від останньої, надає користувачеві більше зручний і розширений вибір параметрів.

Переваги:

– керування зображенням. Указавши формат вихідного зображення («Output Format») і папку збереження («Output Folder») можна заощадити час на виконанні цих функцій для кожного зображення окремо;

– функція «Search and Replace». З її допомогою назва файлів коректується без повної зміни імені.

Недоліки:

– базовий набір перетворень. Перелік доступних перетворень: коректування розміру, текстури, перспективи, додавання водяного зображення, добірка рамок;

– відсутність режиму (вікна) передперегляду;

– нераціональне застосування області інтерфейсу. Більша частина робочого простору являє собою область для вибору файлів.

Програма XnConvert

Програма XnConvert є одним з компонентів відомого багатьом користувачам переглядача зображень XnView. Створені на одній базі, програми виконують практично ідентичні дії, але все-таки мають ряд відмінностей, які будуть розглядатися нижче.

XnConvert є програмою-конвертером, призначеною тільки для перетворення зображень, і не містить у собі функцію перегляду. З однієї сторони це є плюсом програми, тому що вона виконує строго свої функції. А з іншої сторони для роботи необхідно задіяти іншу програму-переглядач.

Основні принципи роботи програми максимально прості. Зображення, що піддаються обробці, додаються методом перетаскування або за допомогою кнопок. Далі зі списку вибираються варіанти перетворення. Списки перебувають зверху й відбиваються у вигляді ескізів, що являє собою деякі незручності для користувачів. Справа в тому, що такого роду сортування не дозволяє переглянути інформацію про вихідні файли, тому що це могло бути при сортуванні у вигляді таблиці. Тому сортування є більше умовним, ніж функціональним.

Тепер більш докладно про функції перетворення. Список перетворень перебуває на основній вкладці «Дії» і ділиться на 4 групи:

– зображення: робота, спрямована на трансформацію файлу, або зв'язана із властивостями файлу;

– корекція: обробка колірної гами й рівнів;

– фільтри: размивка зображення, налаштування різкості, зміна фокуса;

– різне: інші функції.

Вкладка «Вихідні дані». У цій вкладці відбувається робота з файлами, що пройшли перетворення: вказуються параметри для збереження (ім'я, шлях збереження, формат). Є додаткові параметри для збереження, вони застосовуються для таких форматів, як GIF, PNG, JPG та ін.

Кнопка «Завантажити сценарій». Використовується для експорту зображень, призначених для роботи з Інтернетом. Упакування або відправлення по FTP або на e-mail, завантаження оброблених на Picasa– або Flickr-аккаунт зображень – це функції перебувають вище зазначеної кнопки.

І в завершення потрібно відзначити, що програма підтримує більше 500 форматів для читання. Правда, деякі з них більше специфічні, вимагають додаткових ресурсів. У цьому випадку необхідна установка GhostScript або плагіна CAD.

Програма XnView

Спрощений варіант XnConver. Простота роботи полягає в тому, що в цій програмі передбачені тільки дві вкладки. Перша – для формату й параметрів збереження. Друга містить у собі список перетворень. Для вибору й застосування перетворення необхідно додати в список половину вікна, що перебуває праворуч. Є функція «Збереження сценарію». Вікно налаштувань перебуває в меню «Інструменти – Пакетна обробка…»...

Програма «Фотоконвертер»

Програма створена в декількох редакціях: домашня (мінімальна), стандартна, професійна. Професійна редакція дозволяє використовувати функції

Adobe Photoshop, підтримує більше 400 графічних форматів, функціонує з командного рядка. Стандартна версія відрізняється простотою й зручністю покрокових налаштувань, підтримує тільки самі затребувані формати зображень (JPEG, TIFF, GIF, PNG, BMP). Інтерфейс програми розділений на дві частини, тому перший крок перетворення – це додавання зображення в праву частину вікна. Другий крок – це безпосередньо саме перетворення, що відбувається через кнопку «додати дія». У мінімальній версії, крім стандартних функцій, передбачений ефект розмивання/різкості, усунення червоних очей. Стандартна версія допускає додавання водяного знака. І третій, завершальний, крок – збереження. На цьому етапі вибирається формат збереження, додатково є опція «Перейменування файлу по масці». Завершенням і збереженням є натискання кнопки «Старт».

Хотілося б відзначити в цій програмі більше логічне розміщення функцій перетворення, ніж у розглянутій вище. Але, все таки, є певні недоробки: наприклад, функції корекції рівнів і кадрування ставляться до редагування, а в програмі вони перебувають у групі налаштувань «Автоматичні». Однак згодом при частому використанні програми це не викликає яких-небудь складностей.

Один з мінусів програми – довідка англійською мовою, але подивитися відповіді на питання, що вас цікавлять, можна на сайті.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Як мова програмування обрана Python. Python – високорівнева мова програмування загального призначення з акцентом на продуктивність розроблювача й читаність коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Python підтримує кілька парадигм програмування, у тому числі структурне, об'єктно-орієнтоване, функціональне, імперативне й аспектно-

орієнтоване. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточні обчислень і зручні високорівневі структури даних. Код у Python організовується у функції й класи, які можуть поєднуватися в модулі (які у свою чергу можуть бути об'єднані в пакети).

Еталонною реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ. Він поширюється вільно під дуже ліберальною ліцензією, що дозволяє використовувати його без обмежень у будь-яких застосунках, включаючи пропрієтарні. Є реалізації інтерпретаторів для JVM (з можливістю компіляції), MSIL (з можливістю компіляції), LLVM і інших. Проект PyPy пропонує реалізацію Python на самому Python, що зменшує витрати на зміни мови й постановку експериментів над новими можливостями.

Python – мова програмування, що активно розвивається, нові версії (з додаванням/зміною язикових властивостей) виходять приблизно раз у два з половиною року. Внаслідок цього й деяких інших причин на Python відсутні ANSI, ISO або інші офіційні стандарти, їхня роль виконує CPython.

Python портований і працює майже на всіх відомих платформах – від КПК до мейнфреймів. Існують порти під Microsoft Windows, практично всі варіанти UNIX (включаючи FreeBSD і Linux), Plan 9, Mac OS і Mac OS X, iPhone OS 2.0 і вище, Palm OS, OS/2, Amiga, AS/400 і навіть OS/390, Symbian і Android.

При цьому, на відміну від багатьох портуємих систем, для всіх основних платформ Python має підтримку характерних для даної платформи технологій (наприклад, Microsoft COM/DCOM). Більше того, існує спеціальна версія Python для віртуальної машини Java – Jython, що дозволяє інтерпретаторові виконуватися на будь-якій системі, що підтримує Java, при цьому класи Java можуть безпосередньо використовуватися з Python й навіть бути написаними на Python. Також кілька проектів забезпечують інтеграцію із платформою Microsoft .NET, основні з яких – IronPython і Python.Net.

Python підтримує динамічну типізацію, тобто тип змінної визначається тільки під час виконання. Тому замість «присвоювання значення змінної» краще говорити про «зв'язування значення з деяким ім'ям». У Python є убудовані типи: бульові, рядки, Unicode-рядки, цілі числа довільної точності, числа із плаваючої коми, комплексні числа й деякі інші. З колекцій Python підтримує кортежі (*tuples*), списки, словники (асоціативні масиви) і, починаючи з версії 2.4, безлічі. Всі значення в Python є об'єктами, у тому числі функції, методи, модулі, класи.

Додати новий тип можна або написавши клас (class), або визначивши новий тип у модулі розширення (наприклад, написаному мовою C). Система класів підтримує спадкування (одиночне й множинне) і метапрограмування. Можливе спадкування від більшості убудованих типів і типів розширень.

Всі об'єкти діляться на посилальні й атомарні. До атомарного ставляться int, long, complex і деякі інші. При присвоюванні атомарних об'єктів копіюється їхнє значення, у той час як для посилальних копіюється тільки покажчик на об'єкт, таким чином, обидві змінні після присвоювання використовують те саме значення. Посилальні об'єкти бувають змінювані й незмінні. Наприклад, рядки й кортежі є незмінними, а списки, словники й багато інших об'єктів – змінюваними. Кортеж у Python є, по суті, незмінним списком. У багатьох випадках кортежі працюють швидше списків, тому якщо ви не плануєте змінювати послідовність, то краще використовувати саме їх.

Мова має чіткий і послідовний синтаксис, продуману модульність й масштабованість, завдяки чому вихідний код написаних на Python програм легко читаємий.

Python – стабільна й розповсюджена мова. Він використовується в багатьох проектах і в різних якостях: як основна мова програмування або для створення розширень і інтеграції застосунків. На Python реалізоване велика кількість проектів, також він активно використовується для створення прототипів майбутніх програм. Python використовується в багатьох великих компаніях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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

Розробка критеріїв оперативності передачі даних у корпоративних мережах із графічним трафіком

У [1] наведене дослідження пошуку середнього часу доставки одного пакета в комп'ютерній мережі. Як результат наведені формули знаходження середнього математичного часу доставки пакета, а також дисперсії часу доставки цього пакета:

$$D_{\frac{T_{ñð}}{T_Ï}} = \frac{\left(T_{ÒÀ} + T_Ï\right)^2 \cdot \left(1 - P_{êâ}\left(P_{ïð} + P_{íî}\right)\right)}{T_Ï^2 \cdot \left(P_{êâ}\left(P_{ïð} + P_{íî}\right)\right)^2}, \quad (3.1)$$

де $\tau = \frac{T_{ÒÀ} + T_Ï}{T_Ï}$ – наведений час на одну спробу передачі інформації, а $p = P_{êâ}\left(P_{ïð} + P_{íî}\right)$ – наведена ймовірність удалої доставки пакета інформації. Однак для оцінки статистичних величин необхідно мати значення багатьох коефіцієнтів непрямого виміру. Тому пропонується інший підхід для одержання ймовірності кількості спроб передачі пакетів доставки багатопакетного повідомлення. На основі практично обмірюваного часу доставки повідомлення пропонується експериментально встановити статистичні характеристики мережі, а також довести пропорційну залежність часу доставки повідомлення від його розміру.

Побудова математичної моделі залежності часу передачі повідомлення від кількості пакетів переданої інформації

Припустимо що в мережі, що складається умовно з одного сегмента, передається повідомлення. Процес передачі складається з установки каналу зв'язку за час $\tau_{до}$ й передачі самих пакетів з даними. Імовірність успішного

прийняття пакета *p*, величина якої залежить від якості й виду каналу зв'язку. Імовірність втрати пакета дорівнює *q*= 1-p, після втрати виробляється повторна передача пакета. Існують ситуації, коли пакет може передаватися кілька разів. У випадку передачі безлічі пакетів, що передає сторона посилає наступний пакет, не чекаючи одержання підтвердження про правильний прийом попереднього пакета. Тому, при невдалій передачі пакета, загальний час збільшується тільки на час повторної посилки одного пакета. Цей механізм відображений у формулах (3.2) як послідовність незалежних випробувань до першої вдалої спроби:

$$p_1(i) = pq^{i-1},\ \tau(i) = i\tau_p,\ \text{для } i \geq 1. \qquad (3.2)$$

Отут $p_1(i)$ – імовірність доставки одного пакета з і-*тої* спроби; $\tau(i)$ – час зайнятості каналу передачі при доставці пакета за *i* спроб. Знаючи значення часу і їхньої ймовірності (3.2), можна оцінити загальний середньостатистичний час [1, 2] доставки повідомлення (3.3):

$$\tau(i) = \tau_k + N\sum_{i=1}^{i_{\max}} pq^{i-1}\tau_p i, \qquad (3.3)$$

де N – кількість пакетів у повідомленні, що пропорційно кількості інформації в цьому повідомленні. Внаслідок того, що τ_k, $\sum_{i=1}^{i_{\max}} pq^{i-1}\tau_p i$ постійні для конкретного шляху в телекомунікаційній мережі, то час доставки інформації пропорційно кількості інформації в повідомленні, плюс постійна встановлення каналу зв'язку й маршрутизації (3.4):

$$\tau(S) = SA + B, \qquad (3.4)$$

де *S* – кількість байтів у повідомленні; *A*, *B* – шукані постійні характеристики телекомунікаційного з'єднання. Проведемо перевірку й доповнення гіпотези (3.4).

Розподіл імовірності доставки повідомлення з *N* пакетів

Використовуючи формули (3.2) можна побудувати розподіл імовірності доставки одного пакета повідомлення. Такий розподіл називається розподілом Паскаля, що задає ймовірність першої вдалої спроби в серії експериментів.

Для побудови розподілу ймовірності двох пакетів, потрібно об'єднати розподілу [1]. Два пакети вимагають уже почергової передачі, тому ймовірність, що два пакети прийдуть за i квантів часу τ_p є сума добутків всіх варіантів $p(j)p(k)$ при $j+k=i$. Тоді для двох пакетів з розподілами (3.2) одержимо:

$$p_2(i)=\sum_{j=1}^{i-1}p_1(j)\cdot p_1(i-j)\rightarrow p_2(i)=\sum_{j=1}^{i-1}pq^{j-1}\cdot pq^{i-j-1}\rightarrow p_2(i)=p^2\sum_{j=1}^{i-1}q^{i-2},$$

де p_2 – імовірність того що два пакети будуть доставлені за час $i\tau_p$.

Остаточно:

$$p_2(i)=p^2q^{i-2}(i-1),\ i\geq 1. \qquad (3.5)$$

Аналогічно можна одержати розподіл для передачі трьох пакетів:

$$p_3(i)=\sum_{j=1}^{i-1}p_2(j)\cdot p(i-j)\rightarrow p_3(i)=p^3\sum_{j=1}^{i-1}(j-1)q^{i-3}\rightarrow p_3(i)=p^3q^{i-3}\sum_{j=1}^{i-1}(j-1),$$

Сума представляє арифметичну прогресію, сума якої дорівнює:

$$S_1=\sum_{j=1}^{i-1}(j-1)\rightarrow S_1=\frac{(i-1)(i-2)}{2}.$$

Звідки одержуємо загальне вираження, придатне для використання без підсумовування:

$$p_3(i)=p^3q^{i-3}\frac{(i-1)(i-2)}{2}. \qquad (3.6)$$

Тепер, для полегшення здогаду про вид загальної формули для знаходження ймовірності передачі N пакетів за i спроб, знайдемо $p_4(i)$. Для цього знову використовуємо згортку розподілів імовірностей:

$$p_4(i)=\sum_{j=1}^{i-1}p_3(j)p_1(i-j)\rightarrow p_4(i)=\sum_{j=1}^{i-1}p^3q^{j-3}\frac{(j-1)(j-2)}{2}pq^{i-j-1},$$

$$p_4(i)=p^4\sum_{j=1}^{i-1}q^{j-3+i-j-1}\frac{(j-1)(j-2)}{2}\rightarrow p_4(i)=\frac{p^4q^{i-4}}{2}\sum_{j=1}^{i-1}(j-1)(j-2).$$

Знайдемо значення суми $S_4(i)=\sum j(j-1)$ зі зміненими границями для більше простих викладень. Для цього приймемо що $f(j)=j^2-j$, і

$S_4(i) = f(0) + f(1) + f(2) ... + f(i-2)$. Щоб знайти суму, побудуємо $F(j) = Aj^3 + Bj^2 + Cj$ таке, що $f(j) = F(j+1) - F(j)$. Тоді $f(j) = A(j+1)^3 + B(j+1)^2 + C(j+1) - Aj^3 - Bj^2 - Cj$, розкривши дужки, одержимо $f(j) = j^2(3A) + j(3A + 2B) + (A + B + C)$. З останньої рівності одержимо: $j^2 - j = j^2(3A) + j(3A + 2B) + (A + B + C)$. Тепер можна записати систему рівнянь для визначення шуканих коефіцієнтів: $\begin{cases} 3A = 1 \\ 3A + 2B = -1 \\ A + B + C = 0 \end{cases} \rightarrow \begin{cases} A = 1/3 \\ B = -1 \\ C = 2/3 \end{cases}$, звідки $F(j) = j(j-1)(j-2)/3$. Замінимо суму $f(j)$ різницями $F(j+1)$-$F(j)$:

$$S_4(i) = (F(1) - F(0)) + (F(2) - F(1)) + ... + (F(i-1) - F(i-2)).$$

Скорочуючи $F(i)$ однаковими аргументами, одержимо вираження для пошуку суми:

$$S_4(i) = F(i-1) - F(0) \rightarrow S_4(i) = \frac{(i-1)(i-2)(i-3)}{3}.$$

На основі отриманого вираження для пошуку суми $S_4(i)$, запишемо загальну формулу для пошуку ймовірності відправлення чотирьох пакетів за i квантів часу (3.7).

$$p_4(i) = p^4 q^{i-4} \frac{(i-1)(i-2)(i-3)}{2 \cdot 3}. \qquad (3.7)$$

На основі формул (3.5), (3.6) і (3.7) можна припустити, що загальний вид формули має вигляд:

$$p_N(i) = p^N q^{i-N} \frac{(i-1)(i-2)...(i-(N-1))}{(N-1)!}. \qquad (3.8)$$

Доведемо що формула (3.8) дає ймовірність відправлення N пакетів інформації за i квантів часу.

Нам уже відома справедливість формул (3.5), (3.6) і (3.7). Тому, твердження можна довести методом математичної індукції встановивши

$$p_{N+1}(i)=\sum_{j=1}^{i-1}p_N(j)p_1(i-j)$$

справедливість рівності: . Проведемо підстановки й винесемо константи за знак підсумовування:

$$p_{N+1}(i)=\sum_{j=1}^{i-1}p^N q^{j-N}\frac{(j-1)(j-2)...(j-(N-1))}{(N-1)!}pq^{i-j-1},$$

$$p_{N+1}(i)=\frac{p^{N+1}q^{i-(N+1)}}{(N-1)!}\sum_{j=1}^{i-1}(j-1)(j-2)...(j-(N-1)).$$

Аналогічно виводу формули (3.7) приймемо за $f(j)=j(j-1)(j-2)...(j-(N-2))$, тоді $F(j)=j(j-1)(j-2)...(j-(N-1))/N$.

Перевіримо правильність здогаду про вид *F*(*j*), що є антирізницю узагальненого ступеня в різницевому вирахуванні:

$$f(j)=F(j+1)-F(j)\rightarrow\prod_{k=0}^{N-2}(j-k)=\frac{1}{N}(j+1)\prod_{k=0}^{N-2}(j-k)-\frac{1}{N}\prod_{k=0}^{N-1}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k)=((j+1)-(j-(N-1)))\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k)=(j+1-j+N-1)\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

$$\prod_{k=0}^{N-2}(j-k)=N\frac{1}{N}\prod_{k=0}^{N-2}(j-k),$$

що підтверджує правильність припущення про вид *F*(*j*). Тепер їсти можливість замінити підсумовування вираженням *F*(i-*1*)-*F*(0):

$$p_{N+1}(i)=\frac{p^{N+1}q^{i-(N+1)}}{(N-1)!}\sum_{j=1}^{i-1}f(j-1),$$

$$\sum_{j=1}^{i-1}f(j-1)=\sum_{j=1}^{i-1}(F(j+1-1)-F(j-1)),$$

$$\sum_{j=1}^{i-1}f(j-1)=F(i-1)-F(0),$$

$$\sum_{j=1}^{i-1} f(j-1) = \frac{(i-1)(i-2)...(i-N)}{N},$$

$$p_{N+1}(i) = \frac{p^{N+1} q^{i-(N+1)}}{(N-1)!} \cdot \frac{(i-1)(i-2)...(i-N)}{N},$$

і остаточно спрощуючи:

$$p_{N+1}(i) = p^{N+1} q^{i-(N+1)} \frac{(i-1)(i-2)...(i-N)}{N!},$$

що відповідає формулі (3.8) при підстановці N:=N+1.

Твердження доведене.

Використовуючи розподіл імовірності (3.8), побудуємо математичне очікування й дисперсію кількості спроб на відправлення N пакетів інформації.

Середній статистичний час і дисперсія доставки повідомлення з N пакетів

Раніше зроблене припущення про лінійність часу доставки інформації від кількості пакетів, які ця інформація займає. Для повної впевненості потрібно одержати математичне очікування часу доставки пакетів аналітично з розподілу ймовірності (3.8).

Раніше показано, що ймовірність доставки повідомлення з одного пакета за i спроб (1) $p_1(i) = pq^{i-1}$. Математичне очікування кількості спроб для успішної передачі пакета буде $M(p_1) = \sum_{i=1}^{\infty} pq^{i-1} i$. Для знаходження аналітичного подання значення суми ряду приймемо за $f(i) = q^i i$ вираження під знаком суми, і знайдемо $F(i)$ таке що $F(i+1) - F(i) = f(i)$. Шукати $F(i)$ будемо у вигляді $F(i)=(Ai+B)q^i$. Тоді

$$F(i+1) - F(i) = (Ai + A + B)q^{i+1} - (Ai + B)q^i \rightarrow f(i) = q^i(iA(q-1) + q(A+B) - B)$$

.

Для рівності необхідно щоб $\begin{cases} A(q-1)=1 \\ q(A+B)-B=0 \end{cases}$, звідки $\begin{cases} A=-\dfrac{1}{1-q} \\ B=-\dfrac{q}{(1-q)^2} \end{cases}$.

Далі підставляючи й зводячи подібні одержуємо:

$$\sum_{i=1}^{\infty} f(i)=F(\infty+1)-F(1) \rightarrow \sum_{i=1}^{\infty} f(i)=0-\left(-\frac{1}{1-q}-\frac{q}{(1-q)^2}\right)q,$$

$$M(p_1)=\frac{p}{q}\sum_{i=1}^{\infty} f(i),$$

$$M(p_1)=\frac{p}{q}\left(\frac{1}{1-q}+\frac{q}{(1-q)^2}\right)q,$$

$$M(p_1)=(1-q)\left(\frac{1}{1-q}+\frac{q}{(1-q)^2}\right).$$

Спростивши останнє вираження, одержуємо середню математичну кількість спроб для передачі одного пакета даних, без обліку часу на встановлення каналу зв'язку й маршрутизації:

$$M(p_1)=\frac{1}{1-q}. \qquad (3.9)$$

З урахуванням того факту, що при одержанні розподілів p використане додавання незалежних випадкових величин, математичні очікування й дисперсії підсумуються:

$$M(p_N)=\frac{N}{1-q}, \qquad (3.10)$$

що погодиться з раніше отриманим (3.3). Однак, з метою доказу правильності одержання узагальненої формули розподілу ймовірності (3.8), знайдемо значення середнього часу очікування доставки інформації аналітично.

По визначенню середньостатистичного

$$M(p_N) = \sum_{i=1}^{\infty} i p^N q^{i-N} \frac{(i-1)(i-2)...(i-(N-1))}{(N-1)!}.$$

З метою спрощення записів у подальших викладеннях перепишемо останнє вираження

$$M(p_N) = \frac{p^N}{q^N (N-1)!} \sum_{i=0}^{\infty} q^i i^{(N)},$$

де $i^{(N)}=i(\ i\text{-}1)(\ i\text{-}2)...(\ i\text{-}i\text{-}(N\text{-}1))$ – різницевий ступінь, змінена границя суми від значення 0. Зміна границі підсумовування не змінить значення суми, тому що при i=0 значення під знаком суми має значення нуль. Використовуючи $F(i)$, для якого $F(i+1)\text{-}F(i)=f(i)$, де $f(i)=q^i i^{(N)}$, можна виразити значення суми: $\sum_{i=0}^{\infty} f(i) = F(\infty) - F(0)$. Шукати $F(i)$ потрібно у вигляді:

$$F(i) = q^i \left(A_0 + A_1 i + A_2 i^{(2)} + ... + A_{N-1} i^{(N-1)} + A_N i^N\right).$$

З метою використання більше короткого запису введемо позначення $\Delta F(i) = F(i+1) - F(i)$. Тоді $\Delta F(i) = \Delta G(i) \cdot V(i) + G(i) \cdot \Delta V(i) + \Delta G(i) \cdot \Delta V(i)$, позначивши $G(i) = A_0 + A_1 i + A_2 i^{(2)} + ... + A_{N-1} i^{(N-1)} + A_N i^N, V(i) = q^i$. Знайдемо різниці:

$$\Delta G(i) = A_1 + 2A_2 i^{(1)} + ... + (N-1) A_{N-1} i^{(N-2)} + N A_N i^{N-1}, \Delta V(i) = -(1-q) q^i .$$

$$\Delta F(i) = q^i \left((qA_1 - (1-q)A_0) + (q2A_2 - (1-q)A_1) i^{(1)} + ...\right.$$

$$\left.)... + (qkA_k - (1-q)A_{k-1}) i^{(k-1)} + ... + (qNA_N - (1-q)A_{N-1}) i^{(N-1)} - (1-q)A_N i^{(N)} .\right.$$

Дорівнюємо коефіцієнти $\Delta F(i)$ і $f(i)$ при рівних ступенях і одержуємо систему рівнянь:

$$-(1-q)A_N = 1,$$

$$qNA_N - (1-q)A_{N-1} = 0,$$

$$q(N-1)A_{N-1} - (1-q)A_{N-2} = 0,$$

...

$$q(k-1)A_{k-1} - (1-q)A_{k-2} = 0,$$

...

$$q3A_3 - (1-q)A_2 = 0,$$

$$q2A_2 - (1-q)A_1 = 0,$$

$$qA_1 - (1-q)A_0 = 0.$$

Вирішивши систему визначаємо що $A_0 = \frac{-q^N N!}{(1-q)^{N+1}}$. З огляду на, що $F(\infty)\to 0$, знаходимо значення суми

$$M(p_N) = \frac{p^N}{q^N (N-1)!}\left(0 - \frac{-q^N N!}{(1-q)^{N+1}}\right),$$

$$M(p_N) = \frac{N}{1-q},$$

яка збігається з формулою (3.10). Цей результат доводить правильність знаходження аналітичного вираження (3.8) для пошуку розподілу ймовірності.

Для дисперсії аналогічно можна одержати $D(p_N) = \sum_{n=1}^{N} D(p_1)$, звідки $D(p_N) = N \cdot D(p_1)$. Визначимо значення дисперсії для відправлення одного пакета:

$$D(p_1) = \sum_{i=1}^{\infty} p \cdot q^{i-1} i^2 - M^2(p_1) \to D(p_1) = \frac{p}{q}\sum_{i=1}^{\infty} q^i i^2 - M^2(p_1).$$

Далі потрібно знайти значення суми ряду $\sum_{i=1}^{\infty} q^i i^2$. Приймемо що $f(i)=q^i i^3$. Будемо шукати $F(i)$ таке, що $f(i)=F(i+1)-F(i)$. Антирізниця $F(i)$ має вигляд: $F(i) = (Ai^2 + Bi + C)q^i$. Звідки одержимо

$$F(i+1) - F(i) = q(A(i+1)^2 + B(i+1) + C)q^i - (Ai^2 + Bi + C)q^i.$$

Спрощуючи

$$f(i) = (q(Ai^2 + 2\mathrm{A}i + A + Bi + B + C) - (Ai^2 + Bi + C))q^i,$$

$$f(i)=\left((q-1)Ai^2+(2\mathrm{A}q+B(q-1))i+(qA+qB+C(q-1))\right)q^i,$$

одержуємо систему рівнянь для знаходження шуканих коефіцієнтів:

$$\begin{cases}(q-1)A=1\\ 2\mathrm{A}q+B(q-1)=0\\ qA+qB+C(q-1)=0\end{cases},\quad \begin{cases}A=-1/(1-q)\\ B=-2q/(1-q)^2\\ C=-q(1+q)/(1-q)^3\end{cases}.$$

Тепер можна підставити знайдені коефіцієнти для пошуку суми ряду:

$$\sum_{i=1}^{\infty}f(i)=F(\infty+1)-F(1),\quad \sum_{i=1}^{\infty}f(i)=0-\left(\frac{-1}{1-q}-\frac{2q}{(1-q)^2}-\frac{q(1+q)}{(1-q)^3}\right)q,$$

$$\sum_{i=1}^{\infty}f(i)=\left(\frac{1}{1-q}+\frac{2q}{(1-q)^2}+\frac{q(1+q)}{(1-q)^3}\right)q.$$

Після цього можна записати й спростити вираження для обчислення дисперсії:

$$D(p_1)=(1-q)\left(\frac{1}{1-q}+\frac{2q}{(1-q)^2}+\frac{q(1+q)}{(1-q)^3}\right)-\frac{1}{(1-q)^2},$$

$$D(p_1)=1+\frac{2q}{1-q}+\frac{q(1+q)-1}{(1-q)^2},$$

$$D(p_1)=\frac{q}{(1-q)^2}.$$

І відповідно до властивості підсумовування дисперсій, для передачі N пакетів інформації маємо дисперсію:

$$D(p_N)=N\frac{q}{(1-q)^2}. \qquad (3.11)$$

де q – імовірність відкладеної передачі пакета, через ненадійність каналу передачі або очікування в черзі.

Також існує можливість відновити ймовірність втрати пакета або його затримки в черзі на проміжних вузлах мережі по експериментально певній

дисперсії часу доставки пакетів інформації, виразивши цю ймовірність із вираження (3.11):

$$q = 1 + \frac{N - \sqrt{N(4D_N + N)}}{2D_N} \quad (3.12)$$

.

У реальній мережі пакети передаються по ланцюзі сегментів, у кожній з яких відбуваються затримки в передачі пакета. Однак, як показано раніше, кожний сегмент має характеристику середнього часу на доставку пакета, що дає для повного часу передачі пакета від відправника до одержувача суму часів очікувань на окремих сегментах. Також цей вивід можна підтвердити теоремою про послідовні випадкові події з розподілом часу очікування $f_1(t)$ і $f_2(t)$. В [3] доведене, що математичне очікування об'єднаного розподілу $M(f_1(t)*f_2(t))$ послідовні настання обох подій, буде сума математичних очікувань розподілів $M(f_1(t))$ і $M(f_2(t))$:

$$M(f_1(t) * f_2(t)) = M(f_1(t)) + M(f_2(t)). \quad (3.13)$$

Практично це означає, що для ланцюга сегментів буде отриманий аналогічний розподіл із середнім часом очікування повідомлення, що дорівнює сумі середніх часів очікування повідомлення на окремих сегментах мережі (за умови незмінності маршруту).

Закон додавання математичних очікувань при додаванні розподілів послідовних випадкових величин дає важливий вивід: ***ділянку багатосегментної мережі можна замінити статистично рівноцінним односегментним зв'язком.***

Факт відповідності теоретичних результатів з експериментально отриманими характеристиками мережі при багаторазовому завантаженні файлів.

Як результат можна виділити:

1. Час доставки інформації в телекомунікаційній мережі має лінійну залежність від кількості цієї інформації.

3. Зменшення кількості переданої інформації в k раз, у стільки ж раз зменшує середній час доставки інформації, а також в k раз зменшує дисперсію часу доставки повідомлення.

3. Експериментальне знаходження дисперсії часу доставки повідомлення дозволяє оцінити ймовірність відкладеної передачі пакета інформації з вираження (3.12).

4. Використання додаткового стиску інформації й відкидання незначної інформації не тільки скоротить час її доставки, але й за рахунок зменшення дисперсії, також знизить відхилення від середнього часу доставки. Цей результат можна використовувати практично, наприклад, для розробки протоколу більше точної синхронізації годин за допомогою телекомунікаційної мережі.

3.2 Розробка структурної схеми

Розробка методу підвищення ефективності передачі інформації в телекомунікаційних корпоративних системах і мережах за допомогою вдосконаленого прогресивного кодеку графічного контенту на основі SPIHT

Як показано вище, на якість роботи системи сильно впливає розмір додатково стислої графічної інформації [5, 6, 8]. Зменшити кількість переданої інформації можна використовуючи більше зроблені методи кодування зображень. Наприклад, тривимірний варіант вейвлет перетворення був використаний для створення більше зробленого кодування відео.

Порівняння бітових щільностей при використанні різних методів кодування графічної інформації

Основними методами, які претендують на використання в системі додаткового стиску, є JPEG2000 і SPIHT. Однак можна показати, що надмірність інформації в упакованому SPIHT зображенні має більший ступінь. Це можна показати, установивши ентропію інформації результату впакування зображення.

З огляду на, що кодування в обох методах побітове, установимо відношення між одиничним бітом і нульовим.

З наведених графіків видна нерівномірність розподілу нульових і одиничних бітів у вихідному не кодованому зображенні. При одержанні кодованого зображення JPEG2000, відбувається рятування від надлишкової інформації, що приводить до рівномірного розподілу ймовірностей зчитування одиничного й нульового біта.

Це говорить про якісний стиск JPEG2000, але SPIHT кодування по своїй суті віддає перевагу передачі безлічами по бітових шарах, що приводить до більше частої передачі нульового біта, а після, коли істотних пікселів ставати більше – частіше передається одиниця.

Ситуація з нерівномірним розподілом співвідношення нульових і одиничних бітів є наслідком особливостей SPIHT кодування й відсутності додаткового стиску, що є присутнім в JPEG2000. Для ситуації нерівномірного побітового кодування, коли один біт має дробове значення кількості інформації в результуючому файлі, підходить реалізація математичного стиску [2].

Використання постійного коефіцієнта для ймовірностей влучення 0 або 1, для одержання мінімального розміру вихідного файлу ці ймовірності повинні мінятися. Таке кодування буде називатися адаптивним, одним з таких алгоритмів є Q-Кодування. Однак у ході адаптивного кодування параметри кодеку міняються із запізнюванням, що негативно позначається на ступені стиску. Тому необхідно передбачити блокове кодування, для якого окремо визначають імовірності знаків. Для цього необхідно визначитися з розміром блоку й точністю завдання ймовірності надходження одиничного біта.

Нехай для запису відносини частки одиничних бітів на N_i бітів файлу використовується запис із фіксованою крапкою розміром в b біт: $p(1)$= 1-p(0). Тоді розмір коду буде

$$N_i^* = -N_i\left[p(0)\log_2 p(0) + p(1)\log_2 p(1)\right] + b, \qquad (3.14)$$

де N^*_i розмір упакованої частини N_i, $p(0)$ імовірність появи нульового біта рівного N_{i0}/N_i, $p(1)$ імовірність появи одиничного біта рівного N_{i1}/N_i. Використовуючи формули визначення ймовірностей, виключаємо з (3.14) кількості одиничних і нульових біт, залишивши тільки ймовірність одиничного біта й розмір кодованої частини зображення:

$$N_i^* = -N_i\left(\left(1-p(1)\right)\log_2\left(\left(1-p(1)\right)\right)+p(1)\log_2\left(p(1)\right)\right)+b, \qquad (3.15)$$

де перший доданок залежить від властивостей інформації, а друге є константою з фіксованої коми розміром в 16 або 8 біт, що містить значення ймовірності $p(1)$. Повний розмір файлу при цьому буде сумою розмірів упакованих блоків:

$$N^* = -\sum N_i\left(\left(1-p(1)\right)\log_2\left(1-p(1)\right)+p(1)\log_2 p(1)\right)+n\cdot b, \qquad (3.16)$$

де n – кількість частин для впакування зображення. Коефіцієнт стиску ентропійним методом залежно від $p(1)$ показане на мал. 3.5. При цьому в частині для кодування спостерігається коливання ймовірності появи 1, тому ступінь стиску маленьких шматків даних повинні дати кращий результат. Однак, при збільшенні кількості частин, потрібно використовувати більшу кількість інформації для запису значення ймовірностей, доданок nb у формулі (3.16).

Це приводить до завдання оптимізації:

$$N^*(n) = \frac{-N}{n}\sum\left[\left(1-p_i(1)\right)\log_2\left(1-p_i(1)\right)+p(1)_i\log_2 p_i(1)\right]+n\cdot b \to min. \qquad (3.17)$$

Зміст такої оптимізації складається в більше точному визначенні ймовірності появи одиничного біта на ділянці коду, але для запису такого коефіцієнта теж потрібно виділяти певну кількість інформації. При збільшенні кількості проміжків n удвічі, кількість інформації для запису коефіцієнтів теж виростає вдвічі. Цей ріст повинен бути компенсований поліпшенням ступеня стиску зображення. Приклад підбора кількості проміжків показане на мал. 3.6. Виразно, що при надлишковому розподілі потоку інформації на блоки, ступінь стиску може не тільки погіршитися, але й у результаті можна одержати файл більшого розміру, ніж вихідний.

Ухвалити рішення щодо подальшому зменшенні блоків з узагальненою інформацією об частки одиничних бітів можна з аналізу формули (3.15). Допустимо, що розмір блоку зменшується вдвічі, при цьому позначимо:

$$p_{2i,2n}(1) = p_a, p_{2i+1,2n}(1) = p_b, p_{i,n} = p, p = (p_a + p_b)/2.$$

Зрозуміти в скільки разів змінитися ступінь стиску інформації одного блоку можна зі співвідношення:

$$K = \frac{2(p\log_2 p + (1-p)\log_2(1-p))}{p_a\log_2 p_a + (1-p_a)\log_2(1-p_a) + p_b\log_2 p_b + (1-p_b)\log_2(1-p_b)}. \qquad (3.18)$$

$K(p_a,p_b)$ – симетрична щодо аргументів функція, що дорівнює 1 при $p_a=p_b$, що означає відсутність зміни в розмірі стисливої інформації й у цьому випадку користі від розбивки потоку даних на більше дрібні частини не буде. Коли $p_a \neq p_b$, те K>1, і це означає зменшення вихідної кількості інформації на передачу.

Ступінь стиску інформації значно зростає при максимальній різниці між p_a і p_b. Для ухвалення рішення про зменшення блоків кодування необхідно мати оцінку |pa-pb.|. Зробити оцінку можна за допомогою поняття фрактальної розмірності числової послідовності [1].

З метою показати застосовність визначення фрактальної розмірності для оцінки поліпшення стиску інформації розглянемо два крайніх випадки:

1) Фрактальна розмірність F близька до 2. При збільшенні числа звітів кривій, ця крива поступово заповнює площину – нові злами мають розмах порівнянний з попередніми. Як наслідок, з (3.17) ми одержуємо високий коефіцієнт стиску.

2) Фрактальна розмірність F ближче 1. У цьому випадку уточнена крива буде практично повторювати попередню, і уточнюючи ймовірність вибірки одиниці збільшити коефіцієнт стиску інформації не вдасться.

Для послідовності ймовірностей $\{p_0,p_1,...,p_{2\ n-1}\}$ на більших блоках одержимо набір імовірностей $\{(p_0+p_1)/2,(p_2+p_3)/2,...,(p_{2\ n-2}+p_{2\ n-1})/2\}$ на умовному одиничному відрізку. Довжини кривих для першої й другої послідовностей будуть:

$$L_1 = \sum p_i \frac{1}{2\text{n}}, \quad L_2 = \sum \frac{p_{2\text{i}} + p_{2\text{i}+1}}{2} \cdot \frac{1}{n}.$$

Локальна фрактальна розмірність кривої буде

$$F(n) = -\log_2(L_2/L_1).$$

Розмірність отут розуміється як число F, що показує, у який ступінь потрібно звести a, що б одержати коефіцієнт подовження кривій, зменшивши «лінійку» в a раз. При довжині L лінія має n відрізків $\{(p_i,i/n);(p_{i+1},(i+1)/n)\}$. Середня довжина одного відрізка буде $<L>=L/n$. При збільшенні кількості блоків від n_1 до n_2 ($n_1<n_2$), тоді подовження лінії буде приблизно дорівнює $L_2 \approx L_1(n_2/n_1)^F$ і $<L_2>=L_1(n_2/n_1)^F/n_3$. Тут і далі позначення $<L>$ використовується для середнього арифметичного або математичного очікування. Остаточно, середня різниця між послідовними значеннями ймовірностей появи одиничного біта буде

$$\langle|\Delta p_2|\rangle = \langle|\Delta p_1|\rangle \left(\frac{n_2}{n_1}\right)^{F-1}. \qquad (3.19)$$

Для початкової розбивки послідовності бітів, що складається з n_1 сегментів фрактальної розмірності F, перейшли до розбивки з $n_2=zn_1$ сегментів. З (3.19) треба

$$\langle|\Delta p_2|\rangle = \langle|\Delta p_1|\rangle z^{F-1}. \qquad (3.20)$$

Зміст цього вираження полягає в постійному зростанні розходження щільності одиничних біт при розгляді усе більше коротких підпослідовностей з головної послідовності. Зі співвідношень (3.18) і (3.20) треба збільшення коефіцієнта стиску K до деякої межі. Найбільш песимістичною оцінкою зміни коефіцієнта стиску буде, якщо прийняти p=0,5, p_a= p-p-Δp/2, p_b=p+Δp/2:

$$K(F,z) = \frac{z\log_2 0{,}5}{\sum (0{,}5 + z^{F-1}/2)\log_2(0{,}5 + z^{F-1}/2) + (0{,}5 - z^{F-1}/2)\log_2(0{,}5 - z^{F-1}/2)} \qquad (3.21)$$

.

Однак, при доповненні кожного блоку інформацією про діл одиничних бітів, буде потрібно додаткова інформація розміром в zb. Завдання оптимізації (3.17) зводяться до відшукання мінімуму рівняння з однієї натуральної змінної z:

$$N / K(F,z) + zb \to min\,. \qquad (3.22)$$

Цільова функція (3.22) не лінійна, тому її мінімум шукається за допомогою чисельних методів. Завдання є простій, оскільки *z* може приймати тільки натуральні значення й має тільки один мінімум через монотонність другій похідній (3.21) і монотонності *zb*.

Принцип поділу бітових полів в SPIHT кодуванні і його модифікації за принципом мінімізації відносних погрішностей

Перегляд графічної інформації в телекомунікаційних мережах у більшості випадків відбувається у два етапи, вибір актуального контенту, а потім його перегляд. При оцінці актуальності графічного контенту, користувач комунікаційної мережі не має потреби в максимально припустимій якості. В особливих випадках, коли пропускна здатність каналу передачі істотно низька, виникає проблема оперативної доставки зображення з максимальною якістю й обмеженням за часом доставки. У такому випадку заздалегідь невідома кількість інформації, що одержить приймач, і потрібно забезпечити передачу найбільш інформативних частин у першу чергу. SPIHT кодування найбільше відповідає поставленим вимогам, за критерієм першочерговості передачі максимальної енергії сигналу [5].

Класичне використання алгоритму для прогресивної передачі вейвлет коефіцієнтів перетвореного зображення розділяє масив коефіцієнтів на істотні й не істотні коефіцієнти й не істотні безлічі. У не істотних безлічах кожний з коефіцієнтів менший певної величини. Результатом є передача коефіцієнтів тільки старших бітів значимих вейвлет коефіцієнтів. Наступним кроком список істотних пікселів збільшується, і передається інформація про значення бітів наступної бітової площини. Фактично, для байтовых коефіцієнтів, передається інформація про зміст біта 128, що випливає кроком про наявність бітів для 64, і так далі. При цьому наявність малопотужних складових сплесків передається в останню чергу, угруповання малих коефіцієнтів у не істотну безліч дозволяє одним бітом передати нульовий біт для всього безлічі пікселів. За рахунок цього

й досягається зменшення кількості бітів для передачі. У випадку припинення передачі вмісту файлу, будуть прийняті коефіцієнти в неповному виді з рівномірною втратою молодших бітів.

Аналіз робіт дозволяє сформулювати висновок, що підтверджує актуальність покращеного відображення контурної інформації за рахунок зниження загальної якості зображення [5].

Для поліпшення деталізації зображення пропонується передавати наступні значимі біти, зберігши відносну погрішність коефіцієнтів.

При цьому уточнення амплітуди довгих сплесків відбувається тільки після одержання інформації про всі деталі зображення. Візуально неповно прийняте зображення буде мати зональні перекручування яскравості й кольоровості без втрат малоконтрастних деталей.

Фактично це означає, що при меншій кількості прийнятої інформації око людини побачить більше деталей на зображенні, у якому зберігалася в першу чергу відносна погрішність вейвлет коефіцієнтів.

Наслідком перестановки послідовності біт передачі розмір переданої інформації й ступінь стиску не змінитися, однак зростає кількість переданих деталей при неповній передачі даних або при сильних коефіцієнтах стиску. У режимі попередньої оцінки актуальності зображення це може скоротити трафік передачі на 1%-50% залежно від типу зображення.

Модифікація проводитися на етапі передачі значимих бітів. При цьому значимі біти передаються тільки після того, як на приймачі вже буде отримана інформація про знак і старший одиничний біт всіх вейвлет коефіцієнтів.

Така модифікація можлива завдяки вже створеним спискам значимих вейвлет коефіцієнтів з однозначною їхньою послідовністю. Тому черговість передачі значимих бітів повністю повторюється на приймачі графічної інформації.

Розробка структури й алгоритмів роботи серверів додаткового стиску графічного контенту телекомунікаційних корпоративних систем і мереж

Розробка порядку роботи системи підвищення оперативності доставки часто використовуваної інформації за рахунок зниження кількості передачі низько пріоритетної інформації вимагає більше докладного опису.

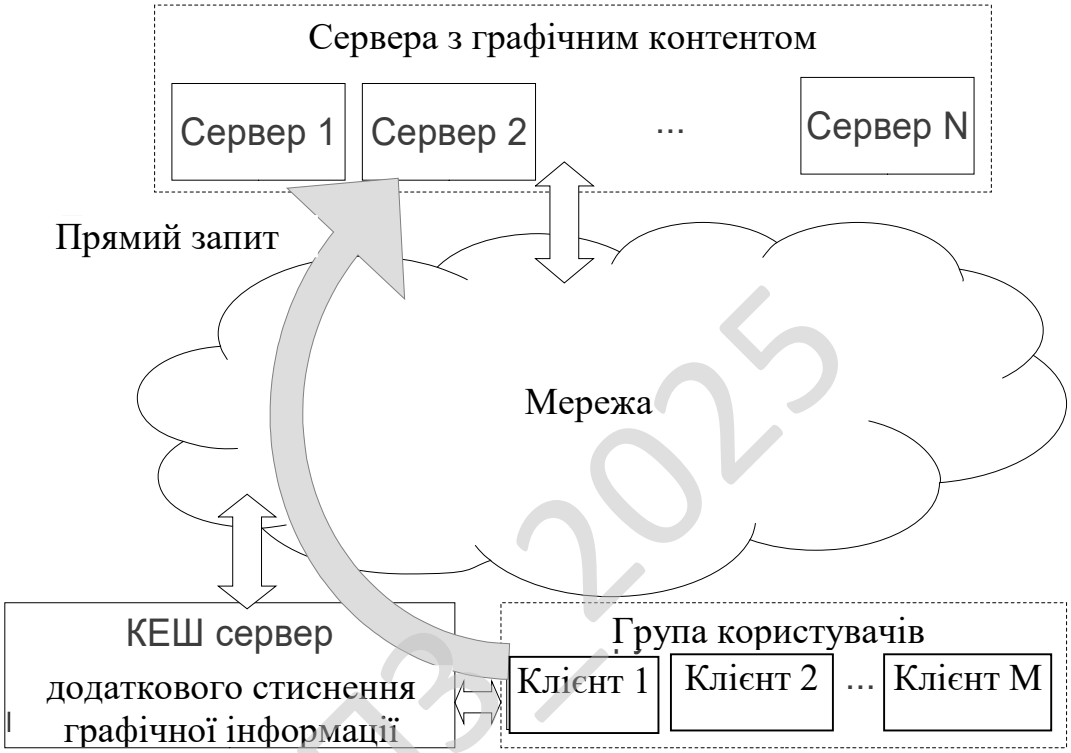

Рисунок 3.1 – Структурна схема системи

Поряд з використанням більше якісного прогресивного стиску інформації необхідно враховувати ситуації, при яких інформація відсутня на проміжному сервері й запит іде до оригінальних даних. При цьому кінцевий користувач не буде очікувати додаткового стиску, можливо тривалого, через більше складний алгоритм кодування. Наповненість сервера й частота запитів до інформації, що відсутній на сервері, сильно впливає на результуючу оперативність доставки інформації. Тому потрібно враховувати динаміку наповнення сервера.

Структура, диктує алгоритм взаємодії клієнта із запропонованою системою.

Відповідно до алгоритму, взаємодія починається з оцінки завантаженості по обчислювальних можливостях і пропускній здатності, що узгоджується із сервером автоматично у фоновому режимі. За результатами оцінки, клієнт автоматично повинен частину запитів проводити в класичному режимі, з метою не допустити глобальної відмови в обслуговуванні. Також, за результатами оцінки завантаженості, адміністраторам буде забезпечено амортизаційний час для нарощування обчислювальних можливостей устаткування.

У випадку низької завантаженості серверів, або ухвалення рішення звертання до сервера перепакованої інформації при високій завантаженості, клієнт відправляє запит до сервера додаткового стиску. На сервері вироблятися перевірка наявності запитаної інформації в перепакованому виді й, при її наявності, чорнова копія відправляється клієнтові. Інакше, клієнтові відсилається оригінальний варіант без додаткового стиску й, окремо створюється перепакована копія оригінальної інформації. При повторному запиті, від цього або іншого клієнта буде переданий чорновий варіант інформації, заощаджуючи трафік на передачі низькопріорітетної інформації.

Одержавши чорновий варіант, клієнт має можливість оцінити цінність повної якості або тонко регулювати необхідну якість одержуваної інформації. Завдяки прогресивному кодуванню, є можливість уникнути повного завантаження інформації, досить одержати її відсутню частину.

Розробка математичної моделі роботи серверів додаткового стиску графічного контенту телекомунікаційних корпоративних систем і мереж

Вивчення поводження системи додаткового стиску графічної інформації можливо або на математичній моделі або її реалізації, що означає розробку й впровадження без упевненості в перспективності методу. Це накладає додаткові умови моделювання поводження системи в процесі її роботи з урахуванням динамічної зміни контенту мережі й кількості клієнтів цієї системи

Математична модель поряд з уведеними позначеннями: T_3, T_n, k, S, S_k, містить у собі й $L(t)$ – кількість активних користувачів системи, а L – їхнє

максимальне значення; відношення більше повільного сегмента підключення до глобальної мережі в r раз, що відповідно збільшить час завантаження T_3 r; і вже отримане співвідношення визначення коефіцієнта збільшення швидкості передачі (1.5).

Коефіцієнт k не постійний. Він виражає ймовірність обігу клієнта до інформації, що вже перебуває на сервері й необхідності звертання до оригінального контенту немає. З огляду на той факт, що інформація постійно додається із середньою швидкістю $v_д$ і застаріває зі швидкістю $v_у$, контент графічної інформації постійно міняється, $dS = S(t+dt) - S(t)$. Цей показник є характеристикою мережі.

Показник швидкості додавання інформації на сервер додаткового стиску графічної інформації залежить від частоти влучення повз завантажений контенту. Таке невлучення відбувається з імовірністю 1-k від загальної кількості запитів користувачів. Відповідно:

$$dS_k \sim (1-k)L(t)dt .$$

З метою більше повного опису додамо позначення:

S_n – кількість зображень не в кэш-сервері, S_n=S(1-k), S_n= S-Sk;

v_n – швидкість додавання нових зображень у мережі, v_n=d/dt;

v_o – швидкість збільшення кількості застарілої інформації, отут прийнята як постійна частина від загальної кількості зображень v_o=γS, 0<γ<1 – коефіцієнт пропорційності;

v_k – швидкість додавання нових зображень у базу даних, залежить від кількості користувачів системи. Кожний запит до нової інформації, який ще немає на сервері, імовірність такого запиту 1-k=S_n/S, супроводжується додаванням нового контенту в базу. Відповідно, кількість доданого контенту за одиницю часу пропорційно кількості активних користувачів системи L(t), або v_k=dS_k/dt.

Частки кешованого контенту можна описати наступним відношенням $k(t) = S_k / S$ і $k(t+dt) = \frac{S_k + dS_k}{S + dS}$. Одержимо співвідношення приросту частки графічної інформації в базі даних:

$$k(t+dt) - k(t) = \frac{Sk(t) + (v_k - k(t)v_o)dt}{S + (v_n - v_o)dt} - k(t).$$

Скоротивши й опустивши нескінченно малі, одержимо

$$\frac{dk}{dt} = \frac{v_k - kv_n}{S}.$$

Для моделювання зміни кількості активних користувачів системи була використана модель поширення слухів (рівняння Ферхюльста), відповідно до якої ймовірність повідомити новина незнаючому пропорційна кількості необізнаних людей і також пропорційна кількості обізнаних, які цю новину поширюють:

$$\frac{dL(t)}{dt} = \alpha L(t)(L - L(t)).$$

Використовуючи раніше уведені позначення, запишемо систему рівнянь динаміки розвитку системи додаткового стиску, що й буде її математичною моделлю:

$$\begin{cases} \frac{dk}{dt} = \frac{v_k - kv_n}{S} \\ \frac{dS}{dt} = v_n - v_o \\ v_k = \beta(1-k)L(t) \\ \frac{dL(t)}{dt} = \alpha L(t)(L - L(t)) \end{cases},$$

де α, β – коефіцієнти пропорційності моделі. Модель застосовна у випадку перевищення швидкості додавання графічного контенту в базу дані системи швидкості появи нового контенту в мережі.

Відомість рівнянь до загального дає диференціальне рівняння розвитку системи:

$$\frac{dk}{dt} = \left(\frac{(1-k)\beta L}{1+e^{-\alpha L t + C}} - k v_n \right) / S . \qquad (3.23)$$

Знаючи зміну в часі *k*, можна визначити зміна середнього часу одержання графічної інформації з мережі. Рівняння (3.23) вирішувалося чисельно, через гладкість функції, методом Ейлера. Для розрахунків прийнято, що всі постійні коефіцієнти рани одиниці, а коефіцієнт зміни розміру графічного контенту v_k дорівнює 0,5. При цьому коефіцієнт середнього часу доставки повідомлення залежно від часу експлуатації системи був розрахований по (1.5).

Зробимо оцінку отриманих результатів по моделі. Модель ілюструє неефективність використання системи з малою кількістю користувачів при їхньому швидкісному підключенні до мережі. Це приводить до передчасного зниження популярності сервісу й зупинці його розвитку. На етапі становлення системи, від того як буде спостерігатися приріст швидкості доставки повідомлень, необхідно передавати клієнтові не змінений контент. При цьому клієнти будуть одержувати інформацію без очікування її додаткового стиску. Однак при значному зниженні швидкості передачі інформації на етапі провайдер-клієнт, система буде ефективної відразу, і виграш за часом буде тільки збільшуватися в міру використання системи.

Існує поріг швидкості підключення до глобальної мережі, при якому система дає виграш у швидкості доставки повідомлень навіть без використання проміжного зберігання інформації. У цьому випадку економія часу на передачу меншої кількості інформації окупає витрати на одержання контенту з основного джерела й додатковому стиску інформації.

3.3 Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Схема складається з двох великих функціональних блоків:

– Блок компресії.

– Блок декомпресії.

На блок компресії подається вхідне зображення, яке необхідно стиснути.

Після цього це зображення послідовно, у блоці компресії, обробляється наступними модулями:

– Модуль розбиття зображення на регіони.

– Модуль створення доменного зображення.

– Модуль розбиття доменного зображення на блоки.

– Модуль здійснення афінних перетворень.

– Модуль пошуку доменів, та відображення найбільш схожих на регіони.

– Модуль заміни регіонів на домени.

– Модуль запам’ятовування коефіцієнтів афінних перетворень, положення доменних областей, та схеми розділення зображення на домени.

– Модуль формування стиснутого файлу.

Функціональний блок декомпресії складається з наступних модулів:

– Модуль створення початкового зображення.

– Модуль багатократного застосовування до початкового зображення відображення W.

– Модуль формування розпакованого файлу.

Після того, як стиснуте зображення буде піддано послідовній обробці вищеперерахованих модулів, отримується вихідне зображення.

Класи зображень

Статичні растрові зображення являють собою двовимірний масив чисел. Елементи цього масиву називають пікселями. Всі зображення можна підрозділити на дві групи – з палітрою й без неї. У зображень із палітрою в пікселі зберігається число – індекс у деякому одномірному векторі кольорів, називаному палітрою. Найчастіше зустрічаються палітри з 16 і 256 кольорів.

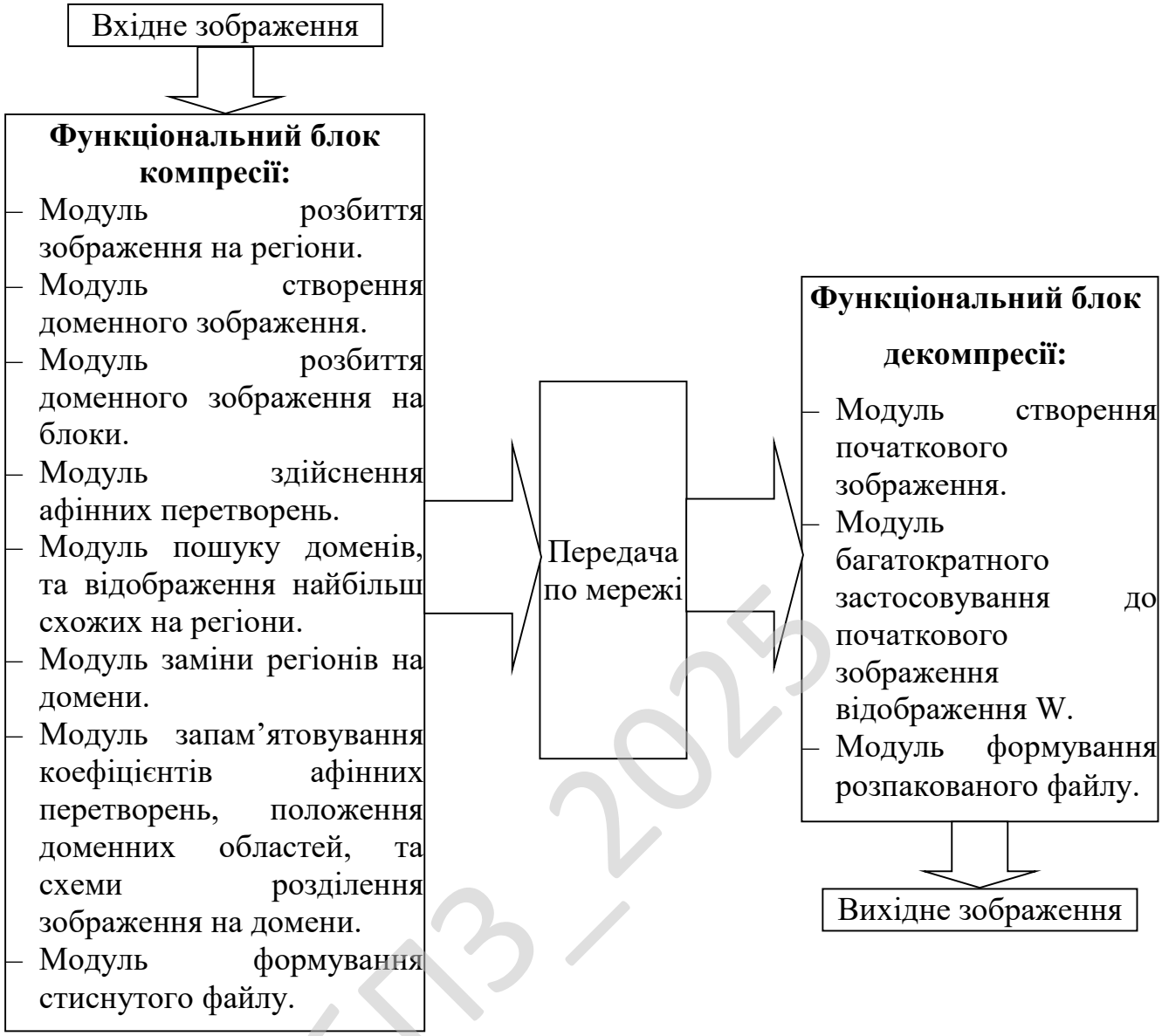

Рисунок 3.2 – Функціональна схема системи

Зображення без палітри бувають у якій-небудь системі кольоропредставлення й у градаціях сірого (grayscale). Для останніх значення кожного пікселя інтерпретується як яскравість відповідної точки. Зустрічаються зображення з 2, 16 і 256 рівнями сірого. Одне із цікавих практичних завдань полягає в приведенні кольорового або чорно-білого зображення до двох градацій яскравості, наприклад, для печатки на лазерному принтері. При використанні якоїсь системи кольоропредставлення кожний піксель являє собою запис (структуру), полями якої є компоненти кольору. Найпоширенішої є система RGB,

у якій колір представлений значеннями інтенсивності червоної (R), зеленої (G) і синьої (B) компонент. Існують і інші системи кольоропредставлення, такі, як CMYK, CIE XYZccir 60-1 і т.п. Нижче ми побачимо, як використовуються колірні моделі при стиску зображень із втратами.

Для того, щоб коректніше оцінювати ступінь стиску, потрібно ввести поняття класу зображень. Під класом буде розумітися якась сукупність зображень, застосування до яких алгоритму архівування дає якісно однакові результати. Наприклад, для одного класу алгоритм дає дуже високий ступінь стиску, для іншого – майже не стискає, для третього – збільшує файл у розмірі. (Відомо, що багато алгоритмів у найгіршому разі збільшують файл.)

Розглянемо наступні приклади неформального визначення класів зображень:

1. Клас 1. Зображення з невеликою кількістю кольорів (4-16) і більшими областями, заповненими одним кольором. Плавні переходи кольорів відсутні. Приклади: ділова графіка – гістограми, діаграми, графіки й т.п.

2. Клас 2. Зображення, із плавними переходами кольорів, побудовані на комп'ютері. Приклади: графіка презентацій, ескізні моделі в САПР, зображення, побудовані по методу Гуро.

3. Клас 3. Фотореалістичні зображення. Приклад: відсканованні фотографії.

4. Клас 4. Фотореалістичні зображення з накладенням ділової графіки. Приклад: реклама.

Розвиваючи дану класифікацію, як окремі класи можуть бути запропоновані неякісно відскановані в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт. (Помітимо, що цей клас не тотожний класу 4). Формально будучи 8– або 24-бітними, вони несуть навіть не растрову, а чисто векторну інформацію. Окремі класи можуть утворювати й зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і фас із електронного досьє. Досить складним і цікавим завданням є пошук найкращого алгоритму для конкретного класу зображень.

Нема рації говорити про те, що якийсь алгоритм стиску краще іншого, якщо ми не позначили класи зображень, на яких рівняються наші алгоритми.

Класи додатків

Розглянемо наступну просту класифікацію додатків, що використовують алгоритми компресії:

1. Клас 1. Характеризуються високими вимогами вчасно архівування й розархівування. Нерідко потрібен перегляд зменшеної копії зображення й пошук у базі даних зображень. Приклади: Видавничі системи в широкому змісті цього слова. Причому якісні публікації, що як готовлять (журнали) зі свідомо високою якістю зображень і використанням алгоритмів архівування без втрат, так і газети, що готовлять, і інформаційні вузли в WWW, де є можливість оперувати зображеннями меншої якості й використовувати алгоритми стиску із втратами. У подібних системах доводиться мати справа з повнокольоровими зображеннями самого різного розміру (від 640x480 – формат цифрового фотоапарата, до 3000x2000) і з великим двоцвітними зображеннями. Оскільки ілюстрації займають левову частину від загального обсягу матеріалу в документі, проблема зберігання коштує дуже гостро. Проблеми також створює більша різнорідність ілюстрацій (доводиться використовувати універсальні алгоритми). Єдине, що можна сказати заздалегідь, це те, що будуть переважати фотореалістичні зображення й ділова графіка.

2. Клас 2. Характеризується високими вимогами до ступеня архівування й часу розархівування. Час архівування ролі не грає. Іноді подібні додатки також жадають від алгоритму компресії легкості масштабування зображення під конкретний дозвіл монітора в користувача. Приклад: Довідники й енциклопедії на CD-ROM. При створенні енциклопедій і ігор більшу частину диска займають статичні зображення й відео. Таким чином, для цього класу додатків актуальність здобувають істотно асиметричні за часом алгоритми (симетричність за часом – відношення часу компресії вчасно декомпресії).

3. Клас 3. Характеризується дуже високими вимогами до ступеня архівування. Додаток клієнта одержує від сервера інформацію з мережі. Приклад: “Всесвітня інформаційна павутина” – WWW. У цій гіпертекстовій системі досить активно використовуються ілюстрації. При оформленні інформаційних або рекламних сторінок хочеться зробити їх більше яскравими й барвистими, що природно позначається на розмірі зображень. Найбільше при цьому страждають користувачі, підключені до мережі за допомогою повільних каналів зв'язку. Якщо сторінка WWW перенасичена графікою, то очікування її повної появи на екрані може затягтися. Оскільки при цьому навантаження на процесор мале, то тут можуть знайти застосування ефективно стискаючі складні алгоритми з порівняно більшим часом розархівування. Крім того, ми можемо видозмінити алгоритм і формат даних так, щоб переглядати огрублене зображення файлу до його повного одержання. Можна привести безліч більше вузьких класів додатків. Так, своє застосування машинна графіка знаходить і в різних інформаційних системах. Наприклад, уже стає звичним досліджувати ультразвукові й рентгенівські знімки не на папері, а на екрані монітора. Поступово в електронний вид переводять і історії хвороб. Зрозуміло, що зберігати ці матеріали логічніше в єдиній картотеці. При цьому без використання спеціальних алгоритмів більшу частину архівів займуть фотографії. Тому при створенні ефективних алгоритмів рішення цього завдання потрібно врахувати специфіку рентгенівських знімків – перевагу розмитих ділянок. Це неминуче накладає свої обмеження на алгоритм компресії. В електронних картотеках і досьє різних служб для зображень характерна подоба між фотографіями в профіль, і подоба між фотографіями у фас, що також необхідно враховувати при створенні алгоритму архівування. Подоба між фотографіями спостерігається й у будь-яких інших спеціалізованих довідниках. Як приклад можна привести енциклопедії птахів або квітів. Нема рації говорити про те, що якийсь конкретний алгоритм компресії краще іншого, якщо ми не позначили клас додатків, для якого ми ці алгоритми збираємося порівнювати.

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

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.

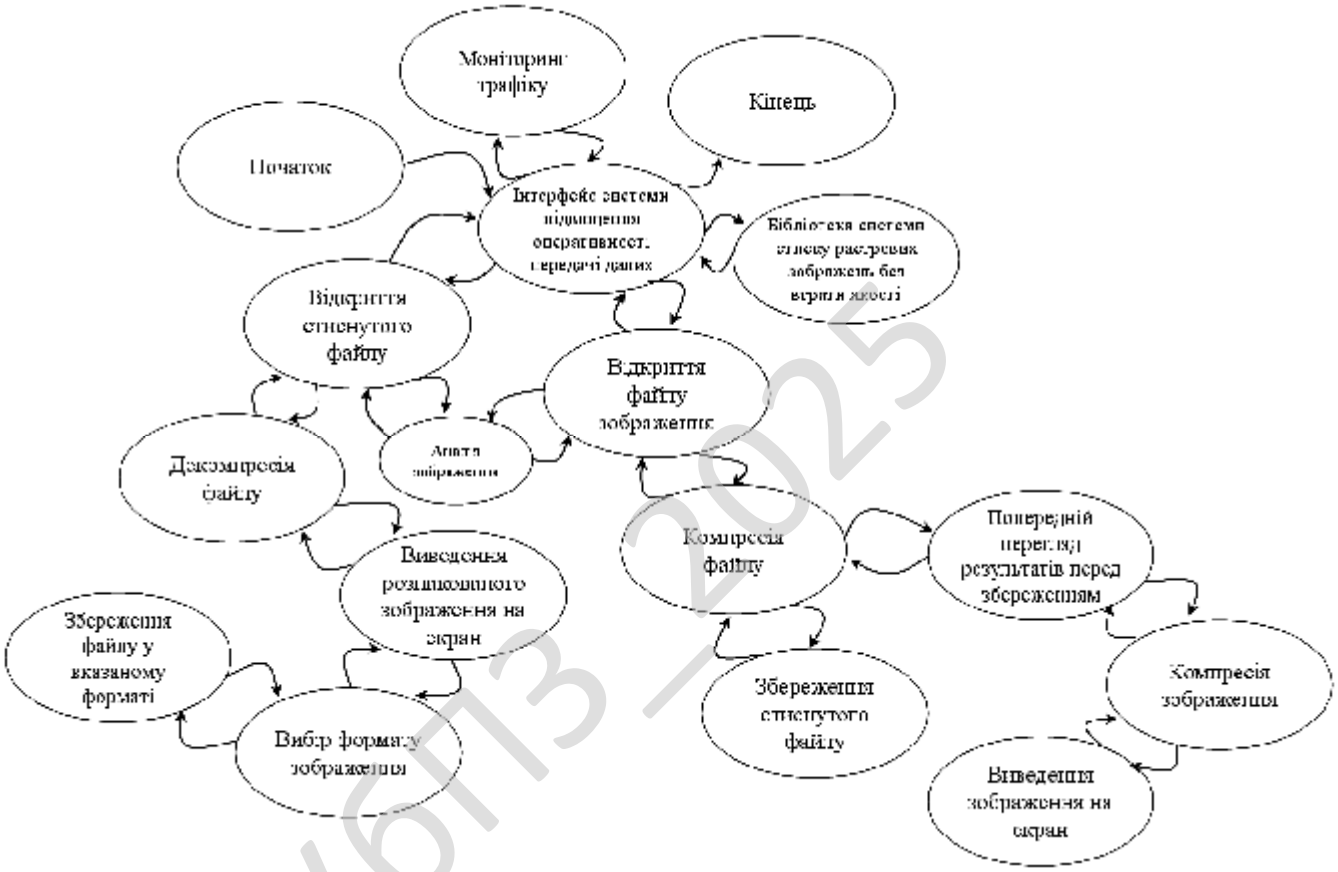

Рисунок 3.3 – Діаграма взаємодії процесів

Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає

уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

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

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Під час роботи над бакалаврською дипломною роботою було створено блок-схеми. Перед їх розглядом необхідно провести роз’яснення який саме тип блок-схем використовується.

Блок-схема це представлення задачі для її аналізу або розв'язування за допомогою спеціальних символів (геометричних образів), які позначають такі елементи, як операції, потік, дані тощо. Блок вхідних та вихідних даних прийнято позначати паралелограмом, блок обчислень (обробки) даних – прямокутником, блок прийняття рішень – ромбом, еліпсом – початок та кінець алгоритму.

У інформаційних технологіях функціональна схема складається з функціональних блоків, які являють собою конструктивно відособлені частини (елементи або пристрої) автоматичних систем, які виконують певні функції. Функціональні блоки на схемі позначають прямокутниками, всередині яких надписують їх найменування відповідно до функцій, що виконуються. Зв'язки між функціональними блоками (внутрішні впливи) позначаються лініями зі стрілками, які вказують напрям впливів.

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

У другому варіанті схема відображається більш детально, що полегшує її читання та ілюструє принцип роботи.

Основні елементи схем алгоритму це термінатор, процес, рішення, зумовлений процес (підпрограма), дані та з'єднувач.

Термінатор це елемент відображає вхід із зовнішнього середовища або вихід з неї (найчастіше застосування – початок і кінець програми). Всередині фігури записується відповідна дія.

Процес це виконання однієї або кількох операцій, обробка даних будь-якого виду (зміна значення даних, форми подання, розташування). Всередині фігури записують безпосередньо самі операції.

Рішення це показує рішення або функцію перемикального типу з одним входом і двома або більше альтернативними виходами, з яких тільки один може бути обраний після обчислення умов, визначених всередині цього елементу. Вхід в елемент позначається лінією, що входить зазвичай у верхню вершину елементу. Якщо виходів два чи три то зазвичай кожен вихід позначається лінією, що виходить з решти вершин (бічних і нижній). Якщо виходів більше трьох, то їх слід показувати однією лінією, що виходить з вершини (частіше нижній) елемента, яка потім розгалужується. Відповідні результати обчислень можуть записуватися поруч з лініями, що відображають ці шляхи.

Зумовлений процес (підпрограма) це символ відображає виконання процесу, що складається з однієї або кількох операцій, що визначені в іншому місці програми (у підпрограмі, модулі). Всередині символу записується назва процесу і передані в нього дані.

Дані це перетворення у форму, придатну для обробки (введення) або відображення результатів обробки (виведення). Цей символ не визначає носія даних (для вказівки типу носія даних використовуються специфічні символи).

З'єднувач це символ відображає вихід в частину схеми і вхід з іншої частини цієї схеми. Використовується для обриву лінії та продовження її в іншому місці (приклад: поділ блок-схеми, що не поміщається на листі). Відповідні сполучні символи повинні мати одне (при тому унікальне) позначення.

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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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

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

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

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.

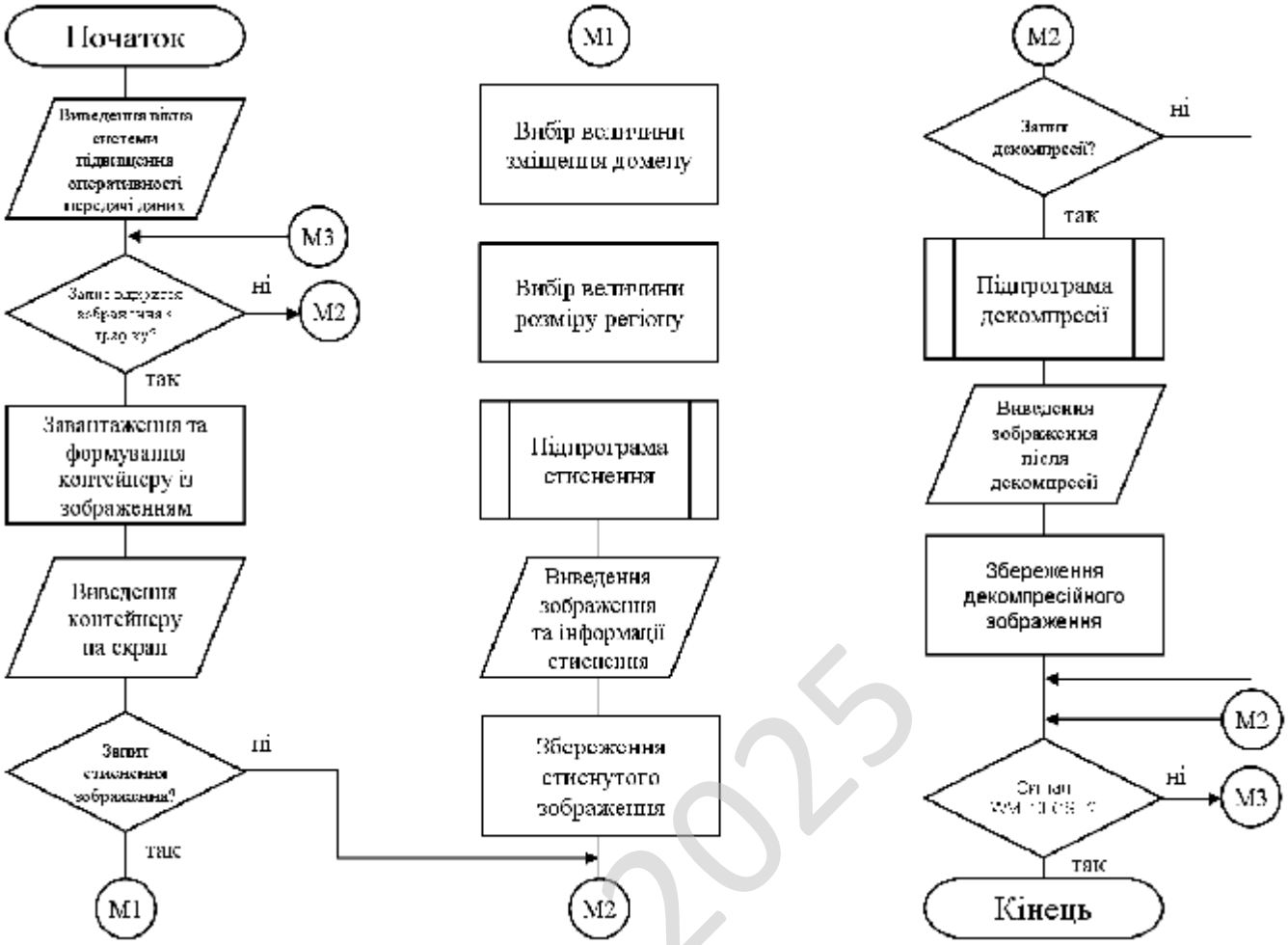

Рисунок 4.1 – Блок-схема основної програми

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.

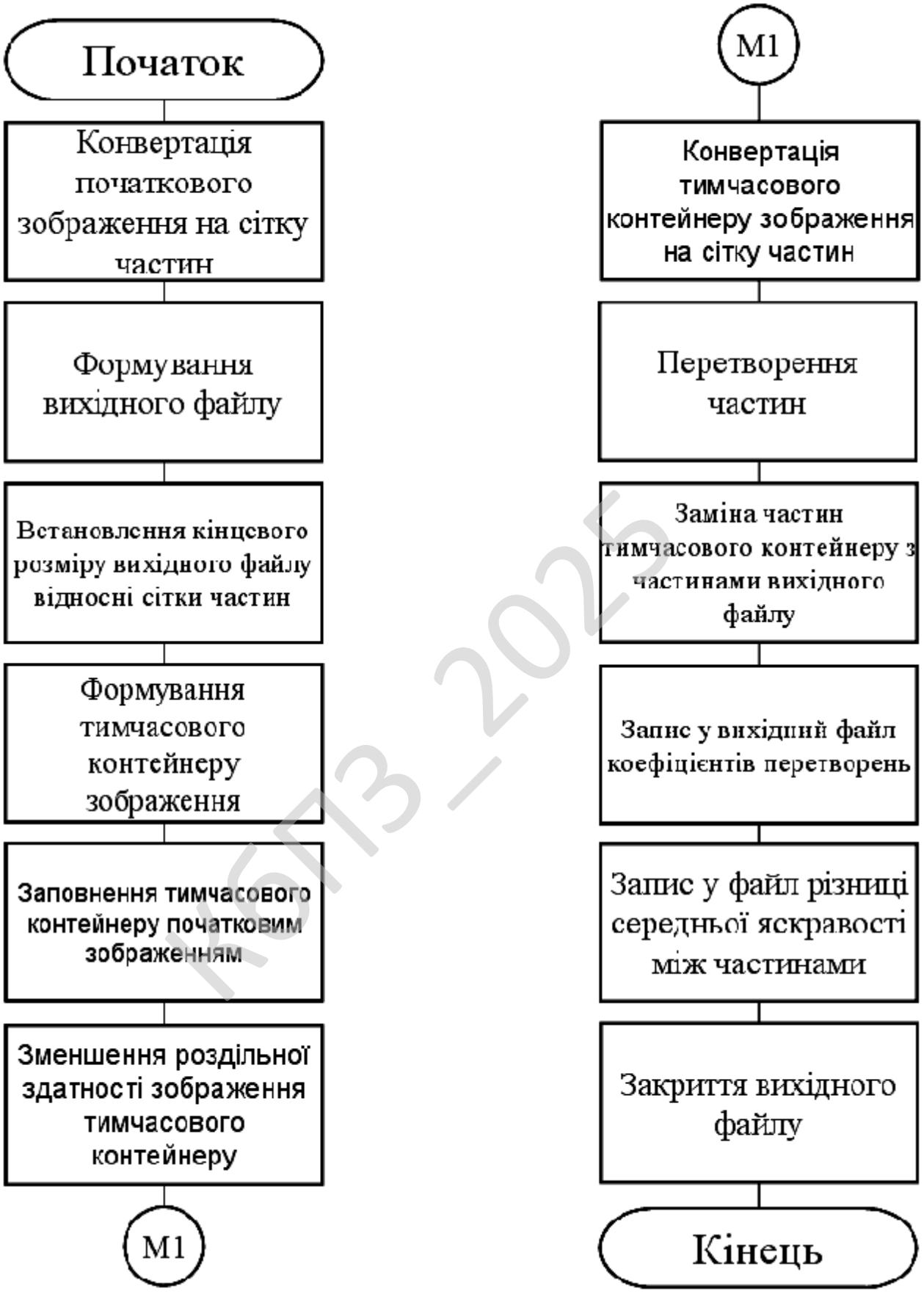

Рисунок 4.2 – Блок-схема роботи підпрограми

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.

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.

Також при розробці бакалаврської дипломної роботи було використано наступні підходи UML: діаграма діяльності (діаграми поведінки типу); діаграма прецедентів (діаграми поведінки типу); Діаграма класів.

Діаграма діяльності. Це візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Дія є фундаментальною одиницею визначення поведінки в специфікації. Дія отримує множину вхідних сигналів, та перетворює їх на множину вихідних сигналів.

Одна із цих множин, або обидві водночас, можуть бути порожніми. Виконання дії відповідає виконанню окремої дії. Подібно до цього, виконання

діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності.

Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Діаграма прецедентів це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також, перекладається як діаграма варіантів використання.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених границею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть даної діаграми полягає в наступному: проектована система представляється у вигляді безлічі сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система при діалозі з актором.

При цьому нічого не говориться про те, яким чином буде реалізована взаємодія акторів із системою.

У мові UML є кілька стандартних видів відношень між акторами і варіантами використання:

– асоціації (association relationship);

– включення (include relationship);

– розширення (extend relationship);

– узагальнення (generalization relationship).

При цьому загальні властивості варіантів використання можуть бути представлені трьома різними способами, а саме – за допомогою відношень включення, розширення і узагальнення.

Відношення асоціації – одне з фундаментальних понять у мові UML і в тій чи іншій мірі використовується при побудові всіх графічних моделей систем у формі канонічних діаграм.

Включення (include) у мові UML – це різновид відношення залежності між базовим варіантом використання і його спеціальним випадком. При цьому відношенням залежності (dependency) є таке відношення між двома елементами моделі, при якому зміна одного елемента (незалежного) приводить до зміни іншого елемента (залежного).

Відношення розширення (extend) визначає взаємозв'язок базового варіанта використання з іншим варіантом використання, функціональна поведінка якого задіюється базовим не завжди, а тільки при виконанні додаткових умов.

Діаграма класів це статичне представлення структури моделі. Відображає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення.

Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак їх динаміка розкривається в інших типах діаграм.

Діаграма класів (class diagram) служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування. На цій діаграмі показують класи, інтерфейси, об'єкти й кооперації, а також їхні відносини.

В UML існують наступні типи зв'язків які використовуються у діаграмі класів: Асоціації; Агрегація; Композиція.

Асоціації це якщо між двома класами визначена асоціація, то можна переміщатися від об'єктів одного класу до об'єктів іншого. Цілком припустимі випадки, коли обидва кінці асоціації відносяться до одного і того ж класу. Це означає, що з об'єктом деякого класу дозволено зв'язати інші об'єкти з того ж класу. Асоціація, що зв'язує два класи, називається бінарної. Можна, хоча це рідко буває необхідним, створювати асоціації, що зв'язують відразу кілька класів. Графічно асоціація зображується у вигляді лінії, що з'єднує клас сам з собою або з іншими класами.

Асоціації може бути присвоєно ім'я, яке описує природу відносини. Зазвичай ім'я асоціації не вказується, якщо тільки ви не хочете явно задати для неї рольові імена або у вашій моделі настільки багато асоціацій, що виникає необхідність посилатися на них і відрізняти один від одного. Ім'я буде особливо корисним, якщо між одними і тими ж класами існує кілька різних асоціацій.

Клас, що бере участь в асоціації, грає в ній деяку роль. По суті, це "обличчя", яким клас, що знаходиться на одній стороні асоціації, звернений до класу з іншого її боку. Можна явно позначити роль, яку клас грає в асоціації.

Часто при моделюванні буває важливо вказати, скільки об'єктів може бути пов'язано допомогою одного примірника асоціації. Це число називається кратністю (Multiplicity) ролі асоціації та записується або як вираз, значенням якого є діапазон значень, або в явному вигляді.

Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці. Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *). Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на

одному концептуальному рівні і ні один не є більш важливим, ніж інший. Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого». Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

Композиція це більш суворий варіант агрегації. Відома також як агрегація за значенням.

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Система підвищує оперативність передачі даних шляхом зменшення трафіку графічної інформації за рахунок оптимізації зображень перед їх передачею через мережу система використовує мову програмування Python що дозволяє ефективно працювати з бібліотеками для обробки зображень та мережевою взаємодією система містить модуль обробки зображень.

Який зчитує зображення з файлової системи та виконує стискання графічного контенту шляхом зменшення якості зображення та зміни формату файлу модуль мережевої передачі організовує встановлення з’єднання із сервером та передачу оптимізованих даних через протокол TCP.

Що забезпечує надійність зв’язку система має модуль розрахунків який аналізує вихідний обсяг даних порівнює його з розміром стисненого файлу та визначає відсоток економії трафіку.

Розрахунки базуються на співвідношенні первинного розміру файлу до розміру після стиснення що дозволяє підтвердити правильність проектних рішень

Як приклад розрахунків показує що зображення розміром 200 кілобайт стискається до 50 кілобайт що дає можливість зменшити трафік на 75 відсотків архітектура системи організована як модульна платформа де кожен компонент відповідає за окрему функціональність та взаємодіє із іншими блоками через чітко визначені інтерфейси що сприяє гнучкості та масштабованості розробка коду ведеться з використанням сучасних підходів оптимізації обчислювальних ресурсів.

Це дозволяє скоротити час передачі графічної інформації код організовується у вигляді функцій які виконують операції зчитування зображення обчислення розмірів файлів стискання зображення та передачі даних через мережу система забезпечує обробку помилок та ведення журналу подій що сприяє оперативному реагуванню на можливі збої вибір даного підходу підтверджується розрахунками та експериментальними даними які демонструють значне скорочення обсягу переданої інформації при збереженні необхідної якості графічного контенту.

Використання Python дозволяє швидко реалізувати необхідну функціональність та інтегрувати додаткові бібліотеки для розширення можливостей системи

```
import os
import socket
from PIL import Image
from io import BytesIO

# функція для зчитування зображення
def read_image(file_path):
    # читає зображення з файлової системи
    try:
        image = Image.open(file_path)
        return image
    except Exception as e:
```

```
            print("Помилка при зчитуванні зображення", e)
            return None

    # функція для стискання зображення
    def compress_image(image, quality=50):
        # виконує стискання зображення з використанням заданої якості
        buffer = BytesIO()
        image.save(buffer, format="JPEG", quality=quality)
        compressed_image_data = buffer.getvalue()
        return compressed_image_data

    # функція для передачі зображення через мережу
    def send_image_data(image_data, host="127.0.0.1", port=5000):
        # відкриває з'єднання з сервером та передає зображення
        try:
            sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            sock.connect((host, port))
            sock.sendall(image_data)
            sock.close()
        except Exception as e:
            print("Помилка при передачі зображення", e)

    # функція для розрахунку зменшення трафіку
    def calculate_savings(original_size, compressed_size):
        # розраховує відсоток зменшення обсягу даних
        if original_size == 0:
            return 0
        reduction = (original_size - compressed_size) / original_size * 100
        return reduction

    # головна функція системи
    def main():
        # визначає шлях до вхідного зображення
        file_path = "example.jpg"
        image = read_image(file_path)
        if image is not None:
            # отримує первинний обсяг даних зображення
            original_size = os.path.getsize(file_path)
            # стискає зображення з заданою якістю
            compressed_image_data = compress_image(image, quality=50)
            # отримує обсяг стиснених даних
            compressed_size = len(compressed_image_data)
```

```
        # розраховує відсоток зменшення трафіку
        savings = calculate_savings(original_size, compressed_size)
        print("Початковий обсяг", original_size, "байт")
        print("Стиснений обсяг", compressed_size, "байт")
        print("Зменшення трафіку", round(savings, 2), "відсотків")
        # передає стиснене зображення через мережу
        send_image_data(compressed_image_data)
    else:
        print("Не вдалося зчитати зображення")
```

4.2 Захист розробленого програмного забезпечення

Дані які використовуються у даній роботі захищаються алгоритмом ДСТУ 9041:2020. Його повна назва: ДСТУ 9041:2020. Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

Цей алгоритм призначений для шифрування коротких (до 616 біт) повідомлень для будь-яких алгоритмів шифрування, в тому числі визначених національними стандартами України.

Як і стандарт цифрового підпису ДСТУ 4145:2002, новий алгоритм використовує криптографічні перетворення у групі точок еліптичних кривих, використовуючи замість кривих у формі Вейєрштрасса найновітніші розробки у галузі еліптичної криптографії – криві у формі Едвардса. Це дає суттєві переваги у швидкодії більш ніж у 3 рази.Новий стандарт розроблений з урахуванням усіх найсучасніших вимог до стійкості криптографічних алгоритмів. Так, нижня межа стійкості криптосистем у цьому стандарті дорівнює 2127 (≈1042) (це більш ніж у півтора рази вище, ніж у ДСТУ 4145) і можуть бути обрані інші рівні, такі як 2255 (≈1085), 2383 (≈10127) та 2767 (≈10255); крім того, строго обґрунтована його стійкість як до атак на відновлення відкритого тексту, так і до розрізнюючих атак.

Проект алгоритму шифрування, що ліг в основу цього стандарту, пройшов апробацію як в Україні, так і за її межами (Центрально-Європейська конференція з криптографії (червень 2020 року) – форум ведучих криптологів з усього світу).

Стандарт ДСТУ-9041 узгоджений з усіма діючими в Україні національними стандартами. Новиною стандарту є його сфера застосування – інкапсуляція ключів, найсучасніший математичний апарат, а також новий алгоритм генерації псевдовипадкових послідовностей, який, на відміну від аналогічного алгоритму генерації з ДСТУ 4145, використовує виключно національні криптографічні алгоритми національних стандартів та не містить посилань на відповідні пост-радянські стандарти, термін дії яких вже практично вичерпався.

Новий стандарт не належить до так званих постквантових стандартів. Але його стійкість буде під загрозою лише тоді, коли з'являться квантові комп'ютери з 700 і більше кубітами (на даний час кількість "робочих" кубітів, які вдалося створити, – близько 50). Його перевагою перед постквантовими алгоритмами є відносно невелика довжина ключа (у десятки або навіть у сотні разів менша, ніж у постквантових алгоритмах).

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської дипломної роботи.

Розроблене програмне забезпечення підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації складається з наступних функціональних блоків:

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

– Блоку виведення дерево папок.

– Блоку відображення обраної папки.

– Блоку відображення робочих функції та функціональних кнопок ПЗ.

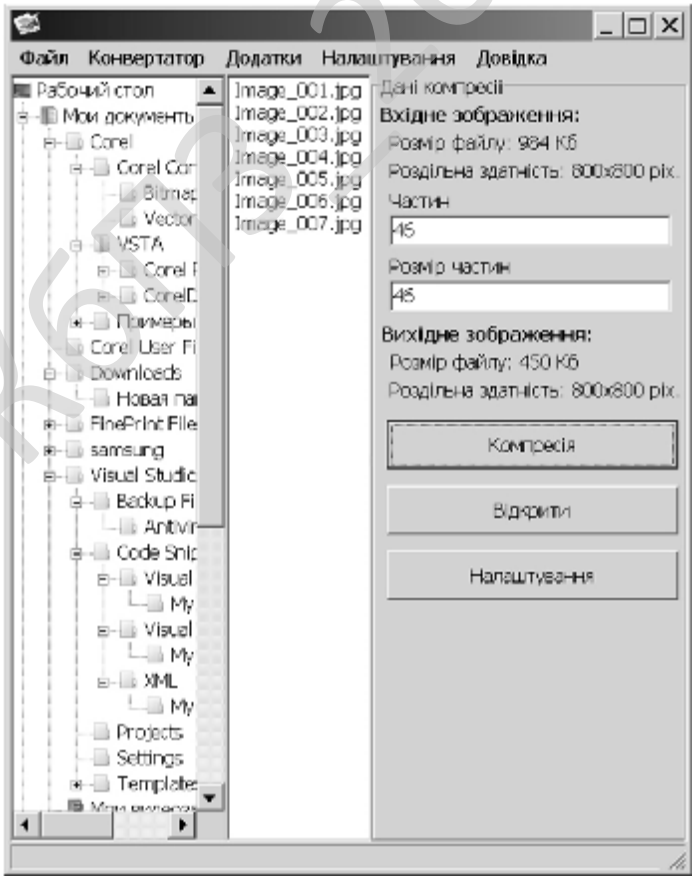

Рисунок 5.1 – Головне вікно розробленого ПЗ

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.

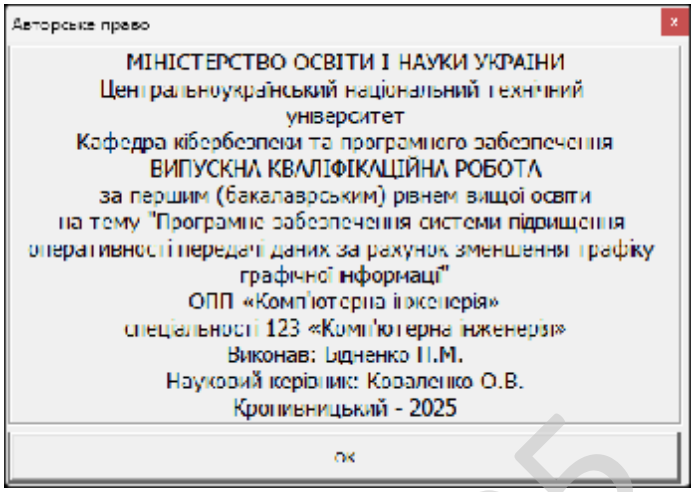

Рисунок 5.2 – Вікно розробника ПЗ

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 білої скриньки засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреною, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління.

У цьому випадку формуються тестові варіанти, в яких:

– Гарантується перевірка всіх незалежних маршрутів програми.

– Знаходяться гілки True, False для всіх логічних рішень.

– Виконуються всі цикли (у межах їхніх кордонів та діапазонів).

– Аналізується правильність внутрішніх структур даних.

Недоліки тестування "білої скриньки":

– Кількість незалежних маршрутів може бути дуже велика.

– Повне тестування маршрутів не гарантує відповідності програми вихідним вимогам до неї.

– У програмі можуть бути пропущені деякі маршрути.

– Не можна виявити помилки, поява яких залежить від даних.

Переваги тестування "білої скриньки" пов'язані з тим, що принцип «білої скриньки» дозволяє врахувати особливості програмних помилок:

– Кількість помилок мінімально в «центрі» і максимально на «периферії» програми.

– Попередні припущення про ймовірність потоку керування або даних у програмі часто бувають некоректними. У результаті типовим може стати маршрут, модель обчислень за яким опрацьована слабо.

– При записі алгоритму програмного забезпечення у вигляді тексту на мові програмування можливе внесення типових помилок трансляції (синтаксичних та семантичних).

– Деякі результати в програмі залежать не від вихідних даних, а від внутрішніх станів програми.

Обрано умови розповсюдження – 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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

– Досліджена система підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

– На основі отриманих результатів досліджень створена програмна реалізація системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації. 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 ДСТУ 9041:2020.

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. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

2. Peter Shirley, Steve Marschner. Fundamentals of Computer Graphics. 2009

3. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

4. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

5. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

6. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

7. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

8. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

9. Kuznetsov, O., Frontoni, E., Kryvinska, N., Chevardin, V., Smirnov, O. «Wireless Network Encryption Stream Ciphers, Computational Modeling, and Security Analysis». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 379–402.

10. Kuznetsov, O., Frontoni, E., Kryvinska, N., Smirnov, O., Imoize, G.L. «Computational Modeling of Enhanced Spread Spectrum Codes for Asynchronous Wireless Communication». *Computational Modeling and Simulation of Advanced Wireless Communication Systems*, 2024, pp. 403–447

11. 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.

12. 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.

13. 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.

14. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

15. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

16. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

17. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

18. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

19. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

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

21. 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.

22. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

23. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

24. 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.

25. 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.

26. 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.

27. 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.

28. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

29. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

30. 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.

31. 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.

32. 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.

33. 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.

34. 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.

35. 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.

36. 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.

37. Smirnov A.A., Kuznetsov 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.

38. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

39. 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.

40. 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.

41. 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.

42. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

43. 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.

44. 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.

45. 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.

46. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

47. 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.

48. 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).

49. 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

50. 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.

51. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

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

Додаток А
(обов’язковий)

Технічне завдання

Зміст

					ВКРБ-123.25.0003.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Бідненко П.М.			Програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Коваленко О.В.				Б	
Н. Контр.		Коваленко А.С.				ЦНТУ КІ-21-1	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації.

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 підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації;

– 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 – 73 аркуші.

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

_____ Коваленко О.В.

Програмне забезпечення системи підвищення оперативності передачі даних за рахунок зменшення трафіку графічної інформації

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 21

Літера: РП

Кропивницький – 2025 року

Основна програма

```
# Імпорт необхідних модулів для роботи з графікою, стисненням та мережею
import socket
import threading
import os
import sys
import time
import zlib
import cv2
import numpy as np
from PIL import Image
from io import BytesIO
import struct

# Оголошення глобальних констант та налаштувань
SERVER_IP = '127.0.0.1'
SERVER_PORT = 5005
BUFFER_SIZE = 4096
JPEG_QUALITY = 50
COMPRESSION_LEVEL = 6
MAX_PACKET_SIZE = 1024

# Ініціалізація базових параметрів програми
image_folder_path = './images'
received_folder_path = './received'

# Перевірка наявності директорій та створення, якщо відсутні
if not os.path.exists(image_folder_path):
    os.makedirs(image_folder_path)

if not os.path.exists(received_folder_path):
    os.makedirs(received_folder_path)

# Оголошення функції для зчитування зображень із директорії
def load_images_from_folder(folder_path):
    images_list = []
    filenames_list = []
    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)
        if os.path.isfile(file_path):
            try:
                image = cv2.imread(file_path)
                if image is not None:
                    images_list.append(image)
                    filenames_list.append(filename)
            except Exception as e:
                print(f"Error loading {file_path}: {e}")
    return images_list, filenames_list

# Оголошення функції для перетворення зображення у формат JPEG із заданою якістю
def convert_image_to_jpeg(image, quality):
    encode_param = [int(cv2.IMWRITE_JPEG_QUALITY), quality]
    result, encimg = cv2.imencode('.jpg', image, encode_param)
    if result:
        return encimg.tobytes()
    else:
        return None

# Оголошення функції для стискання даних з використанням алгоритму zlib
def compress_data_zlib(data, compression_level):
    compressed_data = zlib.compress(data, level=compression_level)
    return compressed_data
```

```
# Оголошення функції для розпакування стислих даних
def decompress_data_zlib(data):
    decompressed_data = zlib.decompress(data)
    return decompressed_data

# Оголошення функції для відправки файлу по мережі
def send_file_over_network(file_data, connection_socket):
    total_size = len(file_data)
    connection_socket.send(struct.pack('>Q', total_size))
    sent_bytes = 0
    while sent_bytes < total_size:
        end_index = min(sent_bytes + MAX_PACKET_SIZE, total_size)
        chunk = file_data[sent_bytes:end_index]
        connection_socket.sendall(chunk)
        sent_bytes = end_index
        time.sleep(0.001)

# Оголошення функції для приймання файлу по мережі
def receive_file_over_network(connection_socket):
    packed_size = connection_socket.recv(8)
    if not packed_size:
        return None
    total_size = struct.unpack('>Q', packed_size)[0]
    received_data = bytearray()
    while len(received_data) < total_size:
        packet = connection_socket.recv(MAX_PACKET_SIZE)
        if not packet:
            break
        received_data.extend(packet)
    return bytes(received_data)

# Оголошення функції-сервера для приймання зображень
def start_server(ip, port):
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((ip, port))
    server_socket.listen(5)
    print(f"Server listening on {ip}:{port}")
    while True:
        conn, addr = server_socket.accept()
        print(f"Connection accepted from {addr}")
        threading.Thread(target=handle_client, args=(conn,)).start()

# Оголошення функції для обробки клієнтського з'єднання на сервері
def handle_client(client_socket):
    try:
        received_compressed_data = receive_file_over_network(client_socket)
        if received_compressed_data:
            decompressed_data = decompress_data_zlib(received_compressed_data)
            nparr = np.frombuffer(decompressed_data, np.uint8)
            image = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
            if image is not None:
                timestamp = int(time.time() * 1000)
                save_path = os.path.join(received_folder_path,
f"received_{timestamp}.jpg")
                cv2.imwrite(save_path, image)
                print(f"Image saved to {save_path}")
            else:
                print("Failed to decode image")
        else:
            print("No data received")
    except Exception as e:
        print(f"Error handling client: {e}")
```

```
    finally:
        client_socket.close()

# Оголошення функції-клієнта для відправки зображень
def start_client(ip, port, images_list, filenames_list):
    for idx, image in enumerate(images_list):
        try:
            print(f"Processing image {filenames_list[idx]}")
            jpeg_data = convert_image_to_jpeg(image, JPEG_QUALITY)
            if jpeg_data is None:
                print(f"Failed to convert image {filenames_list[idx]} to JPEG")
                continue
            compressed_data = compress_data_zlib(jpeg_data, COMPRESSION_LEVEL)
            client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            client_socket.connect((ip, port))
            send_file_over_network(compressed_data, client_socket)
            print(f"Sent image {filenames_list[idx]} successfully")
            client_socket.close()
            time.sleep(1)
        except Exception as e:
            print(f"Error sending image {filenames_list[idx]}: {e}")

# Оголошення функції для тестування стиснення на прикладі одного зображення
def test_compression(images_list):
    if not images_list:
        print("No images to test compression")
        return
    sample_image = images_list[0]
    print("Testing compression on sample image")
    jpeg_data = convert_image_to_jpeg(sample_image, JPEG_QUALITY)
    if jpeg_data is None:
        print("JPEG conversion failed")
        return
    compressed_data = compress_data_zlib(jpeg_data, COMPRESSION_LEVEL)
    decompressed_data = decompress_data_zlib(compressed_data)
    nparr = np.frombuffer(decompressed_data, np.uint8)
    image = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    if image is not None:
        cv2.imshow("Decompressed Image", image)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
    else:
        print("Failed to decode decompressed data")

# Оголошення класу для управління системою передачі зображень
class ImageTransferSystem:
    def __init__(self, server_ip, server_port, folder_path):
        self.server_ip = server_ip
        self.server_port = server_port
        self.folder_path = folder_path
        self.images_list = []
        self.filenames_list = []

    def load_images(self):
        self.images_list, self.filenames_list =
load_images_from_folder(self.folder_path)
        print(f"Loaded {len(self.images_list)} images from {self.folder_path}")

    def send_images(self):
        if not self.images_list:
            print("No images loaded to send")
            return
```

```
        start_client(self.server_ip, self.server_port, self.images_list,
self.filenames_list)

    def run_server(self):
        start_server(self.server_ip, self.server_port)

# Оголошення головної функції програми
def main():
    mode = ''
    while mode not in ['1', '2']:
        print("Select mode:")
        print("1 - Server")
        print("2 - Client")
        mode = input("Enter choice (1/2): ")

    if mode == '1':
        system = ImageTransferSystem(SERVER_IP, SERVER_PORT,
received_folder_path)
        server_thread = threading.Thread(target=system.run_server)
        server_thread.start()
    elif mode == '2':
        system = ImageTransferSystem(SERVER_IP, SERVER_PORT, image_folder_path)
        system.load_images()
        system.send_images()

# Виклик головної функції при запуску скрипту
if __name__ == '__main__':
    main()
```

Файл decompress_data.py

```
import socket
import threading
import os
import sys
import time
import zlib
import cv2
import numpy as np
import struct
import tkinter as tk
import tkinter.filedialog
import hashlib
from flask import Flask, send_from_directory
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes

SERVER_IP = '127.0.0.1'
SERVER_PORT = 6000
BUFFER_SIZE = 4096
MAX_PACKET_SIZE = 1024
ENCRYPTION_KEY = b'Sixteen byte key'
RECEIVED_DIR = './received'
MONITOR_DIR = './monitor'

def pad(data):
    pad_len = AES.block_size - len(data) % AES.block_size
    return data + bytes([pad_len]) * pad_len

def unpad(data):
    pad_len = data[-1]
    return data[:-pad_len]

def encrypt_data(data, key):
    iv = get_random_bytes(AES.block_size)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    padded_data = pad(data)
    ciphertext = cipher.encrypt(padded_data)
    return iv + ciphertext

def decrypt_data(data, key):
    iv = data[:AES.block_size]
    ciphertext = data[AES.block_size:]
    cipher = AES.new(key, AES.MODE_CBC, iv)
    padded_data = cipher.decrypt(ciphertext)
    return unpad(padded_data)

def compress_data(data):
    return zlib.compress(data, level=6)

def decompress_data(data):
    return zlib.decompress(data)

def recvall(sock, n):
    data = b''
    while len(data) < n:
        packet = sock.recv(n - len(data))
        if not packet:
            return None
        data += packet
    return data
```

```
def send_data(sock, data):
    total_size = len(data)
    sock.sendall(struct.pack('>Q', total_size))
    sent = 0
    while sent < total_size:
        chunk = data[sent:sent+MAX_PACKET_SIZE]
        sock.sendall(chunk)
        sent += len(chunk)
        time.sleep(0.001)

def receive_data(sock):
    raw_size = recvall(sock, 8)
    if not raw_size:
        return None
    total_size = struct.unpack('>Q', raw_size)[0]
    data = recvall(sock, total_size)
    return data

def log_event(event):
    with open("server_log.txt", "a") as f:
        f.write(f"{time.ctime()}: {event}\n")

def md5_checksum(filepath):
    md5 = hashlib.md5()
    with open(filepath, 'rb') as f:
        for chunk in iter(lambda: f.read(4096), b""):
            md5.update(chunk)
    return md5.hexdigest()

def handle_client(client_sock):
    try:
        enc_data = receive_data(client_sock)
        if enc_data is None:
            client_sock.close()
            return
        dec_data = decrypt_data(enc_data, ENCRYPTION_KEY)
        raw_data = decompress_data(dec_data)
        nparr = np.frombuffer(raw_data, np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
        timestamp = int(time.time())
        filename = os.path.join(RECEIVED_DIR, f"img_{timestamp}.jpg")
        cv2.imwrite(filename, img)
        log_event(f"Saved file {filename}")
    except Exception as e:
        log_event(f"Error in handle_client: {e}")
    finally:
        client_sock.close()

def server_handler():
    if not os.path.exists(RECEIVED_DIR):
        os.makedirs(RECEIVED_DIR)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((SERVER_IP, SERVER_PORT))
    s.listen(5)
    while True:
        client, addr = s.accept()
        threading.Thread(target=handle_client, args=(client,)).start()

app = Flask(__name__)

@app.route('/')
def index():
    files = os.listdir(RECEIVED_DIR)
```

```
    files = [f for f in files if os.path.isfile(os.path.join(RECEIVED_DIR, f))]
    html = "<html><body><h1>Received Files</h1><ul>"
    for f in files:
        html += f"<li><a href='/files/{f}'>{f}</a></li>"
    html += "</ul></body></html>"
    return html

@app.route('/files/<filename>')
def serve_file(filename):
    return send_from_directory(RECEIVED_DIR, filename)

def run_web_interface():
    app.run(port=5000)

def send_image_file(filepath):
    img = cv2.imread(filepath)
    if img is None:
        return
    ret, buf = cv2.imencode('.jpg', img, [int(cv2.IMWRITE_JPEG_QUALITY), 50])
    if not ret:
        return
    raw_data = buf.tobytes()
    comp_data = compress_data(raw_data)
    enc_data = encrypt_data(comp_data, ENCRYPTION_KEY)
    client_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_sock.connect((SERVER_IP, SERVER_PORT))
    send_data(client_sock, enc_data)
    client_sock.close()

def send_video_file(filepath):
    cap = cv2.VideoCapture(filepath)
    frame_count = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        ret, buf = cv2.imencode('.jpg', frame, [int(cv2.IMWRITE_JPEG_QUALITY),
50])
        if not ret:
            continue
        raw_data = buf.tobytes()
        comp_data = compress_data(raw_data)
        enc_data = encrypt_data(comp_data, ENCRYPTION_KEY)
        client_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client_sock.connect((SERVER_IP, SERVER_PORT))
        send_data(client_sock, enc_data)
        client_sock.close()
        frame_count += 1
        time.sleep(0.05)
    cap.release()

def send_file(filepath):
    ext = os.path.splitext(filepath)[1].lower()
    if ext in ['.jpg', '.jpeg', '.png', '.bmp']:
        send_image_file(filepath)
    elif ext in ['.mp4', '.avi', '.mov', '.mkv']:
        send_video_file(filepath)

def monitor_directory(directory, sent_files):
    while True:
        files = os.listdir(directory)
        for file in files:
            full_path = os.path.join(directory, file)
```

```
            if full_path not in sent_files and os.path.isfile(full_path):
                send_file(full_path)
                sent_files.add(full_path)
        time.sleep(2)

def client_gui():
    root = tk.Tk()
    root.title("File Sender")
    def select_and_send():
        filepaths = tk.filedialog.askopenfilenames()
        for fp in filepaths:
            send_file(fp)
    btn = tk.Button(root, text="Select Files to Send", command=select_and_send)
    btn.pack()
    root.mainloop()

def main():
    if len(sys.argv) < 2:
        print("Usage: python script.py [server|client]")
        sys.exit(1)
    mode = sys.argv[1].lower()
    if mode == "server":
        if not os.path.exists(RECEIVED_DIR):
            os.makedirs(RECEIVED_DIR)
        server_thread = threading.Thread(target=server_handler, daemon=True)
        server_thread.start()
        web_thread = threading.Thread(target=run_web_interface, daemon=True)
        web_thread.start()
        while True:
            time.sleep(1)
    elif mode == "client":
        if not os.path.exists(MONITOR_DIR):
            os.makedirs(MONITOR_DIR)
        sent_files = set()
        monitor_thread = threading.Thread(target=monitor_directory,
args=(MONITOR_DIR, sent_files), daemon=True)
        monitor_thread.start()
        client_gui()
    else:
        print("Invalid mode. Use server or client.")

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

Файл group_images_by_resolution.py

```
import socket
import threading
import os
import time
import zlib
import bz2
import lzma
import cv2
import numpy as np
import hashlib
import struct

SERVER_IP = '127.0.0.1'
SERVER_PORT = 7000
DEFAULT_PACKET_SIZE = 1024
MAX_PACKET_SIZE = DEFAULT_PACKET_SIZE
USER_CREDENTIALS = {'user1': 'password1', 'admin': 'adminpass'}
BACKUP_DIR = './backup'
IMAGE_DIR = './images'

def compress_data(algorithm, data):
    if algorithm == 'zlib':
        return zlib.compress(data, level=6)
    elif algorithm == 'bz2':
        return bz2.compress(data)
    elif algorithm == 'lzma':
        return lzma.compress(data)
    else:
        return data

def decompress_data(algorithm, data):
    if algorithm == 'zlib':
        return zlib.decompress(data)
    elif algorithm == 'bz2':
        return bz2.decompress(data)
    elif algorithm == 'lzma':
        return lzma.decompress(data)
    else:
        return data

def calculate_checksum(data):
    return hashlib.sha256(data).digest()

def recvall(sock, n):
    data = b''
    while len(data) < n:
        packet = sock.recv(n - len(data))
        if not packet:
            return None
        data += packet
    return data

def send_data_with_integrity(sock, data):
    checksum = calculate_checksum(data)
    total_size = len(data)
    header = struct.pack('>Q', total_size) + checksum
    sock.sendall(header)
    sent = 0
    while sent < total_size:
        chunk = data[sent:sent+MAX_PACKET_SIZE]
        sock.sendall(chunk)
```

```
            sent += len(chunk)
            time.sleep(0.001)

    def receive_data_with_integrity(sock):
        header = recvall(sock, 40)
        if not header:
            return None
        total_size = struct.unpack('>Q', header[:8])[0]
        sent_checksum = header[8:40]
        data = recvall(sock, total_size)
        if not data:
            return None
        if calculate_checksum(data) != sent_checksum:
            return None
        return data

    def ping_server(server_ip, server_port):
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            start_time = time.time()
            s.connect((server_ip, server_port))
            s.sendall(b'PING')
            response = s.recv(1024)
            end_time = time.time()
            s.close()
            if response == b'PONG':
                return (end_time - start_time) * 1000
            else:
                return None
        except:
            return None

    def adjust_packet_size():
        global MAX_PACKET_SIZE
        rtt = ping_server(SERVER_IP, SERVER_PORT)
        if rtt is None:
            MAX_PACKET_SIZE = DEFAULT_PACKET_SIZE
        elif rtt < 50:
            MAX_PACKET_SIZE = 2048
        elif rtt < 150:
            MAX_PACKET_SIZE = 1024
        else:
            MAX_PACKET_SIZE = 512

    def group_images_by_resolution(directory):
        groups = {'small': [], 'medium': [], 'large': []}
        for file in os.listdir(directory):
            path = os.path.join(directory, file)
            if os.path.isfile(path):
                img = cv2.imread(path)
                if img is not None:
                    height, width = img.shape[:2]
                    if width < 640:
                        groups['small'].append(path)
                    elif width < 1280:
                        groups['medium'].append(path)
                    else:
                        groups['large'].append(path)
        return groups

    def backup_file(filepath):
        if not os.path.exists(BACKUP_DIR):
            os.makedirs(BACKUP_DIR)
```

```
    filename = os.path.basename(filepath)
    backup_path = os.path.join(BACKUP_DIR, filename)
    with open(filepath, 'rb') as f:
        data = f.read()
    with open(backup_path, 'wb') as f:
        f.write(data)

def authenticate_user(username, password):
    return USER_CREDENTIALS.get(username) == password

def server_authenticate(sock):
    creds_len_bytes = recvall(sock, 4)
    if not creds_len_bytes:
        return False
    creds_len = struct.unpack('>I', creds_len_bytes)[0]
    creds_data = recvall(sock, creds_len)
    if not creds_data:
        return False
    creds = creds_data.decode('utf-8').split(':')
    if len(creds) != 2:
        return False
    username = creds[0]
    password = creds[1]
    return authenticate_user(username, password)

def client_send_credentials(sock, username, password):
    creds = f"{username}:{password}".encode('utf-8')
    creds_len = len(creds)
    sock.sendall(struct.pack('>I', creds_len))
    sock.sendall(creds)

def handle_client(sock):
    if not server_authenticate(sock):
        sock.close()
        return
    data = receive_data_with_integrity(sock)
    if data is None:
        sock.close()
        return
    decompressed = decompress_data(selected_compression, data)
    nparr = np.frombuffer(decompressed, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    if img is not None:
        timestamp = int(time.time())
        filename = os.path.join(BACKUP_DIR, f"recv_{timestamp}.jpg")
        cv2.imwrite(filename, img)
        backup_file(filename)
    sock.sendall(b'PONG')
    sock.close()

def server_handler():
    if not os.path.exists(BACKUP_DIR):
        os.makedirs(BACKUP_DIR)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((SERVER_IP, SERVER_PORT))
    s.listen(5)
    while True:
        client, addr = s.accept()
        threading.Thread(target=handle_client, args=(client,)).start()

def client_send_file(filepath, username, password):
    adjust_packet_size()
    img = cv2.imread(filepath)
```

```
    if img is None:
        return
    ret, buf = cv2.imencode('.jpg', img, [int(cv2.IMWRITE_JPEG_QUALITY), 50])
    if not ret:
        return
    raw_data = buf.tobytes()
    comp_data = compress_data(selected_compression, raw_data)
    client_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_sock.connect((SERVER_IP, SERVER_PORT))
    client_send_credentials(client_sock, username, password)
    send_data_with_integrity(client_sock, comp_data)
    response = client_sock.recv(1024)
    client_sock.close()

def client_send_grouped_images(directory, username, password):
    groups = group_images_by_resolution(directory)
    for category in groups:
        for filepath in groups[category]:
            client_send_file(filepath, username, password)
            time.sleep(0.5)

def client_login():
    uname = input("Username: ")
    pwd = input("Password: ")
    if authenticate_user(uname, pwd):
        return uname, pwd
    else:
        print("Authentication failed")
        exit(1)

def main():
    import sys
    if len(sys.argv) < 2:
        print("Usage: python script.py [server|client]")
        sys.exit(1)
    mode = sys.argv[1].lower()
    global selected_compression
    selected_compression = 'zlib'
    if mode == "server":
        server_handler()
    elif mode == "client":
        username, password = client_login()
        client_send_grouped_images(IMAGE_DIR, username, password)
    else:
        print("Invalid mode")
        sys.exit(1)

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

Файл start_client.py

```
import socket
import threading
import time
import struct
import os
import cv2
import numpy as np
from flask import Flask, Response
import random

def fec_encode(data, block_size):
    blocks = [data[i:i+block_size] for i in range(0, len(data), block_size)]
    max_len = max(len(block) for block in blocks)
    padded_blocks = [block.ljust(max_len, b'\0') for block in blocks]
    parity = bytearray(max_len)
    for block in padded_blocks:
        for i in range(max_len):
            parity[i] ^= block[i]
    blocks.append(bytes(parity))
    return blocks

def fec_decode(blocks, block_size):
    if len(blocks) < 2:
        return b''.join(blocks)
    parity = blocks[-1]
    data_blocks = blocks[:-1]
    missing_index = None
    for i, block in enumerate(data_blocks):
        if block is None:
            missing_index = i
            break
    if missing_index is not None:
        recovered = bytearray(len(parity))
        for i, block in enumerate(data_blocks):
            if i != missing_index:
                for j in range(len(parity)):
                    recovered[j] ^= block[j]
        recovered = bytes([parity[j] ^ recovered[j] for j in
range(len(parity))])
        data_blocks[missing_index] = recovered
    data = b''.join(block.rstrip(b'\0') for block in data_blocks)
    return data

def send_with_retransmission(sock, data, timeout=1.0, retries=5):
    total_size = len(data)
    sock.sendall(struct.pack('>I', total_size))
    ack = sock.recv(2)
    if ack != b'OK':
        return False
    num_packets = (total_size + 1023) // 1024
    for i in range(num_packets):
        start = i * 1024
        end = min(start + 1024, total_size)
        packet = data[start:end]
        sent = False
        for attempt in range(retries):
            sock.sendall(struct.pack('>I', len(packet)))
            sock.sendall(packet)
            sock.settimeout(timeout)
            try:
                response = sock.recv(2)
```

```
                if response == b'AK':
                    sent = True
                    break
            except:
                continue
        if not sent:
            return False
    return True

def receive_with_retransmission(sock):
    total_size_bytes = sock.recv(4)
    if not total_size_bytes:
        return None
    total_size = struct.unpack('>I', total_size_bytes)[0]
    sock.sendall(b'OK')
    received = bytearray()
    while len(received) < total_size:
        packet_len_bytes = sock.recv(4)
        if not packet_len_bytes:
            return None
        packet_len = struct.unpack('>I', packet_len_bytes)[0]
        packet = sock.recv(packet_len)
        received.extend(packet)
        sock.sendall(b'AK')
    return bytes(received)

class MasterServer:
    def __init__(self, host, port, worker_ports):
        self.host = host
        self.port = port
        self.worker_ports = worker_ports
        self.current_worker = 0
    def start(self):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind((self.host, self.port))
        s.listen(5)
        while True:
            client, addr = s.accept()
            threading.Thread(target=self.handle_client, args=(client,)).start()
    def handle_client(self, client_sock):
        data = receive_with_retransmission(client_sock)
        if data is None:
            client_sock.close()
            return
        worker_port = self.worker_ports[self.current_worker]
        self.current_worker = (self.current_worker + 1) % len(self.worker_ports)
        worker_sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        worker_sock.connect((self.host, worker_port))
        send_with_retransmission(worker_sock, data)
        worker_sock.close()
        client_sock.sendall(b'REC')
        client_sock.close()

class WorkerServer:
    def __init__(self, host, port, storage_dir):
        self.host = host
        self.port = port
        self.storage_dir = storage_dir
        if not os.path.exists(self.storage_dir):
            os.makedirs(self.storage_dir)
    def start(self):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind((self.host, self.port))
```

```
        s.listen(5)
        while True:
            client, addr = s.accept()
            threading.Thread(target=self.handle_data, args=(client,)).start()
    def handle_data(self, sock):
        data = receive_with_retransmission(sock)
        if data is not None:
            filename = os.path.join(self.storage_dir,
f"worker_{self.port}_{int(time.time())}.dat")
            with open(filename, "wb") as f:
                f.write(data)
        sock.close()

app = Flask(__name__)
def gen_frames():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        ret, buffer = cv2.imencode('.jpg', frame)
        frame_bytes = buffer.tobytes()
        yield (b'--frame\r\nContent-Type: image/jpeg\r\n\r\n' + frame_bytes +
b'\r\n')
    cap.release()
@app.route('/stream')
def stream():
    return Response(gen_frames(), mimetype='multipart/x-mixed-replace;
boundary=frame')

def analyze_image(image_path, output_path):
    img = cv2.imread(image_path)
    if img is None:
        return
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    hist = cv2.calcHist([gray],[0],None,[256],[0,256])
    mean = np.mean(img)
    std = np.std(img)
    with open(output_path, "w") as f:
        f.write("Mean:" + str(mean) + "\n")
        f.write("Std:" + str(std) + "\n")
        for i in range(256):
            f.write(str(i) + ":" + str(hist[i][0]) + "\n")

def main():
    import sys
    if len(sys.argv) < 2:
        print("Usage: python script.py
[master|worker|stream|analyze|fec_test|retrans_test]")
        sys.exit(1)
    mode = sys.argv[1].lower()
    if mode == "master":
        worker_ports = [8001, 8002]
        master = MasterServer('127.0.0.1', 8000, worker_ports)
        master.start()
    elif mode == "worker":
        port = int(sys.argv[2]) if len(sys.argv) > 2 else 8001
        worker = WorkerServer('127.0.0.1', port, "./worker_storage")
        worker.start()
    elif mode == "stream":
        app.run(host='0.0.0.0', port=9000)
    elif mode == "analyze":
        image_path = sys.argv[2] if len(sys.argv) > 2 else "test.jpg"
```

```
        output_path = sys.argv[3] if len(sys.argv) > 3 else "analysis.txt"
        analyze_image(image_path, output_path)
    elif mode == "fec_test":
        test_data = b"This is a test data for FEC module."
        blocks = fec_encode(test_data, 10)
        blocks[random.randint(0, len(blocks)-2)] = None
        recovered = fec_decode(blocks, 10)
        print("Recovered data:", recovered)
    elif mode == "retrans_test":
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.bind(('127.0.0.1', 9500))
        s.listen(1)
        threading.Thread(target=server_side_retrans, args=(s,)).start()
        time.sleep(1)
        client_side_retrans()
    else:
        print("Invalid mode")
def server_side_retrans(s):
    conn, addr = s.accept()
    data = receive_with_retransmission(conn)
    if data:
        with open("retrans_received.dat", "wb") as f:
            f.write(data)
    conn.close()
def client_side_retrans():
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('127.0.0.1', 9500))
    test_data = b"This is test data for retransmission mechanism. It should be
sent reliably."
    send_with_retransmission(s, test_data)
    s.close()
if __name__ == '__main__':
    main()
```

Файл TransferScheduler.py

```
import socket
import threading
import time
import random
import struct
import os
import sys
import cv2
import numpy as np
import zlib
import hashlib

class NetworkSimulator:
    def __init__(self, sock, delay=0.1, packet_loss=0.1, jitter=0.05):
        self.sock = sock
        self.delay = delay
        self.packet_loss = packet_loss
        self.jitter = jitter
    def send(self, data):
        total_sent = 0
        while total_sent < len(data):
            chunk = data[total_sent:total_sent+1024]
            time.sleep(self.delay + random.uniform(0, self.jitter))
            if random.random() < self.packet_loss:
                total_sent += len(chunk)
                continue
            sent = self.sock.send(chunk)
            total_sent += sent
        return total_sent
    def recv(self, bufsize):
        time.sleep(self.delay + random.uniform(0, self.jitter))
        if random.random() < self.packet_loss:
            return b''
        return self.sock.recv(bufsize)
    def close(self):
        self.sock.close()

class TransferScheduler:
    def __init__(self):
        self.tasks = []
        self.lock = threading.Lock()
    def schedule_transfer(self, func, delay, *args, **kwargs):
        t = threading.Timer(delay, func, args=args, kwargs=kwargs)
        with self.lock:
            self.tasks.append(t)
        t.start()
    def cancel_all(self):
        with self.lock:
            for t in self.tasks:
                t.cancel()
            self.tasks = []
    def get_scheduled_count(self):
        with self.lock:
            return len(self.tasks)

def simulate_file_transfer(file_path, target_ip, target_port, delay=0.1,
packet_loss=0.1, jitter=0.05):
    if not os.path.exists(file_path):
        return
    with open(file_path, 'rb') as f:
        file_data = f.read()
```

```
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((target_ip, target_port))
    sim_sock = NetworkSimulator(s, delay, packet_loss, jitter)
    total_size = len(file_data)
    sim_sock.send(struct.pack('>Q', total_size))
    ack = sim_sock.recv(2)
    if ack != b'OK':
        sim_sock.close()
        return
    sent = 0
    while sent < total_size:
        chunk = file_data[sent:sent+1024]
        sim_sock.send(chunk)
        sent += len(chunk)
    sim_sock.close()

def simulated_server(target_ip, target_port, delay=0.1, packet_loss=0.1,
jitter=0.05):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((target_ip, target_port))
    s.listen(5)
    while True:
        client, addr = s.accept()
        sim_sock = NetworkSimulator(client, delay, packet_loss, jitter)
        total_size_bytes = sim_sock.recv(8)
        if not total_size_bytes:
            sim_sock.close()
            continue
        total_size = struct.unpack('>Q', total_size_bytes)[0]
        sim_sock.send(b'OK')
        received = bytearray()
        while len(received) < total_size:
            chunk = sim_sock.recv(1024)
            if not chunk:
                break
            received.extend(chunk)
        file_name = f"sim_received_{int(time.time())}.dat"
        with open(file_name, 'wb') as f:
            f.write(received)
        sim_sock.close()

def start_simulated_server(target_ip, target_port, delay, packet_loss, jitter):
    server_thread = threading.Thread(target=simulated_server, args=(target_ip,
target_port, delay, packet_loss, jitter))
    server_thread.daemon = True
    server_thread.start()

def schedule_periodic_transfer(scheduler, file_path, target_ip, target_port,
interval, delay, packet_loss, jitter, count):
    def transfer_task(n):
        simulate_file_transfer(file_path, target_ip, target_port, delay,
packet_loss, jitter)
    for i in range(count):
        scheduler.schedule_transfer(transfer_task, i * interval, i)

def simulate_video_capture_transfer(target_ip, target_port, duration=10,
delay=0.1, packet_loss=0.1, jitter=0.05):
    cap = cv2.VideoCapture(0)
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((target_ip, target_port))
    sim_sock = NetworkSimulator(s, delay, packet_loss, jitter)
    start_time = time.time()
    while time.time() - start_time < duration:
```

```
            ret, frame = cap.read()
            if not ret:
                continue
            ret, buffer = cv2.imencode('.jpg', frame)
            if not ret:
                continue
            frame_data = buffer.tobytes()
            frame_size = len(frame_data)
            sim_sock.send(struct.pack('>I', frame_size))
            sim_sock.send(frame_data)
            time.sleep(0.05)
        cap.release()
        sim_sock.close()

def simulated_video_server(target_ip, target_port, delay=0.1, packet_loss=0.1,
jitter=0.05):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind((target_ip, target_port))
    s.listen(5)
    while True:
        client, addr = s.accept()
        sim_sock = NetworkSimulator(client, delay, packet_loss, jitter)
        frames = []
        while True:
            size_data = sim_sock.recv(4)
            if not size_data:
                break
            frame_size = struct.unpack('>I', size_data)[0]
            frame_data = sim_sock.recv(frame_size)
            if not frame_data:
                break
            np_frame = np.frombuffer(frame_data, np.uint8)
            frame = cv2.imdecode(np_frame, cv2.IMREAD_COLOR)
            if frame is not None:
                frames.append(frame)
        video_name = f"sim_video_{int(time.time())}.avi"
        if frames:
            height, width, _ = frames[0].shape
            fourcc = cv2.VideoWriter_fourcc(*'XVID')
            out = cv2.VideoWriter(video_name, fourcc, 20.0, (width, height))
            for f in frames:
                out.write(f)
            out.release()
        sim_sock.close()

def start_simulated_video_server(target_ip, target_port, delay, packet_loss,
jitter):
    video_server_thread = threading.Thread(target=simulated_video_server,
args=(target_ip, target_port, delay, packet_loss, jitter))
    video_server_thread.daemon = True
    video_server_thread.start()

def main():
    if len(sys.argv) < 2:
        print("Usage: python script.py [simulate_file|simulate_video|schedule]")
        sys.exit(1)
    mode = sys.argv[1].lower()
    if mode == "simulate_file":
        if len(sys.argv) != 8:
            print("Usage: python script.py simulate_file file_path target_ip
target_port delay packet_loss jitter")
            sys.exit(1)
        file_path = sys.argv[2]
```

```
        target_ip = sys.argv[3]
        target_port = int(sys.argv[4])
        delay = float(sys.argv[5])
        packet_loss = float(sys.argv[6])
        jitter = float(sys.argv[7])
        simulate_file_transfer(file_path, target_ip, target_port, delay,
packet_loss, jitter)
    elif mode == "simulate_video":
        if len(sys.argv) != 7:
            print("Usage: python script.py simulate_video target_ip target_port
duration delay packet_loss jitter")
            sys.exit(1)
        target_ip = sys.argv[2]
        target_port = int(sys.argv[3])
        duration = float(sys.argv[4])
        delay = float(sys.argv[5])
        packet_loss = float(sys.argv[6])
        jitter = float(sys.argv[7])
        simulate_video_capture_transfer(target_ip, target_port, duration, delay,
packet_loss, jitter)
    elif mode == "schedule":
        scheduler = TransferScheduler()
        target_ip = "127.0.0.1"
        target_port = 9900
        start_simulated_server(target_ip, target_port, 0.1, 0.1, 0.05)
        schedule_periodic_transfer(scheduler, "sample_transfer.dat", target_ip,
target_port, 5, 0.1, 0.1, 0.05, 5)
        start_simulated_video_server(target_ip, 9950, 0.1, 0.1, 0.05)
        scheduler.schedule_transfer(simulate_video_capture_transfer, 2, "dummy",
target_ip, 9950, 10, 0.1, 0.1, 0.05)
        while True:
            time.sleep(1)
    else:
        print("Invalid mode")
        sys.exit(1)

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