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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем”

Виконав здобувач вищої освіти
II курсу, групи КН-21М-1,4
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Водзинський Є.Ю.
«____» _____ 2022 р.

Керівник проекту
кандидат технічних наук
_____ Смірнова Т.В.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 122 *"Комп'ютерні науки"*

Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Водзинському Євгену Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем*

2. Керівник роботи *Смірнова Тетяна Віталіївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 18-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Смірнова Т.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Водзинський Є.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Водзинський Є.Ю. Дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Метою розробки є дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Об'єктом дослідження є процес адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Предметом дослідження є методи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі Embarcadero Delphi.

Ключові слова: комп'ютерні науки, маршрутизація, мережі інформаційних та комп'ютерних систем

ABSTRACT

Vodzynskyi E.Yu. Research and software implementation of the system of adaptive routing optimization of the network of information and computer systems. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of adaptive optimization of the routing of information and computer systems networks.

The purpose of the development is the research and software implementation of the system of adaptive optimization of the routing of the network of information and computer systems.

The object of the study is the process of adaptive optimization of the routing of the network of information and computer systems.

The subject of the research is the methods of adaptive optimization of the routing of the network of information and computer systems.

Research methods are based on methods of building computer networks, methods of mathematical statistics, and methods of software development.

The result of the work is the software implementation of the system of adaptive optimization of the routing of information and computer systems networks.

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 tools are provided.

The program can be used on PCs of IBM PC architecture with Windows 10/11 OS.

The program was developed in the Embarcadero Delphi environment.

Keywords: computer science, routing, networks of information and computer systems

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ІПМ	–	імітаційна програмна модель
ЛОМ	–	локальні обчислювальні мережі
ПрЗд	–	пропускна здатність
ПМ	–	програмна модель
ТПП	–	таймер повторної передачі
ARTCP	–	удосконалений транспортний протокол
BER	–	імовірність бітової помилки
CR	–	запит на з'єднання
CBR	–	протокол передачі даних без підтверджень
FIFO	–	принцип first input fipst output
IP	–	адресний протокол
OSI	–	еталона модель мережної архітектури
RTT	–	час затрачуваємий підтвердженням
TCP	–	протокол транспортного рівня
TCP/IP	–	протокол передачі даних
TPDU	–	повідомлення транспортного протоколу
UDP	–	протокол користувальницьких датаграмм
WAN	–	територіальні мережі

ВСТУП

Актуальність теми. Одним з найважливіших напрямків науково-технічного прогресу в цей час є комунікаційні системи, що представляють собою мережі передачі інформації. Координацію процесів передачі інформації в розподіленій системі, якою є мережа, здійснюють комунікаційні протоколи.

Прийнято розділяти комунікаційні протоколи по ступені спільності задач, розв'язуваних ними, на кілька рівнів, упорядкований набір яких утворить мережну архітектуру. Найпоширенішою й універсальною мережною архітектурою є архітектура TCP/IP [1]. У рамках TCP/IP всі системи в мережі діляться на кінцеві системи, між якими відбувається інформаційний обмін, і проміжні системи, що не є кінцевими або вихідними точками обміну. Кінцеві системи називаються вузлами мережі, а проміжні – маршрутизаторами.

Двосторонній потік інформації між парою суміжних систем у мережі забезпечується каналом, що зв'язує дві системи. Канали характеризуються швидкістю інформаційного потоку (пропускною здатністю), затримкою передачі й імовірністю бітових помилок. У кожній точці підключення маршрутизатора до каналу є буфер, у якому організується чергу даних відправлення, що очікують, по цьому каналі. Буферний простір і пропускна здатність (ПрЗд) являють собою поділювані ресурси мережі. Якщо швидкість прибуття інформації в маршрутизатор перевищує максимально можливу швидкість її відправлення, то відбувається перевантаження мережі, що виражається в переповненні буферів і втратах інформації.

Протокол транспортного рівня займає найважливіше положення в будь-якій мережній архітектурі, у тому числі й в TCP/IP, оскільки він забезпечує надійну й ефективну передачу інформації безпосередньо між кінцевими системами мережі. Для цього транспортний протокол задає погоджений набір правил поведінки для учасників інформаційного обміну. Ці правила регулюють

спільний доступ вузлів до поділюваних ресурсів мережі, тому ефективність транспортного протоколу визначає ефективність роботи всієї мережі в цілому. Програма, що реалізує алгоритм протоколу, називається об'єктом протоколу.

Транспортним протоколом в архітектурі TCP/IP є TCP (Transmission Control Protocol) [4,5,6], який забезпечує надійний двосторонній зв'язок з контролем швидкості передачі. Джерело TCP потоку одержує інформацію від користувача у вигляді послідовності біт, формує з її блоки кінцевої довжини, називані сегментами й відправляє їх до TCP одержувача. Одержувач, приймаючи сегменти, формує з них вихідну послідовність і передає її своєму користувачеві.

Для здійснення обміну TCP установлює логічне з'єднання між парою вузлів мережі, на кожному з яких виконується алгоритм TCP. Потік сегментів по TCP з'єднанню може проходити через упорядковану послідовність маршрутизаторів і каналів. Пропускна здатність з'єднання в цілому обмежена мінімальної із ПрЗд каналів, через які проходить з'єднання. Алгоритм керування потоком, що є частиною TCP, прагне відправляти дані зі швидкістю, що не перевищує менше із ПрЗд з'єднання й швидкості споживання інформації одержувачем.

Набір з'єднань транспортного протоколу, що розділяють загальний канал, являє собою складну систему, що самоорганізується, у змісті Г. Хакена [99]. Поводження кожного з об'єктів протоколу в цій системі визначається алгоритмом протоколу, однак, поводження всієї системи, як цілого, загалом кажучи, не описується сукупністю дій її компонентів. Кожний об'єкт протоколу прагне максимально ефективно адаптуватися до доступних ресурсів мережі в умовах кооперації з іншими об'єктами цього протоколу.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем.

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

– Огляд існуючих систем адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– Дослідження системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– Програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Об'єктом дослідження є процес адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Предметом дослідження є методи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– Розроблено вітчизняний продукт адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Недоліки протоколу TCP

До найбільш істотних недоліків протоколу TCP в області керування потоками відноситься наступне:

1. До основного недоліку протоколу TCP варто віднести проблему не стільки реалізації протоколу, скільки самої логіки функціонування. Так механізми корекції помилок і керування потоком в TCP, що реалізують різні функції, виявилися зв'язаними. Причиною цього є інтерпретація протоколом факту втрати сегмента (і, відповідно, факту спрацьовування механізму корекції помилок) як ознаки перевантаження мережі. Внаслідок цього TCP реагує на будь-яку втрату даних зниженням швидкості передачі. Результатом є вкрай низька ефективність застосування протоколу TCP для систем, де втрати даних відбуваються не тільки внаслідок перевантаження – всіх систем, де існує ненульова ймовірність втрати даних на фізичному-канальному рівнях. Це, зокрема, всі види бездротових мереж: супутникові, стільникові, оптичні.

2. Застосовуваний в TCP метод AIMD пропонує постійне лінійне збільшення навантаження на мережу з метою визначення моменту початку перевантаження. Внаслідок цього мережа постійно перебуває або в стані перевантаження, або в стані виходу з її. Це негативно позначається на з'єднаннях у вигляді збільшеного середнього RTT (часу затрачуваного підтвердженням), великої дисперсії вимірюваних значень RTT, постійному наявності втрат, за допомогою яких мережа сигналізує про початок перевантаження.

3. У більшості умов протокол TCP здійснює відправлення даних дуже нерівномірно. Фактично всі дані в межах вікна відправляються за невеликий час у вигляді сплеску пакетів [46]. Нерівномірний режим відправлення пакетів

приводить до підвищення числа втрат. Оскільки середня довжина черг у мережних пристроях близька до максимального, то ймовірність втрати сегментів у межах сплеску підвищується.

4. Механізм керування передачею TCP залежить від потоку підтверджень у зворотному напрямку, прибуття яких змушує переміщатися вікно й дозволяє відправлення нових сегментів. Такий режим називають синхронізованим по підтвердженнях. Його ефект не помітний для симетричних каналів, однак для каналів, чиї властивості в різних напрямках різні, синхронізація по підтвердженнях веде до обмеження ефективності використання ресурсів. Це відноситься до таких асиметричних систем, як «супутниковий/наземний канали», «кабельний модем/ з'єднання, що комутирується».

Оскільки недоліки TCP широко відомі, то багато робіт бути присвячено створенню окремого транспортного протоколу для асиметричних інфраструктур [51], або бездротових мереж [47,48,49,50]. Більшість запропонованих протоколів не є сумісними з TCP і вимагають наявності агентів-посередників транспортного рівня. Тим самим порушується основний постулат Інтернет який заключається в тому, що мережа повинна забезпечувати безперешкодний зв'язок на транспортному рівні між безпосередніми учасниками обміну. Також зайво ускладнюється вся структура мережі.

Наша робота була спрямована на створення удосконаленого протоколу, названого ARTCP, що міг би стати ефективною заміною існуючому TCP, усунувши найважливіші недоліки останнього.

Ціль роботи

Отже, задача даної роботи полягає в удосконаленні механізму керування потоком для транспортного протоколу в архітектурі мережі з комутацією пакетів (TCP/IP). Новий алгоритм керування потоком повинен виправити основні недоліки, властиві протоколу TCP наведені вище. При цьому новий протокол повинен зберегти всі зовнішні властивості й інтерфейси існуючого транспортного

протоколу. Новий протокол повинен існувати в типовому середовищі виконання транспортного протоколу Інтернет і надавати користувачеві той же сервіс.

Для вивчення характеристик протоколу ARTCP і порівняння їх з TCP, у рамках даної роботи необхідно розробити програмну імітаційну модель, що відтворює основні характеристики мережі, які й визначають функціонування в ній транспортного протоколу: затримку, мультиплексування, втрати й помилки передачі.

На створеній імітаційній програмній моделі передбачається провести ряд модельних експериментів для визначення основних параметрів ARTCP у різних умовах і порівняння їх з TCP. Оскільки, як показує множина досліджень, трафік в TCP/IP мережах має властивість самоподоби, то перевірка ARTCP трафіку на наявність у нього властивості самоподоби також є метою модельного експерименту.

Формальна модель системи

Формалізуємо модель на якій будемо вивчати протокол ARTCP.

Є мережа, у топологічну схему якої входять кілька вузлів, два маршрутизатори й набір каналів з'єднуючі вузли (рисунок 1.1). На кожному вузлі виконується об'єкт протоколу ARTCP.

Вузли об'єднані у дві локальні обчислювальні мережі (ЛОМ), кожна з яких підключена до одного маршрутизатора. Маршрутизатори зв'язують локальні мережі, передаючи трафик по каналі з малої ПрЗд і більшим значенням затримки.

Кожний з вузлів в одній із ЛОМ відправляє сегменти із заданим міжсегментним інтервалом, що і визначає швидкість передачі. Ця швидкість задається алгоритмом керування потоком ARTCP. Передбачається, що джерела потоків завжди мають інформацію для відправлення. Джерела й приймачі трафіку перебувають у різних ЛОМ. Приймачі трафіку ARTCP відправляють підтвердження в протилежному напрямку у вигляді сегментів, що не містять даних.

Задача маршрутизатора в тому, щоб здійснювати передачу сегмента за адресою одержувача в його заголовку. У буфері маршрутизатора R1 організується FIFO черга сегментів, які відправляються далі до маршрутизатора R2. Черга має кінцеву довжину. Сегмент, що надходить на вихідний інтерфейс, міститься в чергу, якщо вона може вмістити цей сегмент, інакше сегмент губиться. Черга маршрутизатора R1 обслуговується зі швидкістю каналу, що з'єднує маршрутизатори.

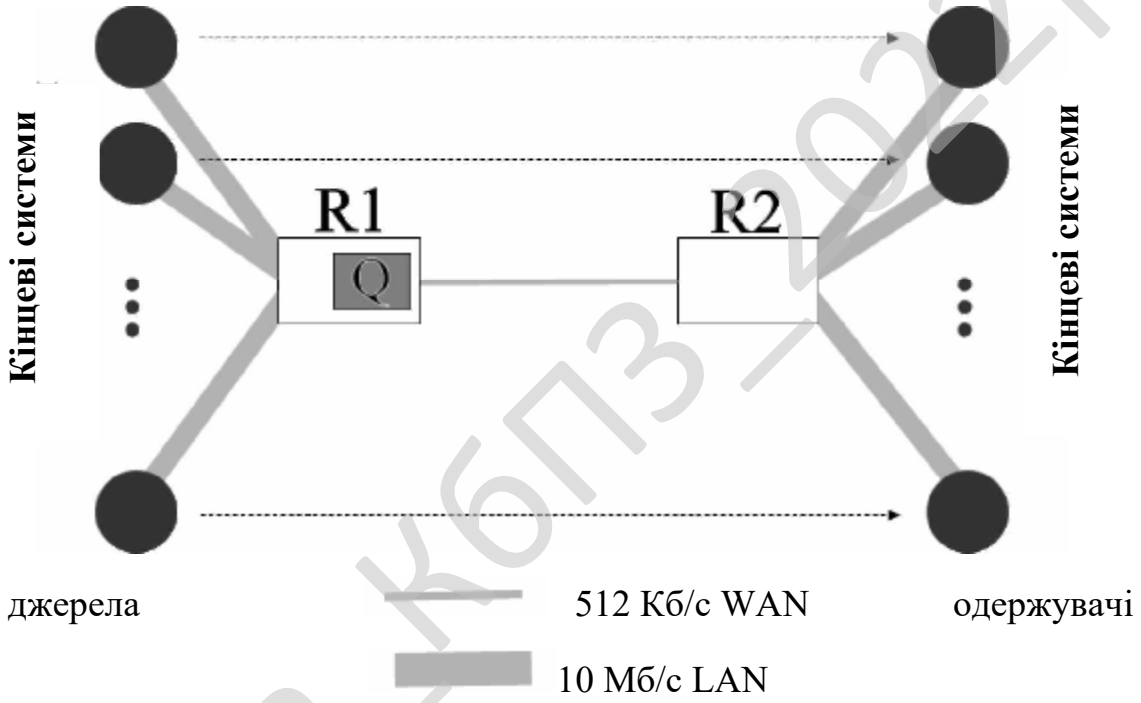

Рисунок 1.1 – Формальна модель досліджуваної мережі. Стрілками зазначений напрямок передачі даних

Ми розглядаємо наступні характеристики каналів: ПрЗд, затримку передачі й імовірність бітової помилки.

Пропускна здатність каналу визначає швидкість надходження біт у канал, затримка передачі характеризує тривалість інтервалу між надходженням певного біта в канал і появою його з каналу. Імовірність бітової помилки BER визначає

ймовірність втрати сегмента як $1 – (1 – BER)^S$ залежно від імовірності помилки при передачі.

Основні характеристики протоколу

У роботі розглядаються наступні основні характеристики протоколу:

1. Відносне число втрат сегментів (відношення числа загублених до загального числа відправлених сегментів)

2. Коефіцієнт використання пропускної здатності каналів:

$$U = \frac{число\ прийнятих\ біт}{швидкість\ каналу \times час}, \quad (1.1)$$

(відношення числа успішно прийнятих битов до максимально можливого їхнього числа).

3. Коефіцієнт рівноправності поділу ресурсів:

$$F = \left(\sum_{i=1}^{n} b_i\right)^2 \Big/ n \times \left(\sum_{i=1}^{n} b_i\right)^2, \quad (1.2)$$

де b – частка пропускної здатності, зайнята i-м з'єднанням.

4. Середня довжина черги Q у маршрутизаторі R1.

Порівняння ARTCP і TCP виробляється по цих основних характеристиках.

Наведені вище характеристики протоколу є стандартними й використовуються в багатьох дослідженнях протоколів [73].

1.2 Область застосування

Областю застосування є мережі передачі даних де використовується стек протоколів TCP/IP. Тобто любі локальні, корпоративні та глобальні мережі.

Розглянемо мережу у вигляді системи, що самоорганізується.

Визначення системи, що самоорганізується, у змісті Г. Хакена наступне: система з великим числом ступенів свободи, під впливом складних внутрішніх сил, що переходить у стан з істотно меншим числом ступенів свободи: декількома параметрами порядку, називається системою, що самоорганізується

[101]. Хаотичне поводження системи на рівні окремих компонентів приводить до детермінованого поводження її середніх величин.

Розглянута нами модель мережі являє собою набір об'єктів транспортного протоколу ARTCP і об'єктів мережі. Кожний об'єкт протоколу може управляти власною швидкістю передачі потоку, як система зі зворотним зв'язком. Взаємодія між всіма потоками здійснюється через загальні для цих потоків об'єкти топології. Використання випадкової змінної алгоритмом кожного із протоколів, замість збільшення числа ступенів свободи, приводить до впорядкування сумарної дії всіх елементів системи. Це впорядкування виражається в тому, що сумарна швидкість всіх потоків протоколу ARTCP вирівнюється зі швидкістю спільно використовуваного цими потоками каналу. Таким чином, мережа з функціонуючими в ній з'єднаннями транспортного протоколу ARTCP, є системою, що самоорганізується.

Все вищесказане дозволяє сформулювати це твердження в наступному виді: Розглянута мережа із з'єднаннями протоколу ARTCP, є системою, що самоорганізується, у змісті Г. Хакена.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Комунікаційні транспортні протоколи

Сучасні комунікаційні мережі

Важливість мереж передачі даних і мультимедіа інформації неможливо переоцінити сьогодні, на початку інформаційної епохи розвитку людства. Системи збору, обробки й розподіли інформації є найважливішою точкою додатка наукового знання. Розвиток технології усунуло розходження між обробкою й передачею інформації, а найбільш актуальною задачею в комунікації стала специфікація й верифікація комунікаційних протоколів, а також підвищення їхньої ефективності.

Сучасне визначення мережі передачі даних таке: мережа це набір автономних обчислювальних пристроїв фізично й логічно зв'язаних між собою. Іншими словами набір рівноправних вузлів маючих можливість обмінюватися інформацією. Таке визначення ставиться до так званої мережі передачі даних, однак, не обмежується тільки комп'ютерними даними й може бути розширене для будь-якої мережі, побудованої за принципом комутації пакетів, у тому числі й універсальній мережі з інтеграцією сервісів.

Мережі передачі даних і мультимедіа інформації в сучасному суспільстві застосовуються повсюдно – у виробництві й у сфері керування, у дослідницькому середовищі й у побуті. Економічна вигода від підвищення ефективності комунікаційних протоколів може виявитися дуже істотної, особливо з урахуванням подальшого росту й розвитку комунікаційних систем.

Будь-яка комунікаційна система складається із двох основних компонентів: апаратної й логічної частин.

Апаратна інфраструктура мереж

Апаратна частина це фізична інфраструктура, за допомогою якої здійснюється поширення фізичних сигналів інформаційні повідомлення, що кодують. Фізична інфраструктура може або здійснювати інформаційний обмін між всіма учасниками мережі одночасно (широкомовна мережа), або тільки між двома вузлами (мережа типу точка-точка). Залежно від масштабу й топології мережної інфраструктури роблять її подальшу класифікацію: локальні обчислювальні мережі (ЛОМ), територіальні мережі (WAN), стільникові мережі й т.і. Багато властивостей сімейства протоколів для комунікаційної системи перебувають у прямої залежності від її фізичного середовища й топології, тому, параметри фізичної інфраструктури мережі ретельно враховуються при розробці її протоколів.

Системи протоколів

Найважливішим компонентом мережі є комунікаційні протоколи, що становлять її логічний компонент. Як неформальне визначення протоколу можна привести наступне: протокол являє собою угоду по процедурі обміну інформацією в розподіленій системі.

Задачею будь-якого протоколу є керування спільним використанням ресурсів. Протоколи подібні з мовами. Визначення протоколу дається в такий спосіб – через визначення його функціональних компонентів [3]:

– протокол визначає точний формат припустимих повідомлень (синтаксис);

– протокол визначає процедурні правила для обміну інформацією (граматика);

– протокол визначає словник правильних повідомлень і їхнього значення (семантика); Розробка, опис і верифікація протоколів становлять найбільш складну задачу в рішенні проблеми створення комунікаційних систем.

Методи формального опису

Існують кілька методів для формального опису протоколів. Це мови SDL (Specification and description language), Lotos і Estelle. Найбільше часто застосовується мова SDL, що є стандартом ITU[2]. Існують дві форми SDL – графічна й програмна [55]. Природно, що із всіх компонентів протоколу саме процедурні правила можуть бути дані за допомогою тої або іншої мови формального опису.

Найбільш складною проблемою розробки або підвищення ефективності протоколу є, таким чином, створення великого набору процедурних правил, що має наступні властивості:

– Він повинен бути мінімальним, тобто не містити недосяжних елементів або ділянок коду.

– Він повинен бути логічно зв'язковим.

– Він повинен задовольняти умові повноти.

– Цей набір повинен бути легко реалізуємо (апаратним або програмним образом).

П'ять елементів протоколу

На додаток до визначення, даному вище, розробка протоколу припускає детальну специфікацію сервісу надаваного протоколом користувачеві, яким може бути прикладна програма або протокол більше високого рівня, а також знання характеристик середовища, у якій протокол буде виконуватися.

Таким чином, виділяють п'ять компонентів протоколу, кожний з яких повинен бути заданий тим або іншому методу формального опису:

1. Сервіс, надаваний користувачеві.
2. Передбачувана характеристика середовища виконання протоколу.
3. Словник припустимих повідомлень.
4. Кодування повідомлень – формат кожного повідомлення зі словника.
5. Процедурні правила, що контролюють обмін повідомленнями.

Формалізований опис сервісу й характеристик середовища виконання протоколу знаходять висвітлення в процедурних правилах. Для реалізації своєї задачі – надання сервісу верхнього рівня протокол повинен виконувати ряд задач низького рівня: синхронізації, розпізнавання й корекції помилок. Те, як протокол виконує ці задачі, залежить від властивостей середовища його виконання.

Словник припустимих повідомлень і їхнє кодування також пов'язані із припущеннями про середовище виконання протоколу – середовище передачі даних. Імовірність виникнення бітових помилок і середня довжина поля послідовності бітових помилок впливають на набір, формат і кодування повідомлень протоколу, а також вибір алгоритму розпізнавання або корекції помилок.

Модель ISO OSI RM

Міжнародною організацією по стандартизації (ISO) з метою уніфікації методів розробки протоколів була запропонована так звана еталонна модель взаємодії відкритих систем (OSI RM).

Еталонна модель OSI RM складається із семи рівнів:

1. Фізичний рівень. У задачу цього рівня входить коректна передача бітового потоку по фізичній лінії зв'язку. Коли відправник передає «1», біт приймаюча сторона повинна одержати його як «1» біт, а не «0». Специфікації цього рівня задають як логічні, так і фізичні й схемотехнічні параметри. Фізичний рівень розглядає інформацію як безперервний бітовий потік.

2. Канальний рівень. Цей рівень створює логічний канал, що не має помилок і збоїв, для потреб мережного рівня. Інформація, що відправляється, розбивається на блоки, називані кадрами, які пересилаються по черзі. У деяких випадках рівень створює й обробляє кадри з підтвердженням коректного прийому даних. Оскільки фізичний рівень має справу з неструктурованим потоком біт, те канальний рівень повинен піклуватися про створення й відстеження границь кадрів. Також канальний рівень може застосовувати різні алгоритми для визначення й по можливості корекції перекручування інформації. Канальний

рівень бере на себе всі турботи про акуратне надання інформації мережному рівню. Інша проблема, що доводиться вирішувати на багатьох рівнях, включаючи канальний, те, як не допустити переповнення повільного одержувача даними від швидкого відправника. Необхідний деякий механізм, що дозволяє відправникові інформації мати дані про наявність вільного буферного простору в одержувача й діяти відповідно до цього знання. Мережі із широкомовною передачею вносять додаткові складності в пристрій канального рівня – керування множинним доступом до каналу.

3. Мережний рівень. Задачею мережного рівня є керування роботою базової підмережі. Найважливіша проблема – розрахувати шлях від точки відправлення те одержувача. Такий маршрут може бути заснований на статичних таблицях пристроїв, що зберігаються в пам'яті, підмережі або ж маршрут може визначатися на початку кожної сесії, альтернативним варіантом є високодинамічна маршрутизація, коли шлях заново визначається для кожного окремого пакета. Якщо в базову підмережу попадає більше пакетів, ніж мережа може обробляти, то виникає перевантаження, боротьба або профілактика якої також є задачею мережного рівня. Одиниця інформаційного обміну на мережному рівні називається пакетом.

4. Транспортний рівень. Транспортний рівень надає послуги (і приховує від вищележачих рівнів) по надійному транспортуванню даних по мережі. Даний рівень забезпечує відкриття, підтримку й нормальне відключення віртуальних каналів, передає повідомлення про помилки й збої, здійснює керування швидкістю потоків інформації, щоб уникнути переповнення буферів мережних пристроїв і приймачів і, відповідно, втрат даних. Відмінною рисою транспортного рівня є те, що об'єкти протоколу транспортного рівня здійснюють обмін прямо, поза залежністю від базової підмережі. Одиниця інформаційного обміну на транспортному рівні називається сегментом або датаграммой.

5. Сеансовий рівень. Цей рівень синхронізує, відкриває, закриває й маніпулює сеансами зв'язку, активними об'єктами рівня презентації (якихось

може бути трохи). Рівень СЕАНСУ привласнює ступінь терміновості повідомленням, керує прискореною передачею повідомлень про виниклі помилки вищележачим рівням.

6. Рівень презентації. Рівень презентації здійснює кодування даних, забезпечуючи сумісність для різних форматів подання інформації.

7. Рівень додатка. Цей рівень відрізняється від інших тим, що його об'єктами є безпосередньо додатки, що виконуються на обчислювальному вузлі. Функція цього рівня полягає у визначенні клієнтів, яким необхідна комунікація, синхронізації комунікації, визначенні наявність ресурсів для проведення сеансу комунікації й детектування помилок.

Модель TCP/IP

У середині 1970-х американською військовою дослідницькою організацією DARPA було ухвалене рішення про створення мережі з комутацією пакетів для забезпечення з'єднання дослідницьких установ на території Сполучених Штатів. У той час дослідники й виробники вперше зштовхнулися із проблемою організації в мережу різноманітних і гетерогенних комп'ютерних систем. З метою вироблення протоколів зв'язку для різнорідних систем DARPA почала фінансування досліджень проведених у Станфордском університеті по створенню сімейства мережних протоколів. Другою метою було створення мережі имеющей можливість продовжувати функціонувати навіть при виході з ладу істотної частки її апаратної частини. У результаті цієї роботи з'явилися протоколи сімейства Internet protocols. Здатність нової розробки з'єднувати мережі, засновані на різних технологіях, зовсім прозорим образом була основною задачею розроблювачів із самого початку. Найбільш практичними й широко використовуваними членами цього сімейства є протоколи Transmission Control Protocol (TCP) – протокол контролю передачі й Internet protocol (IP) протокол інтернету, а сама архітектура стала відомою за назвою еталонної моделі TCP/IP.

Сімейство протоколів TCP/IP застосовується для комунікації через будь-яку кількість проміжних ЛОМ. Протоколи TCP/IP ідеально підходять як для

комунікації в ЛОМ, так і для глобальних обчислювальних мереж. Набір протоколів TCP/IP містить специфікації не тільки протоколів низького рівня, таких як TCP або IP, але також і таких широко розповсюджених додатків як електронна пошта, емуляція віддаленого термінала, протокол передачі файлів і багато чого іншого.

Однак рівні моделі TCP/IP не повністю збігаються з рівнями моделі OSI. Основні складові моделі TCP/IP відповідають мережному й транспортному рівням OSI. Ці протоколи – IP і TCP є ключовими для концепції сучасної комунікаційної архітектури.

Мережний протокол моделі TCP/IP: протокол IP

Протокол IP є єдиним протоколом мережного рівня сімейства TCP/IP, тому всі транспортні протоколи стека TCP/IP використовують сервіси протоколу IP. Тобто сервіс, що представляється IP, є середовищем, у якому виконується протокол TCP.

Протокол IP надає користувачеві ненадійний сервіс по передачі пакетів. На додаток до функцій маршрутизації, тобто визначення оптимального маршруту пакетів від відправника до одержувача на підставі інформації про топологію мережі, IP також відповідальний за фрагментацію й складання пакетів і повідомлення про збійні ситуації.

Транспортні протоколи моделі TCP/IP

Транспортний рівень набору протоколів інтернету складається із двох протоколів: TCP (Transmission Control Protocol – протокол контролю передачі) і UDP (User Datagram Protocol – протокол користувальницьких датаграмм).

TCP надає надійний транспорт із установкою логічного з'єднання. TCP є найбільш важливим транспортним протоколом моделі TCP/IP він забезпечує повністю дуплексний зв'язок з кумулятивним підтвердженням прийому. Він переміщає інформацію у вигляді безперервного неструктурованого потоку, де байти ідентифікуються порядковим номером. Об'єкт протоколу TCP може

одночасно підтримувати множину сеансів інформаційного обміну для протоколів вищих рівнів, здійснюючи мультиплексування потоків.

Транспортний рівень: роль і компоненти

Транспортний рівень є центральним для всієї ієрархії протоколів. Задача цього рівня – забезпечувати надійне й ефективне транспортування інформації від вихідного до кінцевого пристрою поза залежністю від фізичних особливостей мереж або обмежень, що накладаються протоколами мереж.

Сервіс транспортного рівня

Це сервіси, надавані користувачеві. Головна задача транспортного рівня полягає в наданні ефективного й надійного сервісу для відповідних користувачів – головним чином користувальницьким процесам рівня додатка. Для виконання цих задач транспортний рівень покладається на сервіси мережного рівня. Апаратне й/або програмне забезпечення, що виконує функції транспортного рівня називається транспортним об'єктом. Цей об'єкт може бути частиною ядра операційної системи, окремим процесом у користувальницькому просторі, у вигляді бібліотечних об'єктів скомпонованих з користувальницькими програмами або функцією окремої інтерфейсної плати. Блок даних, за допомогою якого відбувається обмін між об'єктами транспортного протоколу, називається TPDU (повідомлення транспортного протоколу). На відміну від всіх нижніх рівнів, транспортний рівень функціонує безпосередньо між учасниками обміну інформацією, прямо й дозволяє цим системам обмінюватися інформацією поза залежністю від проміжних мереж і систем. Транспортні протоколи уможливлюють надавати користувачеві транспортний сервіс набагато більш надійний, ніж використовуваний мережний сервіс. Наявність загублених і ушкоджених пакетів контролюється й компенсується транспортним рівнем. Крім того, сервісні примітиви транспортного рівня можуть бути розроблені таким чином, щоб бути повністю незалежними від мережних сервіс-примітивів, які радикально розрізняються для різних типів мереж. Завдяки наявності транспортного рівня, користувальницькі додатки можуть розроблятися із

застосуванням стандартного інтерфейсу (стандартного набору викликів сервіс примітивів) і використовуватися без змін на самих різних типах мережних технологій, що й має місце на практиці. Транспортний рівень ізолює прикладні програми від особливостей технології, розробки, ненадійності й різнорідності мереж. Саме тому існує поділ рівнів моделі OSI RM на дві групи з 1-го по 4-й і вище 4-й. Перша називається постачальником транспортних послуг, друга – користувачем транспортних послуг. Проведення такого розгалуження, по-перше, впливає на специфіку реалізації рівнів і протоколів, а по-друге, ставить транспортний рівень на місце ключової ланки в ієрархічній моделі, оскільки він перебуває на інтерфейсі між постачальником і користувачем надійного транспортного сервісу. Основна задача транспортного рівня в тому, щоб максимально поліпшити якість послуг мережного рівня.

Розглянемо, яким образом протокол TCP поліпшує параметри сервісу в порівнянні із сервісом мережного рівня. Як було сказано раніше сукупний вплив фізичного, канального й мережного рівнів на передані дані виражається в появі трьох типів помилок даних:

- Втрата.
- Дублювання.
- Перекручування порядку відправлення.

Протокол TCP повністю усуває ці помилки. З погляду користувача транспортного протоколу, мережа виглядає як повністю дуплексний канал, що переміщає потік байтів користувача без втрат, причому байти з'являються з каналу в тому самому порядку, у якому вони надійшли в нього на передавальній стороні й тільки однократно. Виконання всіх цих функцій ніяк не залежить від використовуваних технологій мережного, канального й фізичного рівнів, яких може бути багато на маршруті від відправника до одержувача.

Додатку, що використовує сервіс TCP, досить лише запросити зв'язок з іншим додатком, а потім почати передавати свою послідовність даних після

установки логічного з'єднання, не піклуючись про можливі втрати, помилки, керування швидкістю передачі й т.і.

Примітиви транспортного сервісу.

Примітиви транспортного сервісу вже є достатньою формалізацією для опису сервісного компонента протоколу. Через ці примітиви користувачі (додатки) одержують доступ до послуг транспортного рівня. Кожний тип транспортного сервісу має свій набір примітивів. Транспортний рівень пропонує надійний зв'язок з побудовою логічного каналу.

Оскільки реальні мережі не можуть гарантувати відсутності помилок, то задача транспортного рівня саме в тому, щоб забезпечити надійний зв'язок, користуючись ненадійним зв’язком. Наприклад, два процеси использующие іменований комунікаційний канал (pipe) [86] у середовищі ОС UNIX [64], у своїй роботі припускають, що з'єднання є ідеальним. Розроблювачеві програми реалізуючої ці процеси немає необхідності піклуватися про відстеження підтверджень, втрат повідомлень, перевантажень. З'єднання для нього виглядає як абсолютно надійне. Аналогічно, транспортні протоколи представляють ненадійний канал як бітовий потік вільний від помилок для користувача.

Як додатковий сервіс транспортний рівень може пропонувати й ненадійний зв'язок без підтверджень прийому й керування потоком, у середовищі TCP/IP такий сервіс реалізується протоколом UDP.

Різні також і користувачі транспортного й мережного рівня. Мережний сервіс використовується тільки об'єктами транспортного рівня. Украй рідко програми розробляються так, щоб використовувати сервіси мережного рівня прямо. Основна маса додатків розробляється саме розраховуючи на використання сервісів, а, отже, і примітивів транспортного рівня. Тому специфікація сервісу транспортного рівня повинна бути універсальної й простій у використанні, як, наприклад, інтерфейс *Berkeley Sockets.*

Для того щоб скласти уявлення про примітиви транспортного сервісу розглянемо реальний інтерфейс Berkeley Sockets [84,85]

Це інтерфейс для взаємодії додатків з об'єктами транспортних протоколів, включаючи протокол TCP. Споконвічно був розроблений для роботи з TCP в операційній системі сімейства BSD UNIX [64]в 1980 році. Зараз інтерфейс Berkeley Sockets є промисловим стандартом і використовується в більшості операційних систем.

Таблиця 2.1 – Опис примітивів інтерфейсу Berkeley Sockets

Примітив	Пояснення
SOCKET	Створити нову точку доступу до системи зв'язку (ТДС)
BIND	Привласнити локальна адреса ТДС
LISTEN	Оголосити про готовність до прийому з'єднань, установити розмір черги
ACCEPT	Блокувати ініціатора до надходження запиту на з'єднання
CONNECT	Активна спроба установки з'єднання
SEND	Передача даних
RECEIVE	Одержання даних
CLOSE	Завершити сеанс зв'язку

Користувач одержує доступ до даних примітивів на UNIX системі у вигляді системних викликів, оскільки об'єкт протоколу TCP у системі UNIX є частиною ядра операційної системи [86]. Виклик SOCKET у випадку успішного завершення повертає дескриптор файлу й виділяє місце під таблиці в контрольних блоках на транспортному рівні. Параметри виклику вказують на тип і характеристики необхідного сервісу. Після одержання дескриптора, що характеризує одну сторону з'єднання, йому привласнюється локальна адреса за допомогою виклику BIND. Причина використання окремого виклику в тому, що деякі процеси вимагають використання стандартної адреси, для інших же цього не потрібно. LISTEN повідомляючи про готовність приймати з'єднання, виділяє місце під чергу прийнятих сегментів. LISTEN не є викликом, що блокує. Виклик ACCEPT блокує процес, що викликав його в очікуванні з'єднання. Примітив

CONNECT блокує його процес, що викликав, і починає активну спробу відкрити з'єднання. Коли цей виклик вертається, процес деблокується, і може починати обмін інформацією із уже встановленого з'єднання. Розрив з'єднання відбувається при виконанні примітива CLOSE і є симетричним.

Характеристика середовища виконання

Для того, що б визначити середовище виконання протоколу TCP або його запропонованої модифікації ARTCP необхідно розглянути принципи функціонування рівнів ієрархії, сервісами яких користується транспортний протокол. Відповідно до еталонної моделі OSI RM транспортному рівню передують мережний, канальний і фізичний рівні ієрархії. Оскільки задача кожного з рівнів в тому, щоб максимально ізолювати його від проблем нижніх рівнів, досить розглянути лише сервіс мережного рівня.

Сервіси мережного рівня

Доступ до сервісів мережного рівня можливий на інтерфейсі між мережним і транспортним рівнями. Важливість даного інтерфейсу визначається тим, що часто цей інтерфейс відокремлює споживача комунікаційних послуг від їхнього постачальника, тобто оператора базової мережі. Оператор базової мережі має повний контроль над протоколами попередньому транспортному рівню. Саме із цієї причини інтерфейс повинен пророблятися особливо ретельно. Сервіси мережного рівня будь-якої комунікаційної системи повинні мати наступні основні характеристики:

– сервіси не залежать від канальної технології базової мережі;

– транспортний рівень повинен бути екранований від кількості, типів і топологій різних базових мереж;

– якщо мережні адреси є доступними для транспортного рівня, то вони повинні укладатися в межі стандартної схеми адресації, у якій кожна адреса унікальна.

Розглянемо коротенько функціонування протоколу IP, і проаналізуємо його з погляду середовища виконання транспортного протоколу.

Одержуючи дані від вишележачого рівня у вигляді блоків кінцевого розміру, протокол IP інкапсулює їх у пакет, додаючи до даних користувача свій заголовок. Для нас становить інтерес те, що в заголовку відсутнє поле коду циркулярного контролю. Це пов'язане з тим, що задача перевірки й корекції помилок передачі покладена на канальний рівень, що завжди виконує цю перевірку. Протокол IPv4 здійснював перевірку цілісності даних за допомогою коду контролю парності, але з функціональності IPv6 дана операція була віддалена, оскільки вона дублює аналогічну або могутнішу систему канального рівня.

Закінчивши компонування пакета, IP передає його в мережу на маршрутизатор топологічно більш близький до вузла, якому адресоване повідомлення. Кожний IP пакет у заголовку містить повну адресу відправника й одержувача. Маршрутизатори в складі мережі обмінюються топологічною інформацією за допомогою протоколів маршрутизації й завдяки цьому мають знання топології мережі. Одержуючи IP пакет, кожний маршрутизатор порівнює адресу вузла призначення з наявною в нього таблицею топологічною інформацією й направляє пакет на відповідний вихідний порт. Оскільки різні маршрутизатори можуть мати різне подання про топологію мережі, то й маршрутів між двома вузлами в мережі може бути кілька. Якщо на вихідний порт маршрутизатора надходить більше пакетів в одиницю часу, ніж може його покинути, то пакети організуються в чергу. Оскільки ресурси буферного простору маршрутизаторів кінцеві, то переповнення черги приводить до втрат пакетів.

Виникнення й корекція помилок

Низьке співвідношення енергії сигналу до енергії шуму на лінії приводить до перекручувань прийнятого сигналу і як наслідок до високої ймовірності бітових помилок на приймачі. Помилки каналу передачі даних проявляються як:

1. Вставлені дані: дані, отримані приймачем, але ніколи не передавалися відправником.

2. Загублені дані: дані відправлені, але не дійшли до одержувача (що щезли на фізичному рівні).

3. Дубльовані дані: дані, передані один раз і отримані в декількох екземплярах.

4. Перекручені дані: дані, ушкоджені в транзиті.

5. Розупорядковані дані: дані, послідовність одержання яких не збігається з послідовністю передачі.

Застосовувані на канальному рівні алгоритми розпізнавання й корекції помилок з достатньою ймовірністю дозволяють відобразити помилки типу вставлених і перекручених даних на три типи помилок, що залишилися, а саме втрата, дублювання й розупорядкування даних.

Мережний рівень стека TCP/IP надає користувачеві ненадійний сервіс без установки логічного з'єднання. Сказане означає, що IP пакети можуть почати надходити в мережу без затримки, як тільки інформація готова до відправлення (пакет сформований). Відправлені пакети не підтверджуються одержувачем. Пакет може бути загублений, і ніколи не дійти до одержувача, з кількох причин: він може бути відкинутий з переповненого буфера маршрутизатора внаслідок перевантаження останнього, або дані в складі пакета можуть бути зруйновані в процесі передачі по ненадійному каналі. Таким чином, мережний рівень може привнести додаткові помилки типу втрат даних. Також внаслідок можливості наявності декількох маршрутів до одержувача порядок прибуття пакетів може не збігатися з порядком їхнього відправлення, якщо потік пакетів буде доставлятися по маршрутах з різною затримкою. Більш того, існує можливість доставки одержувачеві декількох копій одного пакета.

Протокол TCP, таким чином, повинен самостійно відслідковувати виникнення трьох типів помилок даних і здійснювати самостійне відновлення в ситуаціях, коли сегмент не приходить взагалі, коли сегменти доставляються мережею не в тому порядку, у якому були відправлені й коли мережа доставляє кілька копій одного сегмента.

Керування потоком

Кожний з нижчележачих рівнів здійснює керування швидкістю передачі даних. Фізичний рівень відповідальний за синхронізацію запису й сканування середовища передачі. Канальний рівень управляє швидкістю передачі пакетів між двома вузлами, застосовуючи схему керування потоком тої або іншої складності, залежно від призначення – Xon/Xoff, Alternating bit [42], slidingwindow[43] і т.п. Мережний рівень також має примітивну схему керування потоком. Маршрутизатор у стані перевантаження може відправити своїм топологічним сусідам повідомлення про наявність перевантаження за допомогою протоколу ICMP. Однак всі алгоритми канального й мережного рівнів здійснюють керування швидкістю потоку лише локально. TCP, як протокол транспортного рівня, здійснює керування потоком по всій довжині логічного каналу між передавальною й приймаючою системами.

У протоколі TCP існує механізм, що обмежує швидкість відправлення сегментів у мережу, для того щоб уникнути переповнення буферів проміжних маршрутизаторів і буфера приймача.

Процедурні правила

Сервіс транспортного рівня реалізується транспортним протоколом між двома об'єктами транспортного рівня. Процедурні правила, необхідні для реалізації цього сервісу занадто громіздки, тому їхній формальний опис наприклад у вигляді кінцевих автоматів був би занадто громіздким. Далі ми дамо опис процедурних правил TCP у вигляді окремих алгоритмів. Окремими важливими аспектами транспортного протоколу, які повинні бути враховані в його процедурних правилах є:

- Адресація й мультиплексування.
- Ініціалізація й закриття зв'язку.
- Буферизація й керування потоком.

Словник транспортного протоколу

Словник найпростішого транспортного протоколу, що забезпечує надійний канал зв'язку й контроль швидкості передачі, повинен дозволяти передавати наступні типи повідомлень [1, 3]:

– Дані (DATA).

– Підтвердження (ACK).

– Розмір прийомного вікна (WND).

Таким чином, мінімальний словник транспортного протоколу, що забезпечує, надійний зв'язок такий: $V = \{DATA, ACK, WND\}$. Якщо розглядати кожний напрямок повнодуплексного транспортного з'єднання роздільно, то дані передаються в напрямку від відправника до одержувача, а підтвердження й відновлення вікна в протилежному напрямку.

Оскільки кожне індивідуальне повідомлення транспортного протоколу інкапсулюється мережним рівнем в окремий пакет, то вигідно об'єднати якнайбільше повідомлень у кожному TPDU. Реально в кожному повідомленні поєднуються всі ці типи. Формат повідомлень транспортного протоколу такий: $\{DATA, SEQ\}\{ACK, SEQ, WND\}$, де SEQ і WND – цілі числа.

Оскільки транспортні з'єднання розраховані на двосторонній обмін повідомленнями, то поєднуються всі поля в єдиному форматі повідомлення: $\{DATA, SEQ, ACK, SEQ, WND\}$. Число SEQ на другій позиції нумерує передані дані, а в третій позиції – підтверджувані дані.

Кодування повідомлень на транспортному рівні

Отже, словник транспортних протоколів складається з повідомлень – так званих TPDU, які інкапсулюють передані дані. Сам TPDU у свою чергу укладений у пакет мережного рівня.

Формат сегмента, як правило, наступний: заголовок фіксованої довжини вирівняний на 32 бітній границі, передує полю даних змінної довжини.

Заголовок сегмента транспортного протоколу містить поле, що дозволяє перевірити цілісність даних і заголовка прийнятого сегмента. У протоколі TCP це

16-ти бітне поле несе перевірочну суму, отриману підсумовуванням по модулю 2 всіх 16-ти бітних слів частини заголовка й поля даних. При одержанні сегмента перевірочна сума обчислюється заново і якщо результат не дорівнює нулю, то сегмент утримуючу помилку відкидається.

Адресація

Для установки зв'язку з віддаленим додатком потрібно вказати його адресу. Адресація необхідна для всіх видів мереж. Існує метод, що дозволяє встановлювати транспортну адресу, по якому додаток-сервер очікує прибуття запиту на з'єднання. Для архітектури TCP/IP це пара: IP адрес й номер локального порту. Об'єкти транспортного рівня допускають використання декількох транспортних адрес, при цьому мультиплексування потоків у межах вузла здійснюється самим транспортним протоколом. Постійно існуючі серверні додатки адресуються прямо, для роботи з додатками, що запускаються лише на час існування сесії, використовується так званий «протокол первісного з'єднання» або інтернет сервер (UNIX inetd) [75].

Установка з'єднання

Без здатності мережі до нагромадження пакетів задача встановлення з'єднань звелася б до двох дій – надіслати запит на з'єднання – дочекатися позитивної відповіді на нього. Насправді ж проблема значно більш складна. Якщо мережа перевантажена, і підтвердження не встигають прибути до відправника вчасно, то й запити на встановлення з'єднання можуть бути відправлені кілька разів. Оскільки маршрутизація кожного пакета відбувається незалежно, то існує ймовірність затримки деяких пакетів протягом більш тривалого часу, ніж інших і, відповідно, збоїв при встановленні з'єднання. Одинична транзакція може, таким чином, випадково відбутися два й більше рази.

Використовуваний в архітектурі TCP/IP метод сполучить ідентифікацію кожного пакета з вимогою обмеження часу життя пакетів у мережі. Причому ідентифікація кожного пакета має на увазі наявність у нього порядкового номера.

У реальності необхідно гарантувати, що не тільки сам пакет зник з мережі, але й все його підтвердження, тому використовується проміжок часу *T*, кратний максимального часу життя пакета. Якщо пройшов час *T* з моменту відправлення пакета, то ми можемо бути впевнені, що ні сам пакет, ні його підтвердження не існують в мережі. Якщо опиратися на таку передумову, то можна розробити надійний спосіб безпечної установки з'єднання. Метод називається «тристоронній обмін» (Three-way handshake). Цей метод не вимагає використання обома сторонами тих самих номерів.

Пристрій А надсилає запит на з'єднання (CR) пристрою В, указуючи при цьому початковий порядковий номер який буде використовуватися транспортним протоколом пристрою А для передачі даних. Пристрій В, одержавши запит, реагує відправленням пакета, що повідомляє про згоду встановити з'єднання, указує свій власний початковий номер *y* і підтверджує номер *x*. Після цього в першому пакеті з даними, що має номер *x*, машина А підтверджує прийом повідомлення з номером *y*.

Схема установки з'єднання, наведена вище, застосовується транспортним протоколом TCP. Альтернативна схема надійної установки з'єднання в умовах наявності затриманих копій повідомлень описана [39].

Розрив з'єднання

Для закриття транспортного з'єднання використовується так званий симетричний розрив, при якому кожний напрямок з'єднання вважається незалежним і закривається окремо. Незважаючи на відносну надійність такого протоколу, він може не спрацювати у випадку втрати початкового CR і всіх N повторних передач. У цьому випадку ініціююча сторона закриє з'єднання, протилежна ж не одержавши інформації про закриття буде активною. Такий стан називається напіввідчинене з'єднання. Для запобігання виникнення таких станів вводиться правило, по якому з'єднання буде автоматично закриватися, якщо протягом певного часу на ньому не зареєстровано ніякої активності.

Якщо ж з'єднання повинне залишатися відкритим протягом тривалого часу, очікуючи появи даних для передачі, то сторонам прийдеться періодично обмінюватися порожніми повідомленнями, що втримують протилежну сторону від закриття.

Керування потоком і буферизація даних

Керування потоком безпосередньо пов'язане з буферизацією. Задачею процесу керування потоком є не допустити переповнення одержувача й проміжних вузлів інформацією, що ті не встигали б обробляти. У тому випадку, якщо мережа надає ненадійний датаграммный сервіс, то відправник повинен буферизовувати відправлені повідомлення на випадок виникнення необхідності повторного відправлення. Одержувач, знаючи, що повідомлення залишаються в пам'яті відправника поки не надійде підтвердження їхнього прийому, може виділяти або не виділяти буферний простір для кожного з'єднання. У випадку буферизації повідомлень в одержувача, що дозволяє істотно підвищити продуктивність системи, виникає питання про схему виділення пам'яті під буфери. Це може бути й система буферів рівних розмірів (якщо всі повідомлення приблизно одного розміру) або великий циркулярний буфер або набір буферів різних розмірів.

Оптимальне рішення відносно буферизації у відправника й одержувача залежить від характеристик потоку. Наприклад, для трафіка породженого інтерактивною роботою з віддаленим терміналом, характеризуємого невеликою швидкістю, краще взагалі не виділяти буферів, а одержувати й передавати інформацію додаткам у міру її надходження (природно, що передане повідомлення повинне залишатися в пам'яті відправника до одержання підтвердження). З іншої сторони для потоку створеного не інтерактивною передачею даних, наприклад – завантаженням віддаленого файлу, ефективність може бути істотно підвищена, якщо одержувач виділить максимальну кількість буферного простору, щоб як можна більше підвищити швидкість передачі.

Таким чином, із часом, відправник і одержувач повинні погоджувати свої схеми виділення буферного простору. Відправник повинен мати можливість запитувати одержувача про виділення певної кількості буферного простору або ж одержувач (не має інформації про потреби відправника) повинен повідомляти, що для з'єднання зарезервовано *x* буферів.

Виділення буферів повинне відбуватися незалежно від приходу підтверджень (на відміну від більшості протоколів канального рівня). Ця схема має на увазі наявність «вікна» змінного розміру. Одержувач виділяє таку кількість буферного простору у відповідь на запит відправника, яке дозволяють його ресурси. Щораз, посилаючи повідомлення, відправник повинен зменшити зазначену кількість і тимчасово припинити передачу, якщо розмір вільного буферного простору в одержувача зменшиться до нуля. Одержувач, у свою чергу повинен указувати поточний розмір вікна й номер підтвердження в потоці, що йде у зворотному напрямку. При збільшенні кількості пам'яті доступної для буферизації мережних з'єднань уже не недолік вільного буферного простору буде обмежувати максимальну швидкість передачі даних, а доступна пропускна здатність мережі. Таким чином, необхідний механізм керування потоком, що враховує також і несучу здатність мережі, а не тільки кількість буферного простору в кінцевих учасників обміну. Якщо використовується механізм обмежуючого вікна, то його розмір повинен відбивати поточну пропускну здатність мережі.

Протокол TCP

Сімейство протоколів TCP/IP містить у собі два протоколи транспортного рівня це TCP – Transmission Control Protocol – і UDP User Datagram Protocol. UDP досить простий, по суті це невелике доповнення до IP, і не використовує віртуальний канал. TCP це надійний протокол, що використовує попередню установку віртуального каналу й доставляє дані користувача без помилок. TCP був спеціально розроблений, щоб забезпечувати надійну передачу байтового потоку від відправника до одержувача через ненадійну мережу. Мережа може

складатися з різних мережних компонентів, що володіють різними топологічними схемами, технологіями, пропускною здатністю, затримками.

Кожний пристрій, що підтримує TCP, містить у собі об'єкт протоколу, що управляє TCP потоками й взаємодіє з рівнем IP. Цей об'єкт приймає потоки даних від локальних процесів, розбиває їх на повідомлення певної довжини (сегменти) і передає сегмент на рівень IP для його передачі у вигляді окремого IP пакета. Після прибуття на адресуєму машину інформація із цих пакетів передається TCP об'єкту, що реконструює оригінальний байтовий потік і передає її користувальницькій програмі. Рівень IP не гарантує доставку пакетів, тому TCP повинен відслідковувати загублені пакети й здійснювати повторне відправлення. Пакети можуть прибувати й у порядку, що відрізняється від порядку, у якому вони були відправлені, тому задачею TCP також є реконструкція порядку в бітовому потоці. Для керування потоком і запобігання перевантажень мережа TCP застосовує механізм ковзного вікна [43].

Модель обслуговування TCP

Доступ до сервісу TCP можна одержати шляхом створення на кінцевих машинах точок доступу (Sockets). Кожна така точка має адресу, що складається з IP адреси машини й 16-ти бітного номера, унікального в межах пристрою, що ідентифікує порт. TCP з'єднання встановлюється між двома точками доступу й пари транспортних адрес однозначно ідентифікує з'єднання. Кілька з'єднань можуть кінчатися на одній точці доступу. Порти з номерами нижче 256 однозначно відповідають набору конкретних додатків. Порти з іншими номерами виділяються додаткам динамічно. Всі TCP з'єднання повнодуплексні й мають тип точка-точка. З'єднання являє собою неструктурований байтовий потік, тобто границі між повідомленнями не зберігаються.

Коли додаток передає дані TCP, протокол може або відразу здійснити їхнє відправлення, або нагромадити достатня їхню кількість у буфері. Якщо ж додатку необхідно послати інформацію миттєво, то він може використовувати прапор PUSH. Крім того, існує можливість пересилати термінову інформацію з

використанням прапора URGENT. Коли додаток установлює цей прапор, то TCP передає всі наявні дані без зволікання, крім того, TCP одержувача генерує переривання для додатка якому призначається термінова інформація й указує на положення її в отриманому потоці.

Кожний байт TCP з'єднання має унікальний 32-х бітний порядковий номер. Ці номери використовуються не тільки для підтверджень, але й для механізму керування вікном. Об'єкти TCP обмінюються інформацією у вигляді сегментів. Сегмент складається з 20-ти байтного заголовка (+ необов'язкова частина) і нуля або більше байтів даних. Протокол сам приймає рішення щодо розміру сегмента. Існують два обмеження розміру сегмента. Перше: сегмент повинен уміщатися в межах максимального розміру поля корисного навантаження IP – 2^{16} байт. Друге: кожна мережна технологія має так звану максимальну одиницю передачі (Maximum Transfer Unit – MTU) у яку й повинен уміститися повний IP пакет, що включає в себе IP і TCP заголовки й дані.

Одночасно з передачею сегмента запускається таймер повторної передачі. Коли сегмент прибуває до одержувача, TCP об'єкт останнього відправляє сегмент, що містить підтвердження прийому цього сегмента. Якщо таймер повторної передачі для сегмента спрацьовує завчасно приходу підтвердження, то даний сегмент ретранслюється.

Формат заголовка TCP

Заголовок сегмента TCP містить наступні поля:

– Порядковий номер, ідентифікує перший байт даних у цьому пакеті, може також використовуватися для вказівки першого номера в серії який буде використовуватися в наступних передачах.

– Номер підтвердження – містить порядковий номер наступного байта даних, що очікується на приймаючому пристрої.

– Прапори – різноманітна контрольна інформація: URG – термінові дані, покажчик на термінові дані в цьому випадку вказує на їхнє положення в сегменті: ACK – 1 виходить, що поле підтвердження містить правильну інформацію; PSH –

одержувач повинен доставити дані додатку якомога швидше; RST – з'єднання повинне бути закрите; SYN – використовується для установки з'єднання, запит на установку містить SYN=1, ACK=0, відповідь на цей сегмент містить SYN=1, ACK=1; FIN – відправник сигналізує про бажання закрити з'єднання. Після відправлення FIN машина може ще необмежено довго продовжувати одержувати інформацію. SYN і FIN сегменти мають порядкові номери, що гарантує їхню обробку в правильному порядку.

Розмір вікна повідомляє про те, як багато байтів можна відправити, починаючи з останнього підтвердженого байта. Цей розмір може бути дорівнює нулю.

Керування з'єднанням у TCP

Установка з'єднання в TCP використовує процедуру тристороннього обміну. Для відкриття з'єднання пасивна сторона виконує примітиви LISTEN і ACCEPT, а інша відкриває з'єднання в активному режимі, виконуючи примітив CONNECT. Коли запит на відкриття з'єднання, прибуває до одержувача, той перевіряє, є чи процес зареєстрований для прийому з'єднань по зазначеному порту. Якщо такий процес існує, то з'єднання встановлюється.

Хоча TCP з'єднання полнодуплексні, закриваються з'єднання в кожному напрямку окремо. У нормальній ситуації потрібно чотири сегменти, щоб повністю закрити з'єднання (два FIN і два ACK). По кожному із симплексних з'єднань дані можуть передаватися необмежено довго. Для запобігання появи напіввідчинених з'єднань використовуються кілька таймерів. Якщо відповідь не сегмент FIN не приходить протягом двох максимальних періодів життя пакета, відправник сегмента FIN розриває з'єднання.

Керування передачею в TCP

Керування вікном у TCP не прив'язано прямо до підтверджень (як у багатьох протоколах канального рівня).

Підтвердження, використовувані протоколом TCP, є кумулятивними, тобто кожне підтвердження відповідає останньому отриманому в правильному

порядку байту потоку. Прихід підтвердження з номером N означає, що всі байти [0,..,.N-1] були успішно доставлені одержувачеві.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium, використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

− Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких

же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4 забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні 64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш

стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

– Велика кількість поліпшень STL від Dinkumware.

– Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.

– Поліпшена підтримка Cmake.

– Велика кількість виправлень для підвищення стабільності і якості.

– Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.

– Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє

використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

– Додана підтримка Metal Driver GPU для macOS і iOS.

– Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.

– Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

– Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.

– Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.

– Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки.

Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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

Алгоритм ARTCP

Запропонований у даній роботі, удосконалений протокол Adaptive Rate Transmission Control Protocol (ARTCP) системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем, позичає деякі механізми від протоколу TCP. В ARTCP повністю переглянутий алгоритм керування потоком, що і відрізняє його від TCP. Разом з тим, запропонований протокол може забезпечувати сумісність із TCP.

Аспекти новизни протоколу ARTCP

ARTCP відрізняється від стандартного TCP тим, що сегменти відправляються в мережу не у вигляді сплеску в межах вікна, а розділені часовими проміжками, тривалість яких визначається поточним значенням швидкості. Швидкість потоку регулюється не розміром змінного вікна, а значенням швидкості, зміною якої здійснюється адаптація алгоритму відповідно до умов. Механізм ковзного вікна в ARTCP застосовується тільки для запобігання переповнення буферів одержувача. На розмір вікна в ARTCP не накладено ніяких обмежень у жодному з режимів роботи. Вікно обмежене лише наявністю буферного простору в одержувача, тому для одержувача рекомендується встановлювати вікно на максимальний розмір.

У випадку, коли одержувач обмежений у буферному просторі, ARTCP буде поводитися як звичайний протокол TCP, обмежений вікном. Таким чином, протокол ARTCP сполучить у собі метод ковзного вікна для регулювання керування потоком від краю до краю й метод контролю швидкості для підстроювання під ПрЗд проміжних вузлів з'єднання.

Другою найважливішою відмінністю ARTCP є те, що як сигнал про стан перевантаження або наявності додаткових ресурсів у мережі використовуються темпоральні характеристики потоку – вимір шпаруватості потоку в одержувача й зміни часу RTT. Втрата пакета ніяк не відбивається на роботі ARTCP крім здійснення ретрансляції загубленого пакета механізмом корекції помилок. Як і в TCP об втрати пакета повідомляють дві можливих події: спрацьовування ТПП або послідовне одержання двох підтверджень тих самих даних.

Таким чином, у порівнянні зі своїм попередником, ARTCP має наступні переваги:

1. ARTCP не потрібно доводити мережа до стану перевантаження, щоб визначити доступну частку ПрЗд, тому виключені втрати пакетів пов'язані із цим процесом.

2. ARTCP істотно знижує вимоги до міжмережних пристроїв. По-перше, для нормального функціонування даного протоколу потрібен менший об'єм буферного простору, ніж для TCP, оскільки режим передачі є згладженим. По-друге, ARTCP не вимагає й не залежить від наявності яких або механізмів диспетчеризації або керування чергами, таких як RED або WFQ.

3. ARTCP не інтерпретує втрату пакета як ознаку перевантаження мережі, використовуючи замість цього темпоральні характеристики потоку. Тому ARTCP повинен особливо ефективно працювати в системах бездротового зв'язку, там де використання TCP неефективно.

4. На відміну від TCP новий протокол не покладається цілком на потік підтверджень у зворотному напрямку для синхронізації процесу передачі. У зв'язку із цим можлива реалізація ARTCP з меншою частотою підтверджень, що не обмежувала б швидкість в асиметричних системах.

Евристика в основі алгоритму ARTCP

Аналіз робіт в області транспортних протоколів, дозволив укласти, що недоліки протоколу TCP досить істотні і є наслідком самого алгоритму, що

лежить в основі протоколу TCP. Тому модифікація TCP без заміни його основних алгоритмів не може привести до істотного поліпшення характеристик протоколу.

Тому в цій роботі було вирішено створити вдосконалений алгоритм транспортного протоколу, що залишається, однак, повністю сумісним з архітектурою TCP/IP.

Для того щоб усунути недоліки, властиві TCP, необхідно було знайти спосіб одержання інформації про стан мережі, відмінний від застосування в цих цілях втрат сегментів. Найбільше добре на роль індикатора стану мережі підходять часові характеристики потоку: час RTT і міжсегментні інтервали. З використанням міжсегментних інтервалів можна також визначити частку ПрЗд каналу. Для цього потрібно запам'ятовувати міжсегментні інтервали потоку у відправника й вимірювати їх в одержувача. Порівняння значень інтервалів характеризує стан мережі, а мінімальне значення вимірюваних інтервалів в одержувача дозволяє визначити доступну частку ПрЗд.

Таким чином, виходить наступна схема: установка швидкості потоку відправником за допомогою ретельної диспетчеризації сегментів, вимір швидкості прибуття потоку в одержувача й передача цієї інформації відправникові разом з іншою контрольною інформацією. Різниця старого й нового значень швидкості відправлення потоку ARTCP на кожному кроці задається випадковою змінною, однак, при наявності сигналу про перевантаження мережі ймовірність зниження швидкості перевищує ймовірність її збільшення на кожному новому кроці.

Параметри й змінні

Нехай τ часовий інтервал між послідовними трансляціями пакетів. Задача функції диспетчеризації сегментів у тому, щоб затримувати відправлення чергового сегмента на час τ після початку передачі попереднього сегмента. Позначимо всі змінні, відносні до відправника індексом s, і r – відносні до одержувача. Отже, τ часовий інтервал між моментами початку відправлення в

мережу сегмента i+1 і *i-го*, а τ_r – інтервал між послідовно прибулими до одержувача сегментами.

Нехай швидкість каналу зв'язку, безпосередньо до якого підключений відправник – R_{ls}, тоді час який потрібен на відправлення одного сегмента (з моменту початку передачі до моменту її закінчення) $t_{ls} = S / R_{ls}$, де s – розмір переданого сегмента.

Очевидно, що максимально можлива швидкість потоку:

$$R^{max}_s = R_{ls} = S / \tau^{min}_s. \qquad (3.1)$$

Мінімальне значення міжсегментного інтервалу в цьому випадку буде $\tau^{min}_s = t_{ls}$, коли пакети відправляються в мережу без затримок з максимальною швидкістю канального рівня. Шляхом зміни τ у межах $[\tau^{min}, \infty)$, ARTCP може контролювати швидкість потоку в межах $[R_{ls}, 0)$. Ілюстрація принципу керування швидкістю наведена на рисунку 3.1.

Затримка відправлення готових сегментів виробляється за допомогою системного таймера. Наприклад, для повного використання ПрЗд каналу в 512 Кб/с при розмірі сегмента в 1000 байт кожний сегмент необхідно відправляти із затримкою τ =0.015625 с, щоб швидкість потоку склала 64 пакета/с.

У таблиці 3.1 наведений список параметрів і змінних використовуваних алгоритмом керування потоком протоколу ARTCP.

Формат повідомлення

Формат повідомлень використовуваних ARTCP може в точності збігатися з форматом пакета TCP. Стандарт TCP [4] передбачає наявність додаткових полів у заголовку сегмента між стандартним заголовком і полем даних. ARTCP може передавати додаткову інформацію в цих полях, що буде гарантувати сумісність із TCP. Усього протокол ARTCP вимагає використання лише двох нових полів: значення попереднього порядкового номера "PS" у напрямку від відправника до одержувача й значення шпаруватості "TI" у напрямку від одержувача до відправника. Значення "TI" можна передавати у вигляді опції часової мітки [6], а значення "TI" вимагає поля, що дозволяє помістити порядковий номер сегмента.

Таблиця 3.1 – Список параметрів і змінних

s	Розмір кадру канального рівня утримуючий сегмент
τ_s	Часовий проміжок між послідовними трансляціями сегментів (від першого біта до першого біта)
τ_R	Часовий проміжок, обмірюваний між послідовними моментами прибуттями сегментів (від останнього біта до останнього біта)
$R_S(t)$	Швидкість потоку, установлювана відправником
$R_R(t)$	Швидкість потоку, що обчислюється одержувачем
$R_e(t)$	Оцінка доступної пропускної здатності в момент t-*RTT*
R'_S	Перша похідна швидкості за часом, установлюється відправником
R_{pe}	Значення оцінки доступної пропускної здатності в момент *i-1*
A_c	Площа області компенсації
SSGR	Параметр експонентного росту R'_S у режимі SS
RTT	Часовий проміжок між моментом відправлення сегмента й моментом приходу його підтвердження
ERTT	Зважене ковзне середнє RTT
SmoothedRe	Зважене ковзне середнє Re
MaxERTT	Максимальне значення згладженого RTT
MinERTT	Мінімальне значення згладженого RTT
MDFACTOR	Коефіцієнт, використовуваний при мультиплікативному зниженні $R_s(t)$
R_S^{amd}	Значення швидкості відправлення даних безпосередньо на виході режиму MD1
R_S^{bmd}	Значення швидкості відправлення даних безпосередньо перед входом у режим MD1
R_e^{amd}	Значення оцінки доступної пропускної здатності в режимі MD1
speedup	Коефіцієнт, що визначає ймовірність збільшення швидкості в режимі FT
slowdown	Коефіцієнт, що визначає ймовірність зниження швидкості в режимі FT
Midpoint	Точка відліку швидкості на кожному кроці в режимі FT
sR_S	Зважене ковзне середнє значення R_s
K	Коефіцієнт критерію здійснення переходу
BER	Bit error ratio – резидентна ймовірність інвертування біта на лінії
R_S^{min}	Початкове мінімальне значення швидкості, з якого починається ріст у стані SS

Диспетчеризація сегментів

Диспетчер сегментів відправляє сегменти на лінію через строго задані міжсегментні часові інтервали. Значення інтервалів визначаються швидкістю відправлення потоку, що задається функцією адаптації.

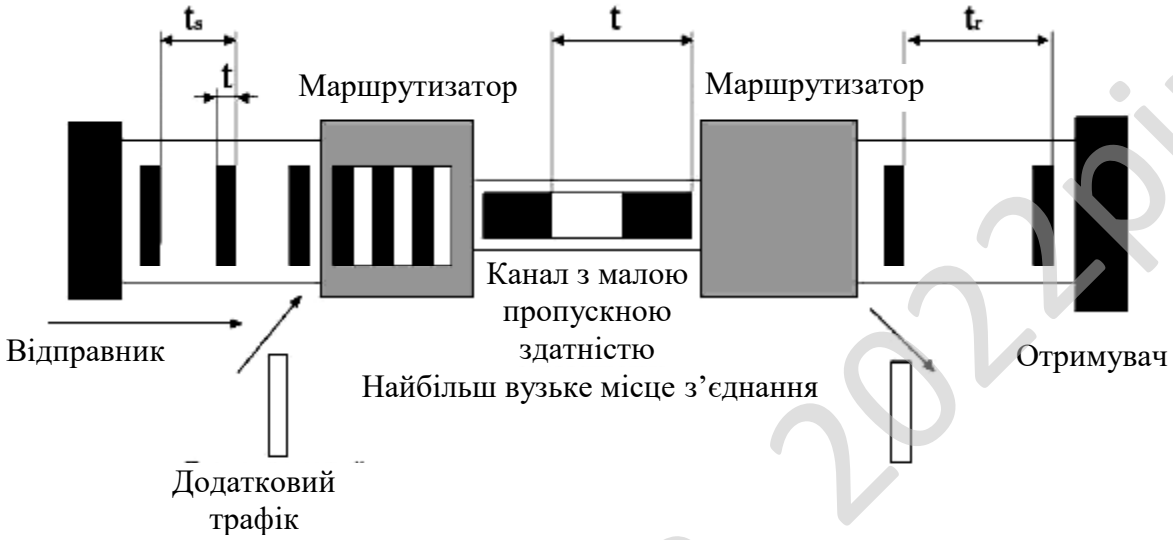

Рисунок 3.1 – Диспетчеризація сегментів і вимір швидкості потоку алгоритмом ARTCP

Чорні прямокутники позначають сегменти одного з'єднання. Розходження ширини прямокутників відбиває розходження швидкостей каналів. Меншій швидкості відповідає більш тривалий час передачі.

Вимір швидкості

Очевидно, що швидкість прийому потоку одержувачем не може бути вище швидкості обслуговування потоку на ділянці з найменшої ПрЗд, через яку проходить з'єднання. Таким чином, знаючи швидкість прибуття потоку до одержувача, можна визначити доступну пропускну здатність мережі. Для коректного виміру швидкості необхідно не враховувати випавші з потоку, тобто загублені сегменти, а також сегменти, що доставляються мережею в зміненому порядку. Для виконання цієї умови в поле "PS" кожного сегмента, що відправляється, записується порядковий номер (або зсув) від попереднього сегмента.

Одержавши сегмент *i*, одержувач обчислює різницю поточного часу й часу прибуття попереднього (*j*) сегмента τ_R і у випадку, якщо поле "PS" *i-го* сегмента містить значення *j*, поміщає $R_r = s/\tau_R$ у поле "TI" підтвердження наступного в протилежному напрямку (рисунок 3.1). Одержувач витягає значення поля "TI" з одержуваних підтверджень і використовує його для керування швидкістю передачі.

Функція адаптації

Опис алгоритму роботи функції адаптації, безпосередньо здійснюючe керування потоком ARTCP, було дано в роботах [52,53,54,76].

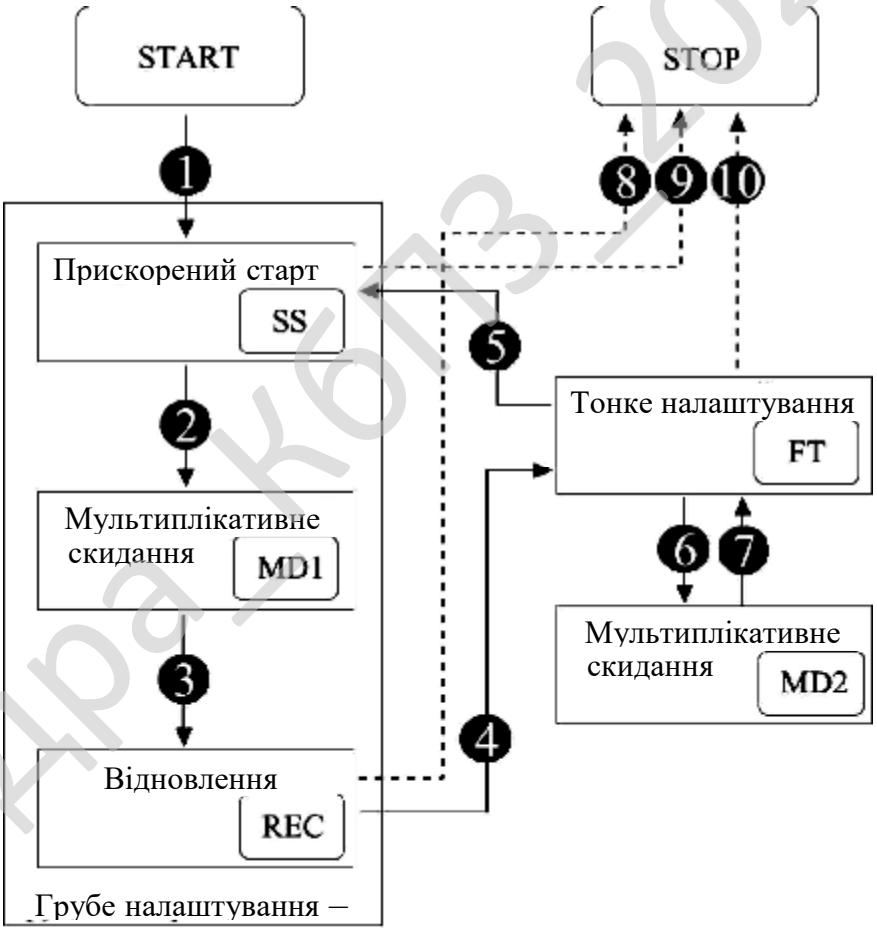

Рисунок 3.2 – Діаграма режимів алгоритму адаптації протоколу ARTCP. Штриховані лінії позначають можливі переходи в стан зупинки системи

Спосіб керування потоком повинен досягти, по-перше, швидкої реакції потоку на умови, що змінюються, з'єднання й, по-друге, стабілізувати швидкість передачі, коли вона дорівнює максимальної швидкості мережі. Алгоритм керування потоками ARTCP функціонує в декількох режимах.

Робота ARTCP починається з режиму швидкого збільшення швидкості, аналогічної механізму вповільненого старту стандартного TCP, для максимально швидкого досягнення з'єднанням верхньої межі доступної пропускної здатності. Після того, як верхня межа досягнута, алгоритм ARTCP переходить у режим точного налаштування, протягом якої втримує швидкість на рівні доступної ПрЗд. У випадку визначення зменшення доступної ПрЗд, ARTCP робить мультиплікативне зниження швидкості, що у випадку тривалого стану перевантаження триває експоненційно. Отже, адаптація швидкості передачі потоку протоколом ARTCP відбувається в п'яти режимах (рисунок 3.2).

Режим прискореного старту (*SS*) має мету максимально швидко збільшити швидкість потоку від мінімального значення до значення, рівного або переважаючого ПрЗд каналу відразу після ініціалізації з'єднання. Для цього швидкість збільшується експоненційно:

$$R'(t+RTT) = R_S(t) \times SSGR, \tag{3.2}$$

$$R_S(t_i) = R_S(ti\text{-}1) + RS(t,) \times (t_i - ti\text{-}1). \tag{3.3}$$

Початкове значення швидкості встановлюється після синхронізації з'єднання як:

$$R_S^{\min} = \frac{SEGSIZE}{\min RTT}. \tag{3.4}$$

Вихід з режиму ***SS*** відбувається, коли $R_e(t_i) < (1-\varepsilon) \times R_S(t_i - \mathrm{RTT})$. Після реалізації переходу 2, алгоритм переходить у стан мультиплікативного скидання **MD1**.

Режим мультиплікативного скидання (MD1) слідує за режимом ***SS***. Після виходу з ***SS*** значення $R_s(t)$ буде перевищувати $R_e(t)$, тому в режимі **MD1** швидкість потоку стрибкоподібно встановлюється свідомо нижче $R_e(t)$:

$$R_S(t_i) = R_e(t_i) - MDFACTOR \times (R_S(\mathrm{ti}\text{-}1) - R_e(t_i)) \tag{3.5}$$

Після зниження швидкості алгоритм переходить у режим відновлення.

Режим відновлення (REC) має на меті, лінійно збільшуючи швидкість, довести її до вже відомого значення ПрЗд каналу: $R_e(t)$, компенсуючи виниклу в режимі ***SS*** перевантаження. У режимі **REC** обчислюється значення площі області компенсації $A_c(t_t)$ як площі фігури, утвореної значеннями $R_s(t)$ над прямій $R_e(t_i)$ за час, поки $R_s(t_t) > R_e(t_t)$ у режимі ***SS***:

Значення площі області компенсації дорівнює сумі площ набору трапецій утворених значеннями $R_s(t)$ над прямій $R_e(t_i)$. Площа кожної трапеції:

$$S_T = \Delta t_i / 2 \ ((2R_S(t_i) - R'(t_i) \times \Delta t_i - 2R_e(t_i)), \tag{3.6}$$

де Δt_i час, протягом якого не відбувалося змін значення $R'(t_i)$.

$$\begin{aligned} A_c(t_i) = {} & \frac{1}{2}\Delta t_0 \left[2R_s(t_i) - \Delta t_0 R_s'(t_i) - 2R_e(t_i)\right] + \\ & + \frac{1}{2} RTT \left[2R_s(t_i) - \frac{R_s'(t_i)}{SSGR} - RTT\,\frac{R_s'(t_i)}{SSGR} - R_e(t_i)\right] + \\ & + \frac{1}{2} RTT \left[2R_s(t_i) - \frac{R_s'(t_i)}{SSGR^2} - RTT\,\frac{R_s'(t_i)}{SSGR^2} - R_e(t_i)\right] + \ , \\ & + \ldots + \\ & + \frac{1}{2} \frac{\left[R_s(t_i) - \frac{R_s'(t_i)}{SSGR^N} - R_e(t_i)\right]^2}{\frac{R_s'(t_i)}{SSGR^N}} \end{aligned} \tag{3.7}$$

де Δt_0 проміжок часу між моментом t_i і моментом попередньої зміни $R'(_i)$ і N – індекс доданка, при якому:

$$R_s(t_i) - \frac{R_s'(t_i)}{SSGR^N} > R_e(t_i). \tag{3.8}$$

Отже, перший доданок наведеної формули є площа трапеції з висотою меншої RTT, останній доданок – площа трикутника, що складаються від 2 до N-1 площі трапецій з висотою рівної RTT.

Наприклад, у випадку, наведеному на рисунку 3.3, $A_c(t_i)$ дорівнює сумі площ трапеції DCBE і трикутника ABE. Δt_0 дорівнює відрізку ED.

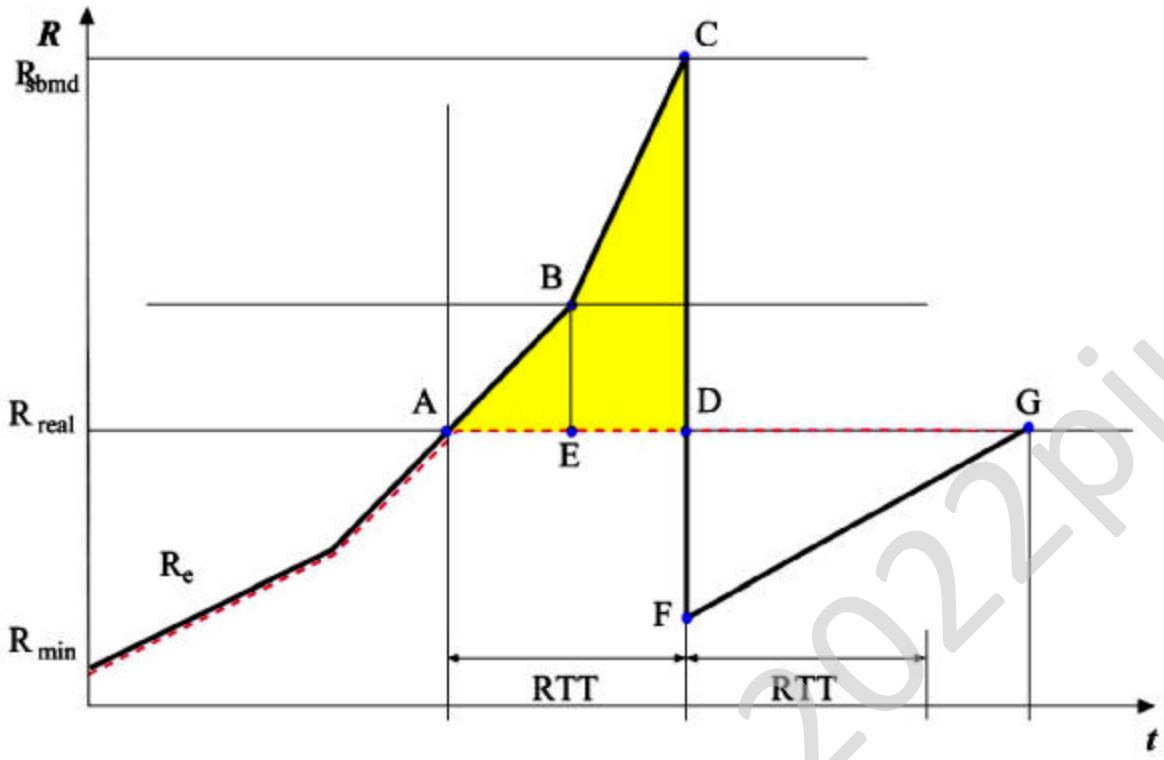

Рисунок 3.3 – Залежність швидкості від часу у фазі грубого налаштування ARTCP

Зафарбована область, що складається з ABE і BCDE є площа області компенсації й виражає об'єм даних, що нагромадилися в буфері. Штрихова лінія позначає поточну оцінку ПрЗд мережі протоколом ARTCP.

Значення $A_c(t_i)$ застосовується для визначення величини $R'_s(t_i)$ у стані відновлення **REC**. Ідея в тому, що із-за затримки інформації про стан мережі на час RTT, у стані швидкого збільшення швидкості відправлення сегментів потік викличе наповнення буферів мережі. Пакети будуть накопичуватися в мережі протягом часу, коли швидкість відправлення сегментів перевищує ПрЗд мережі (відрізок AD на рисунок 3.3). Перебування з'єднання в стані відновлення необхідно для того, щоб мережа впоралася з виниклої до зменшення швидкості відправлення перевантаженням. Очевидно, що кількість даних, що нагромадилися в буферах мережі, визначається площею області $A_c(t_i)$, тому швидкість відправлення сегментів у стані **REC** повинна бути знижена таким

чином, щоб площа фігури DFG була дорівнює $A_c(t_i)$. Швидкість у стані **REC** міняється лінійно, обумовлена значенням:

$$R_s^{\cdot}(t_i) = \frac{\left[R_{pe}(t_i) - R_s(t_i)\right]^2}{2A_c(t_i)}. \quad (3.9)$$

У стані **REC** швидкість відправлення даних зростає лінійно від значення отриманого в попередній стадії **MD1** за законом:

$$R_s(t_i) = R_s^{amd} + t \times R_s^{\cdot}(t_i). \quad (3.10)$$

Вихід зі стану **REC** (перехід 4) здійснюється в тому випадку, коли:

$$R_s(t\ ) \geq R_e^{amd}. \quad (3.11)$$

Режим тонкого налаштування (**FT**) слідує за режимом **REC**. У режимі **FT** швидкість відправлення даних повільно підбудовується під ПрЗд каналу. Відношення коефіцієнтів *speedup* і *slowdown* у стані **FT** визначає ймовірність зниження або підвищення швидкості на кожному кроці. Коефіцієнт *speedup,* відповідальний за підвищення швидкості обернено пропорційний швидкості даного з'єднання. Коефіцієнт *slowdown,* відповідальний за зниження швидкості, пропорційний відношенню вимірюваного RTT до мінімального значення RTT. Значення *speedup* більше при менших значеннях $R_s(t)$, що дає повільним з'єднанням перевага для одержання доступу до більшої відносної частки ПрЗд. Значення *slowdown* однаково для всіх з'єднань і росте при росту RTT. Таким чином, імовірність підвищення швидкості для повільних з'єднань більше, а ймовірність зниження швидкості однакова для всіх з'єднань. Вихід з режиму **FT** відбувається у випадку стрибкоподібної зміни вимірюваного RTT.

У стані **FT** значення швидкості передачі визначаються за законом:

$$R_s(t_i) = midpoint + \left[2 \times Rand - \frac{slowdown}{speedup}\right] \times \frac{speedup}{slowdown} \times INTERVAL \times midpoint, \quad (3.12)$$

де *rand* – рівномірно розподілена випадкова величина, генеруєма функцією drand48 з областю значень [0; 1]. Після влучення в стан **FT** (реалізації переходу 4) значення *midpoint* установлюється рівним R_e^{amd}, надалі значення *midpoint*

установлюється рівним sR_s. Змінні *speedup* і *slowdown* визначають напрямок зміни швидкості відправлення даних залежно від зміни часу RTT.

Коефіцієнти *slowdown* і *speedup* для використання у формулі (3.12) визначаються по наступних формулах:

$$speedup = \left[1 + \frac{S}{\min RTT \times sR_s}\right]^2, \qquad (3.13)$$

де другий доданок являє собою відношення мінімальної кількості даних у транзиті по з'єднанню (*s* байт за час RTT) до реальної їхньої кількості (добуток мінімального часу RTT на середню швидкість відправлення потоку). Відповідно ріст реальної швидкості потоку виражається в зменшенні значення коефіцієнта *s*peedup, таким чином, за інших рівних умов імовірність росту R_s для з'єднання з меншою швидкістю буде більше. За рахунок цього усувається нерівномірність використання ресурсів різними з'єднаннями.

Коефіцієнт *s*lowdown обчислюється в такий спосіб:

Якщо $RTT > minERTT*(1+PRECISION)$,

те $slowdown = 2 \times (RTT/minERTT) \times speedup$

інакше

$slowdown = 1$

Відношення *speedup/slowdown* визначає знак відхилення миттєвого значення швидкості від середнього. Якщо *speedup* > *slowdown* то відхилення від середнього значення для миттєвого значення швидкості буде позитивним, тобто швидкість потоку буде збільшуватися. У випадку *speedup* < *slowdown* швидкість потоку буде знижуватися. Також, у стані **FT** максимальне відхилення миттєвого значення швидкості відправлення пакетів від середнього за попередній період, відповідно до формули (3.12), пропорційно середньому значенню швидкості. У зв'язку із цим потік, роблячи перехід 4 у стані **FT** при більшому значенні оцінки доступної ПрЗд, пристосовується до невеликих змін ПрЗд більш інтенсивно.

Умовою переходу з **FT** у режим мультиплікативного скидання **MD2** (перехід 6) є:

$$ERTT > K \times (FT \max RTT - \min ERTT)/ \quad (3.14)$$

Режим мультиплікативного скидання (**MD2**) необхідний для швидкого зниження швидкості за умови різкого росту RTT*:*

$$midpoint_i = midpoint_{i-1} \times MDFACTOR \quad (3.15)$$

Після цього протокол переходить у стан **FT**, реалізуючи перехід 7. У тому випадку, якщо умова (3.14) продовжує залишатися дійсною, то мультиплікативне зменшення триває, оскільки послідовність переходів 6 – 7 реалізується неодноразово, виражаючись в експонентному зменшенні швидкості передачі даних.

Завершення роботи протоколу може відбутися з будь-якого стану {***SS***, **REC**, **FT**} – переходи (8, 9, 10).

Очікуване поводження алгоритму керування швидкістю потоку в різних режимах зображено на рисунку 3.4.

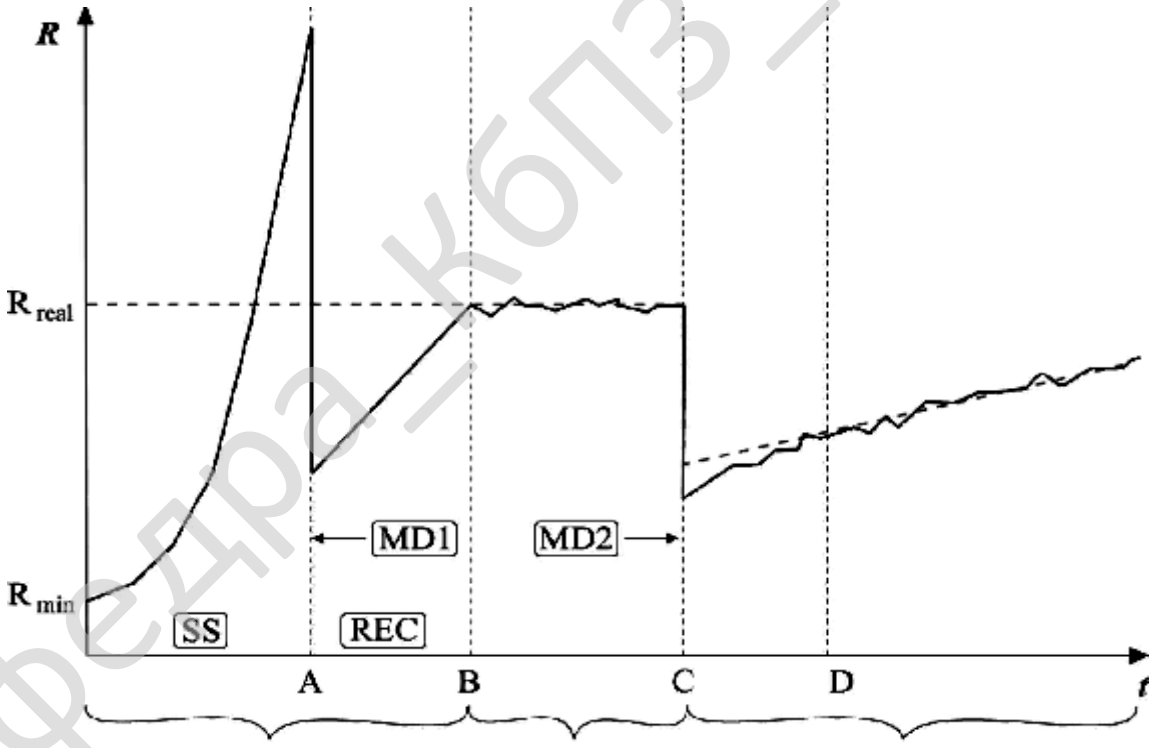

Рисунок 3.4 – Очікуване поводження алгоритму керування швидкістю потоку (залежність швидкості від часу)

Значення t у точках A, B, C позначають моменти переходу в новий режим.

Сумісність із TCP

Система, що підтримує ARTCP, може бути також сумісна з TCP. Для цього, ініціатор з'єднання, що підтримує ARTCP, поміщає в заголовку синхронізуючого пакета опцію "PS". Наявність поля "PS" у заголовку SYN пакета повинне включати механізм ARTCP одержувача. Якщо така можливість є, то одержувач включає в сегмент SYN-ACK поле "TI", якщо ні, то відсутність опції "TI" у відповіді одержувача відключає механізм ARTCP в ініціатора й подальший обмін відбувається по протоколі TCP. Якщо ж опція "TI" є присутнім у відповіді, то ініціатор упевнений, що й одержувач задіяв механізм ARTCP.

Порівняння ARTCP і TCP на основі аналізу алгоритму

Протокол TCP не може обходитися без втрат сегментів, які створюються самим протоколом шляхом переповнення черги, і які обмежують подальший ріст швидкості TCP.

Теорема: Протокол ARTCP не створює втрат сегментів, якщо $\frac{Q_{\max}}{R} > \varepsilon$, де Q_{max} – максимальна довжина черги, ε – граничне значення, використовуване алгоритмом ARTCP, а R – швидкість каналу.

Доказ:

Втрата сегмента може відбутися лише у випадку надходженні чергового сегмента в чергу обслуговуючого пристрою, коли розмір сегмента перевищує різниця $Q_{max} - Q$.

Якщо D сума затримок передачі в каналах на шляху від відправника до одержувача, то в припущенні, що канали в системі симетричні й трафік передається лише в одному напрямку, мінімальний час *min*RTT визначається як $2D$. У випадку якщо середня швидкість відправлення потоку R_s перевищує швидкість R каналу, що обслуговує чергу, виникає черга сегментів у маршрутизаторі. Поява черги довжини Q приведе до збільшення часу RTT на

величину *Q/R*. Однак для ARTCP, якщо різниця RTT-minRTT перевищує деяке граничне значення *ε*, імовірність зниження швидкості відправлення потоку R_s стане відмінної від нуля й швидкість потоку буде знижуватися, поки значення дозволеного перевищення RTT над *min*RTT не стане нижче граничного значення. Таким чином, виконується нерівність: $Q < \varepsilon \times R$. Вибираючи досить мале значення ε, алгоритм ARTCP гарантує, що довжина черги не перевищить певного значення, меншого, ніж максимальна довжина черги $Q < Q_{max}$. Для виконання цього, необхідно вибрати *ε*, так, щоб $\varepsilon < \frac{Q_{max}}{R}$. Отже, при такому виборі граничного значення, відсутність втрат сегментів при роботі ARTCP гарантовано.

Наслідок з теореми: при числі потоків більшому 1, протокол ARTCP на відміну від TCP, може бути настроєний так, щоб взагалі не створювати втрат сегментів.

Для протоколу TCP факт втрати сегмента служить індикатором виникнення перевантаження мережі. Значення ймовірності втрати сегмента визначає швидкість передачі, яка розвивається TCP з'єднанням. Будь-яка втрата сегмента викликає стрибкоподібне зменшення розміру вікна й, отже, зниження швидкості передачі TCP. В умовах, коли причиною втрат є винятково переповнення черги, зниження швидкості TCP при виникненні втрат приводить до того, що виконується рівність середньої швидкості TCP і швидкості обслуговування каналу. Очевидно, що зниження швидкості внаслідок додаткових втрат, викликаних помилками передачі, приведе до того, що середня швидкість TCP потоку стане менше, ніж швидкість каналу. Оскільки TCP не може визначити причину втрати сегмента й реагує зниженням швидкості на будь-яку втрату, те його ефективність у мережах, де втрати сегментів можуть бути наслідком помилок передачі, буде тим менше, ніж вище ймовірність втрати сегмента.

Протокол ARTCP на відміну від TCP не знижує швидкість передачі потоку при виникненні втрати сегмента. Загублені дані ретранслюються, не

роблячи впливу на швидкість передачі. Внаслідок цього, втрати сегментів не роблять впливу на швидкість потоку ARTCP.

Сказане вище можна сформулювати в наступному виді:

Властивість: На відміну від TCP, протокол ARTCP не чутливий до втрат сегментів.

Напрямки подальшого розвитку ARTCP

Протокол ARTCP, запропонований у цій роботі, здатний працювати більш ефективно і якісно, ніж TCP, однак можна виділити кілька напрямків подальших досліджень нового протоколу, які можуть, по-перше, дати можливість ефективно використовувати його в асиметричних системах, а по-друге, досягти рівноправності між потоками з різною довжиною маршруту.

Асиметричні системи

Оскільки в протоколі ARTCP усунута ACK-синхронізація, властива TCP, то відправлення сегментів відбувається незалежно від прибуття підтверджень аж до вичерпання максимального вікна, то на відміну від TCP, ARTCP може бути вдосконалений так, щоб ефективно працювати в системах з асиметричними каналами.

Для використання ARTCP у таких системах необхідно зменшити частоту підтверджень. Оскільки штучна затримка підтверджень викличе збільшення затримки в петлі зворотного зв'язку, то, вимір затримки передачі сегментів потрібно також зв'язати з одержувачем. Оскільки важко домогтися гарної синхронізації системних годин одержувача й відправника, то одержувач може лише помічати зміну часу передачі сегментів, якщо відправник використовує стандартне поле часової мітки [6]. Якщо різниця значень мітки в потоці й системних годинах одержувача змінюється, значить змінюється й абсолютне значення затримки. У цьому випадку одержувач повинен збільшити частоту підтверджень, щоб відправник міг зреагувати на зміну навантаження в мережі. Коли значення швидкості прибуття потоку й затримки передачі не міняються, частота підтверджень може бути знову зменшена.

Алгоритм ARTCP не містить перешкод для цього вдосконалення, крім того, модифікований таким способом ARTCP для асиметричних каналів збереже сумісність із представленим тут протоколом.

З'єднання з різними RTT

Для з'єднань, що володіють різним часом RTT, через розходження довжин їхніх маршрутів, середнє значення коефіцієнта F обмежується деяким числом, меншим 1, як показано в роботах [80,81,82,83]. Це вірно як для ARTCP, так і для TCP. Щоб досягти рівноправності поділу ПрЗд між ARTCP потоками з різною довжиною маршруту, необхідно усунути залежність коефіцієнта *speedup* від мінімального часу RTT. Цього результату можна досягти шляхом модифікації алгоритму адаптації таким чином, щоб у режимі **FT** повністю відмовитися від використання обмірюваного RTT як індикатора перевантаження, використовуючи лише значення міжсегментних інтервалів потоку.

Імітаційна модель

Для дослідження можливостей протоколу й відпрацьовування його механізмів була розроблена програмна імітаційна модель самого протоколу й мережних компонентів, у середовищі яких повинен функціонувати протокол ARTCP. Модель являє собою набір компонентів реальні об'єкти, що імітують, у складі мережі й об'єкти протоколів ARTCP і CBR (Constant Bit Rate). Досліджувана мережа може мати топологічну схему досить великої складності, що будується з необхідного числа екземплярів наявних класів.

Основними способами що дозволяють досліджувати й верифікувати складні мережні протоколи є стенди для тестування (експериментальні мережі), програмні емулятори й системи автоматизованої верифікації. Верифікацію набору процедурних правил протоколу можна здійснювати за допомогою програмного перебору станів кінцевого автомата представляючого протокол [56,57,58], наприклад, в інтерпретаторі мови PROMELA, такому як SPIN [3]. Однак верифікація набору процедурних правил, і вивчення ефективності протоколу переслідують різні цілі й, відповідно, для них застосовуються різні

інструменти. Верифікація протоколу означає застосування до його набору процедурних правил формального методу, що дозволяє довести, що цей набір або моделююча його система кінцевих автоматів (з обміном даними) повна, не містить недосяжних станів, вільна від статичних і динамічних блокувань. Верифікація протоколу не ставить задачею навіть визначення кількісних характеристик його ефективності, з погляду верифікації важливо лише те, що прогресивний обмін даними взагалі відбувається. Внаслідок цього методи верифікації побудовані на повному або частковому аналізі доступних станів КА, що представляє протокол. Для систем з великим числом станів ($>10^5$) верифікація заснована на повному аналізі утруднена на практиці.

Моделювання протоколу не дає гарантії повного аналізу досяжних станів, але зате дозволяє досліджувати кількісні характеристики системи. Разом з тим моделювання дозволяє зробити статистично обґрунтований висновок про надійність протоколу, принаймні, щодо відсутності в ньому блокувань. Складність імітаційного моделювання в тому, що крім реалізації процедурних правил самого досліджуваного протоколу до складу моделі повинні входити всі його компоненти: середовище функціонування, словник і способи кодування повідомлень, модель сервісу протоколу. Однак при цьому моделювання вимагає все-таки менших витрат, ніж розгортання експериментальної мережі для дослідження властивостей протоколу. У цьому випадку наше завдання полягає не у верифікації протоколу ARTCP, а у визначенні чисельних значень його характеристик у різних умовах.

3.2 Розробка структурної схеми

У протоколі ARTCP повністю перероблені всі механізми керування потоком.

Механізм корекції помилок передачі в ARTCP не впливає на швидкість передачі. Від TCP збережені віконний механізм для керування завантаженням одержувача, алгоритми визначення RTT і установки таймера ТПП. Ознакою

втрати сегмента служить спрацьовування ТПП або прихід двох послідовних підтверджень одного сегмента. Алгоритм керування швидкістю містить у собі: функції диспетчеризації сегментів, виміру швидкості й адаптації швидкості (рисунок 3.5). Далі розглянемо ці функції докладно.

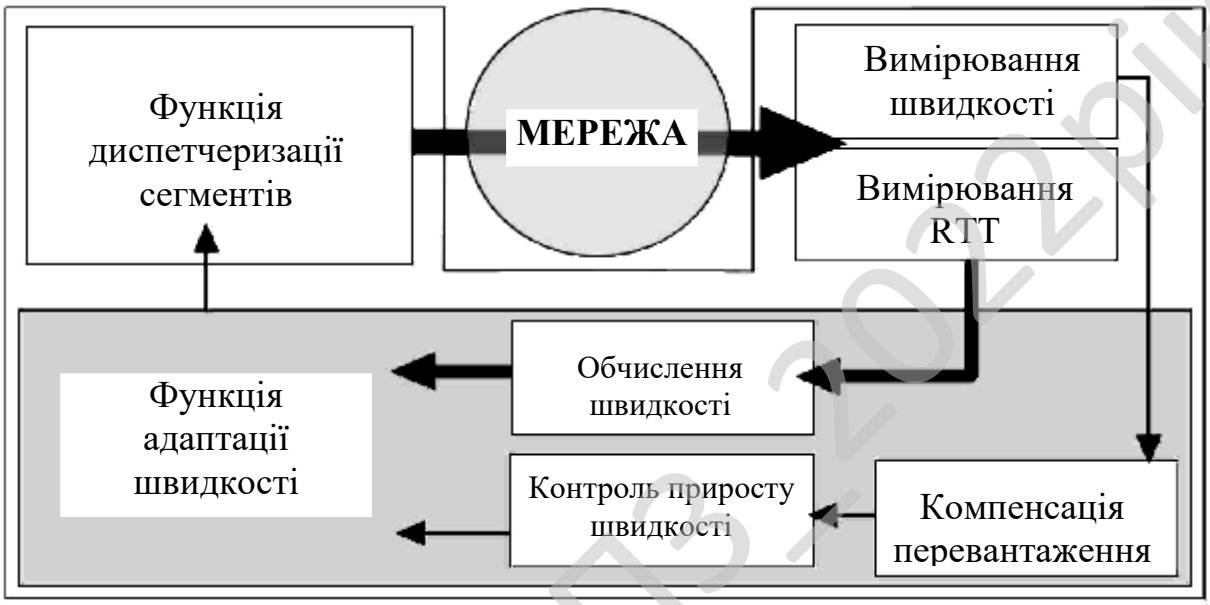

Рисунок 3.5 – Структурна схема сситеми

У задачі функції диспетчеризації сегментів входить відправлення сегментів у мережу зі строго заданою швидкістю, що виражається в значеннях міжсегментних часових інтервалів.

Функція виміру швидкості визначає шпаруватість потоку вступника до одержувача й інформує відправника про темпоральні параметри потоку, що прибуває. Крім того, відправник сам робить вимір часу RTT (у рамках стандартної функціональності протоколу TCP).

Значення шпаруватості потоку обмірюваної одержувачем і часу RTT обмірюваного відправником надходять на вхід функції адаптації, що визначає нове значення швидкості відправлення потоку відповідно до отриманого на вхід значеннями й своїм станом у цей момент часу.

ARTCP використовує дві ознаки початку перевантаження мережі, коли середня швидкість прибуття запитів рівняється із середньою швидкістю обслуговування й початком росту черги: початок росту RTT і стабілізацію $R_r(t)$ при збільшенні $R_s(t)$. Одержувач ARTCP у сегментах з підтвердженнями вказує значення швидкості прибуття потоку. Одержуючи підтвердження сегмента через час RTT після його відправлення, джерело ARTCP одержує інформацію про значення швидкості, з якого потік, що містить цей сегмент, прибув до одержувача й використовує $R_r(t)$ як оцінка $R_e(t)$ ПрЗд мережі.

3.3 Розробка функціональної схеми

Для реалізації протоколу необхідно створити імітаційну модель.

Для організації моделі мережі за найпростішою схемою (рисунок 3.6) потрібен набір з 14 об'єктів (рисунок 3.7)

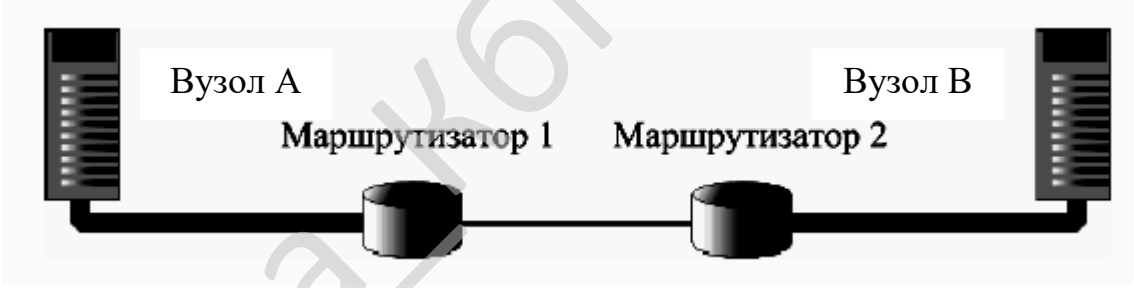

Рисунок 3.6 – Приклад найпростішої топологічної схеми, що складається із двох вузлів і двох маршрутизаторів

На рисунку 3.7 наведена функціональна схема розробленої системи, виходячи зі схеми зображеної на рисунку 3.6. З рисунка 3.7 бачимо, що для реалізації імітаційної моделі встановлюється два хоста та два маршрутизатора, через які йде трафік. Зафарбована область позначає топологічні елементи мережі (канали й маршрутизатори).

На функціональній схемі вказані усі елементи, які необхідні для передачі пакета по протоколу ARTCP у мережі. До них відносяться:

– об’єкти протоколу ARTCP та CBR;

– хости;

– симплексні канали передачі даних;

– інтерфейси по яким відбувається зєднання між маршрутизаторами;

– маршрутизатори;

– комутуюче поле у маршруторі (таблиця маршрутизації).

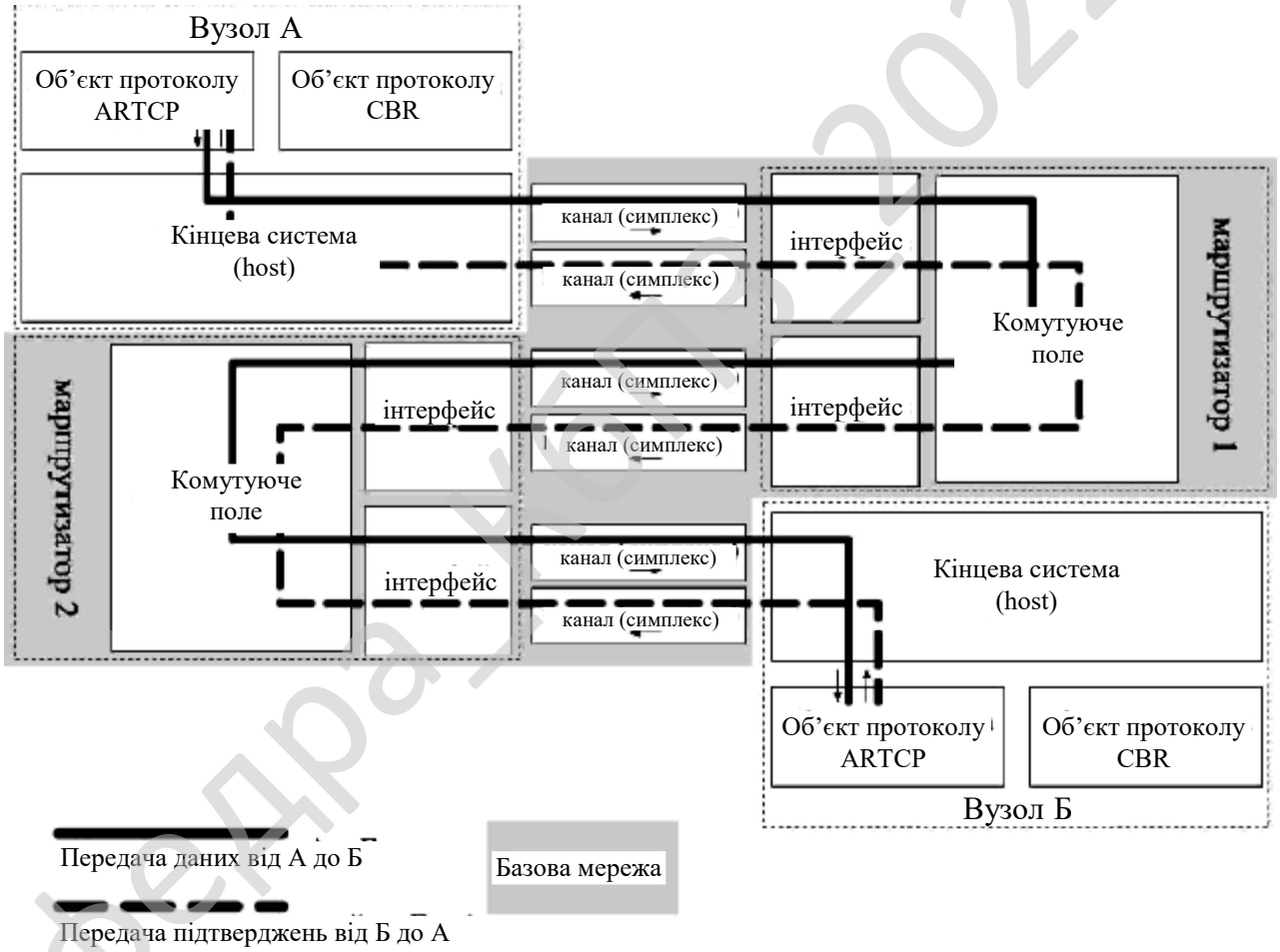

Рисунок 3.7 – Функціональна схема системи

3.5 Розробка діаграми процесів

На рисунку 3.8 зображена діаграма процесів, які відбуваються у системі. З цієї діаграми бачимо, що після початку роботи з програмою запускаються два основні процеси:

– з’єднання з мережею;

– роз’єднання з мережею.

У випадку коли відбувається з’єднання з мережею запускається процес введення параметрів протоколу адаптованого TCP. Для цього вводяться параметри адаптації.

Після цього запускається процес генерації пакетів протоколу. Якщо пакети згенеровані та вони передаються по мережі.

Цей процес дає змогу запустити наступні два процеси :

– процес адаптації швидкості;

– процес відображення стану з’єднання.

Для того, щоб визначити, наскільки розроблений протокол кращий ніж стандартний TCP запускаються наступні процеси, які дають чисельні характеристики виграшу:

– процес контролю приросту швидкості;

– процес визначеня пропускної здатності каналу;

– зміни режимів роботи розробленого адаптованого протоколу;

– процес обчисленя швидкості.

Процесс обчислення швидкості реалізується за рахунок отримання даних від наступних процесів:

– процес суто вимірювання швидкості;

– процес вимірювання затримок RTT.

Перелічени вище процеси та наявність зв’язків при їх взаємодії, дозволяє у повній мірі описати розроблену програму, яка реалізує та досліджує запропонований адаптований протокол.

У наступному розділі перейдемо до розгляду блок-схем реалізованого, у результаті виконання магістерської роботи, програмного продукту.

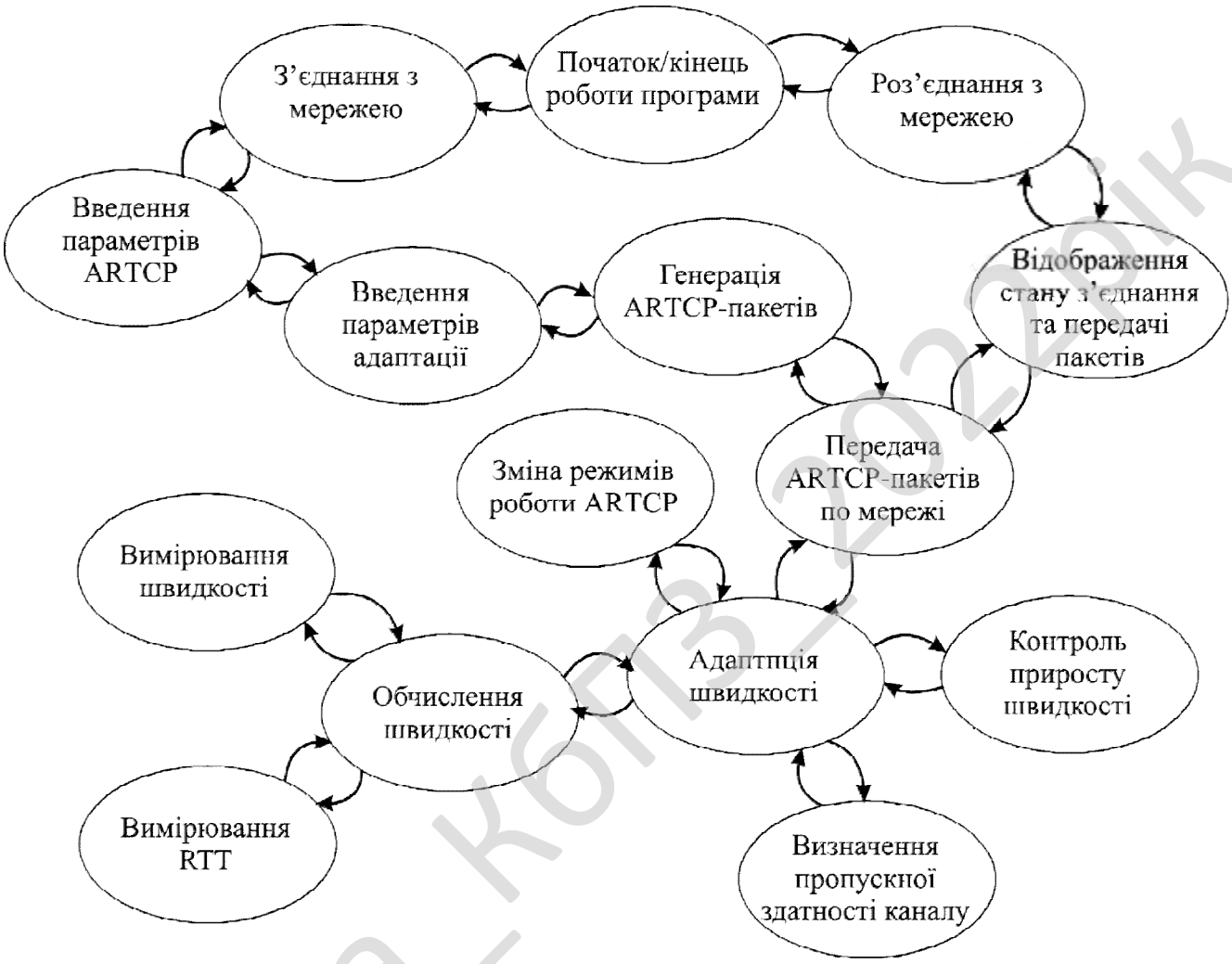

Рисунок 3.8 – Діаграма процесів системи

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

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображена блок-схема роботи основної програми. Робота програми починається з виведення основного вікна програми.

За виведенням основного вікна програми слідує з'єднання з мережею.

Після з'єднання з мережею виводяться параметри розробленого адаптивного транспортного протоколу.

На основі вивчення виведених параметрів, для поліпшення роботи по передачі трафіка по мережі, вводяться параметри адаптації.

Після цього користувач вводить дані, які потрібно передати по мережі.

Тескт розбивається на ARTCP-пакети, які генеруються та відправляються.

При цьому відбувається вимірювання та обчислення швидкості й адаптація швидкості передачі, згідно завантаженості мережі.

Паралельно для користувача виводиться стан мережного з’єднання та стан передачі пакетів.

Після закінчення передачі даних програма або завершає свою працю, або чекає наступних даних, які потрібно передати.

На рисунку 4.2 зображена блок-схема роботи підпрограми з'єднання з сервером та передачі даних по алгоритму ARTCP. Перш ніж перейти до розгляду цієї підпрограми дамо визначення використовуємим у ній позначеням та процесам.

Установка з'єднання

Процес початку сеансу ARTCP називається «потрійним рукостисканням». Клієнт, що має намір установити з'єднання, посилає серверу сегмент із номером послідовності й прапором SYN.

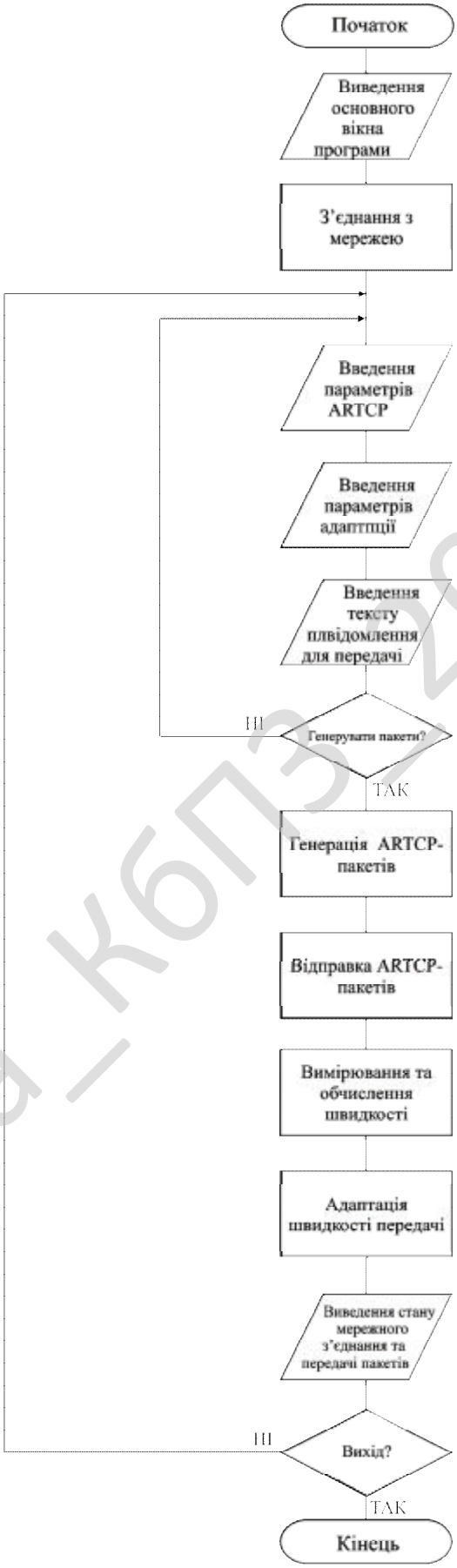

Рисунок 4.1 – Блок-схема роботи основної програми

Сервер одержує сегмент, запам'ятовує номер послідовності й намагається створити сокет (буфери й керуючі структури пам'яті) для обслуговування нового клієнта. У випадку успіху сервер посилає клієнтові сегмент із номером послідовності й прапорами SYN і ACK, і переходить у стан SYN-RECEIVED. У випадку невдачі сервер посилає клієнтові сегмент із прапором RST.

Якщо клієнт одержує сегмент із прапором SYN, то він запам'ятовує номер послідовності й посилає сегмент із прапором ACK, якщо він одночасно одержує й прапор ACK (що звичайно й відбувається), те він переходить у стан ESTABLISHED. Якщо клієнт одержує сегмент із прапором RST, то він припиняє спроби з'єднатися.

Якщо клієнт не одержує відповіді протягом 10 секунд, то він повторює процес з'єднання заново.

Якщо сервер у стані SYN-RECEIVED одержує сегмент із прапором ACK, то він переходить у стан ESTABLISHED. У противному випадку після таймаута він закриває сокет і переходить у стан CLOSED.

Процес називається «потрійним рукостисканням», оскільки можливо процес установлення з'єднання з використанням 4 сегментів (SYN убік сервера, ACK убік клієнта, SYN убік клієнта, ACK убік сервера), але для економії часу використовується 3 сегменти.

Передача даних

При обміні даними приймач використовує номер послідовності, що втримується в одержуваних сегментах, для відновлення їхнього вихідного порядку. Приймач повідомляє передавальну сторону про номер послідовності, до якої він успішно одержав дані, включаючи його в поле «номер підтвердження». Всі одержувані дані, що відносяться до проміжку підтверджених послідовностей, ігноруються. Якщо отриманий сегмент містить номер послідовності більший, ніж очікуваний, то дані із сегмента буферізується, але номер підтвердженої послідовності не змінюється. Якщо в наслідку буде прийнятий сегмент, що

відноситься до очікуваного номера послідовності, то порядок даних буде автоматично відновлений виходячи з номерів послідовностей у сегментах.

Для того, щоб передавальна сторона не відправляла дані інтенсивніше, ніж їх може обробити приймач, ARTCP містить засобу керування потоком. Для цього використовується поле «вікно». У сегментах, що направляються від приймача передавальній стороні в поле «вікно» вказується поточний розмір прийомного буфера. Передавальна сторона зберігає розмір вікна й відправляє даних не більш, ніж указав приймач. Якщо приймач указав нульовий розмір вікна, то передача даних у напрямку цього вузла не відбувається, доти поки приймач не повідомить про більший розмір вікна.

У деяких випадках передавальний додаток може явно зажадати проштовхнути дані до деякої послідовності приймаючому додатку, не буферізуя їх. Для цього використовується прапор PSH. Якщо в отриманому сегменті виявляється прапор PSH, то реалізація ARTCP віддає всі буферізовані на сучасний момент дані приймаючому додатку. «Проштовхування» використовується, наприклад, в інтерактивних додатках. У мережних терміналах нема рації очікувати уведення користувача після того, як він закінчив набирати команду. Тому останній сегмент, що містить команду, зобов'язаний містити прапор PSH, щоб додаток на приймаючій стороні змогло почати її виконання.

Завершення з'єднання

Завершення з'єднання можна розглянути в три етапи:

1. Посилка серверу від клієнта прапорів FIN і ACK на завершення з'єднання.

2. Сервер посилає клієнтові прапори відповіді ACK , FIN, що з'єднання закрите.

3. Після одержання цих прапорів клієнт закриває з'єднання й на підтвердження відправляє серверу ACK , що з'єднання закрито.

Номер послідовності

Номер послідовності виконує два завдання:

Якщо встановлено прапор SYN, то це початкове значення номера послідовності й перший байт даних – це номер послідовності плюс 1.

У противному випадку, якщо SYN не встановлений, перший байт даних – номер послідовності

Оскільки ARTCP-потік у загальному випадку може бути більш довгим, чим число різних станів цього поля, те всі операції з номером послідовності повинні виконуватися по модулі 2^32. Це накладається практичне обмеження на використання ARTCP. Якщо швидкість передачі комунікаційної системи така, щоб протягом MSL (максимального часу життя сегмента) відбулося переповнення номера послідовності, то в мережі може з'явитися два сегменти з однаковим номером, що відносяться до різних частин потоку, і приймач одержить некоректні дані .

Прапори (керуючі біти)

Це поле містить 6 бітових прапорів:

– URG – Поле "Покажчик важливості" задіяне (Urgent pointer field is significant)

– ACK – Поле "Номер підтвердження" задіяне (Acknowledgement field is significant)

– PSH – (Push function) інструктує одержувача проштовхнути дані, що нагромадилися в прийомному буфері, у додаток користувача

– RST – Обірвати з'єднання, скинути буфер (очищення буфера) (Reset the connection)

– SYN – Синхронізація номерів послідовності (Synchronize sequence numbers)

– FIN (final, біт) – прапор, будучи встановлений, указує на завершення з'єднання (FIN bit used for connection termination).

Тепер розглянувши у загальному вигляді процеси та визначення, які нам потрібні для поняття роботи алгоритму ARTCP, перейдемо до опису блок схеми підпрограми з'єднання з сервером та передачі даних по алгоритму ARTCP.

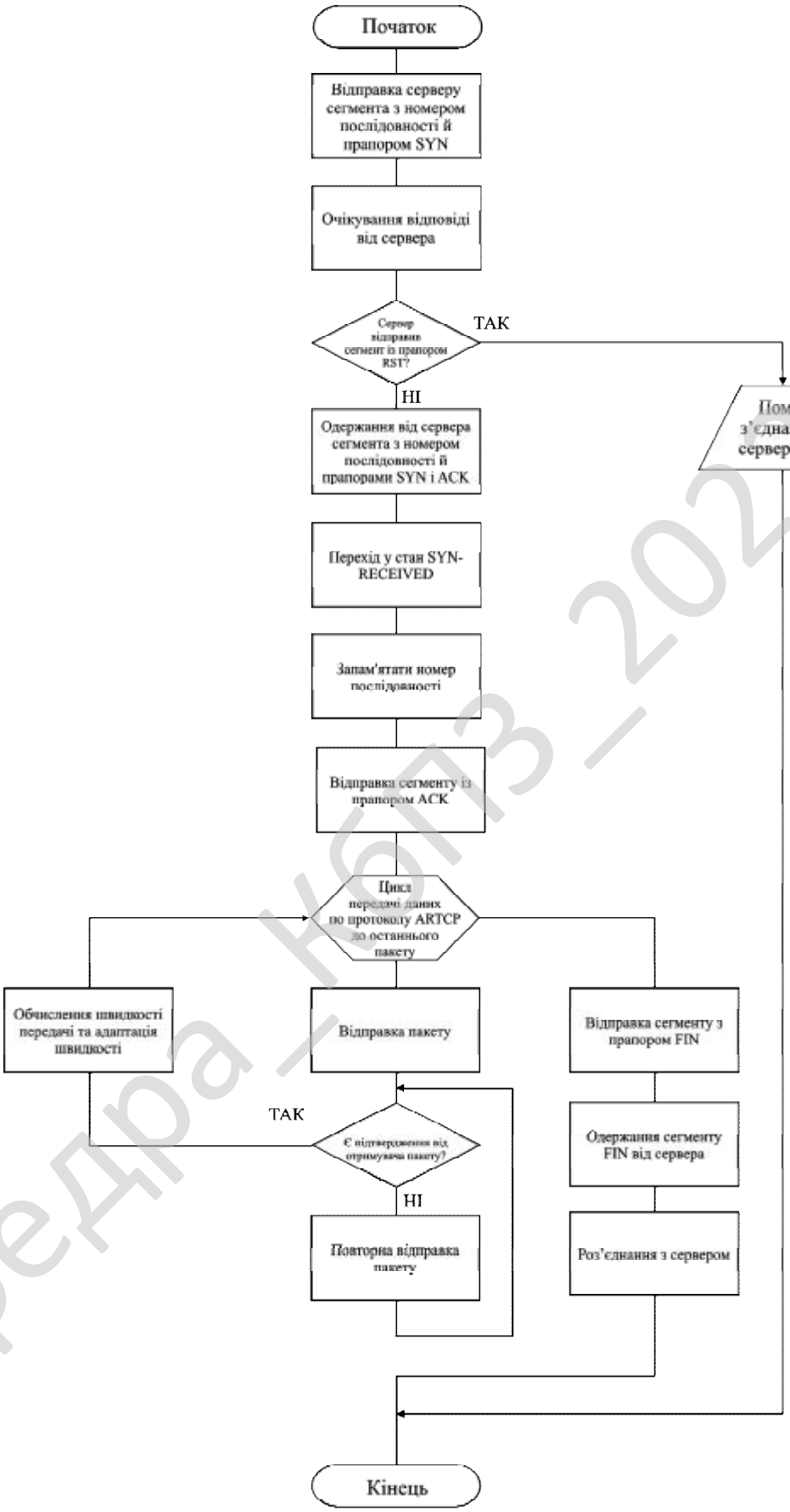

Рисунок 4.2 – Блок-схема роботи підпрограми з'єднання з сервером та передачі даних по алгоритму ARTCP

Підпрограма починає працювати з відправки серверу (або хосту, який приймає інформацію) сегмента з номером послідовності й прапором SYN.

Після цього відбувається очікування відповіді від сервера.

Перевіряється який семент відправив сервер. Якщо він відправив сегмент з прапором RST, то відбулася помилка з’єднання з сервером.

У іншому випадку одержуємо від сервера сегмент з номером послідовності та прапорами SYN та ASK.

Відбувається перехід у стан SYN-RECEIVED.

Після цього відбувається запам’ятовування номеру послідовності та відправка сегменту з прапором АСК.

За цим у циклі починається передача даних по розробленому адаптованому транспортному протоколу.

Цикл працює до моменту підтвердження від отримувача пакету. При цьому паралельнообчислюється швидкість передачі та відбувається адаптація швидкості згідно завантаженості мережі.

Коли усі пакети передано. То відбувається відправка сегменту з прапором FIN.

Коли одержується сегмент FIN від сервера, то відбуається роз’єднання з сервером.

На цьому завершається робота підпрограми з'єднання з сервером та передачі даних по алгоритму ARTCP.

На рисунку 4.3 наведена блок-схема підпрограми адаптації швидкості. Ця підпрограма є основною у алгоритмі адаптації.

Підпрограма працює наступним чином.

Відбувається робота у режимі прискореного старту (SS).

За цим слідує визначення пропускної здатності та RTT.

Якщо верхня меж пропускної здатності не досягнта то відбувається перехід на початок підпрограми до моменту, поки не досягнеться максимальна швидкодія.

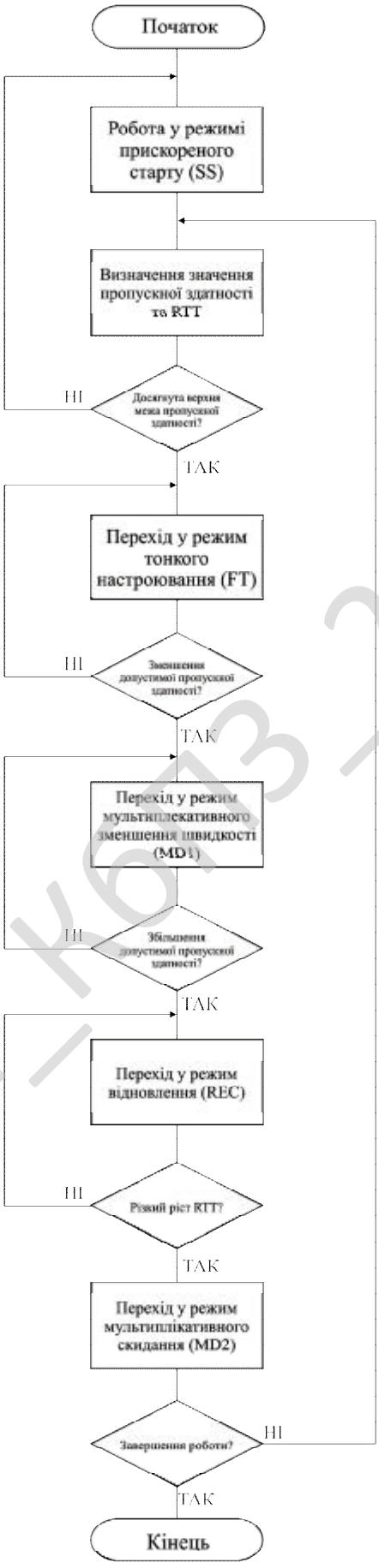

Рисунок 4.3 – Блок-схема роботи підпрограми адаптації швидкості

Якщо ж максимальна швидкість досягнута, то відбувається перехід у режим тонкого налаштування (FF).

Якщо зменшення пропускної здатності не відбулося, то переходимо у режим мультипликативного зменшення швидкості (MD1).

Якщо відбулося збільшення пропускної здатності то переходимо у режим відновлення (REC).

Якщо відбувся різкий ріст RTT, то переходимоу режим мультиплікативного скидання (MD2).

Після цього підпрограма або щавершує роботу, або переходить на початок підпрограми в залежності від потреби.

Далі розглянемо формат повідомлення, використовуваний у моделі, ієрархію й реалізацію її основних класів.

Формат повідомлення

Нові сегменти створюються усередині об'єктів протоколів ARTCP або CBR. Всі об'єкти топології передають один одному покажчик на область пам'яті, що містить відповідний сегмент, замість копіювання реальних даних. Крім того, розміри заголовка сегмента й поля корисного навантаження задаються в самій структурі даних, і всі об'єкти визначають розмір сегмента, інтерпретуючи значення цих полів. Оскільки посилання на об'єкт може одночасно втримуватися у всіх об'єктах, де реалізована буферізація: ARTCP, link, interface, то щоб уникнути конфліктів лічильник посилань зберігає кількість об'єктів, у буферах, в яких перебуває посилання на даний сегмент. Звільнення області пам'яті, що містить сегмент, дозволено тільки у випадку обнуління лічильника. За винятком полів "TI" і "PS" і службових полів link, id, всі інші поля сегмента відповідають полям стандартного TCP заголовка.

Поле заголовка ARTCP у моделі не містить перевірочної суми, ушкодження сегментів у транзиті можуть моделюватися шляхом реалізації випадкового відкидання сегментів на мережних вузлах або в кінцевій системі з

імовірністю, що відповідає ймовірності бітових помилок середовища передачі, яку необхідно змоделювати.

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

Інтерфейс кожного із цих класів визначається трьома найважливішими компонентами:

– прийом сегмента ;

– запит сервісу;

– обробка переривання.

Ці методи є перевизначенням віртуальних функцій базового класу. Крім того, кожний похідний клас розширений додатковими специфічними для нього методами. За допомогою цих елементів інтерфейсу моделюється передача даних у системі (рисунок 3.7). Кожний зі спадкоємців класу element посилається на елемент, з яким він зв'язаний топологічно, по покажчику на базовий клас.

Екземпляр кожного з наведених класів використовує виклик топологічно прив'язаного до нього об'єкта для передачі йому сегмента. Залежно від наявності ресурсів метод викликуваного об'єкта може прийняти або не прийняти сегмент на обслуговування. Метод кожного екземпляра викликається з основної програми для обробки черги й ухвалення рішення про обслуговування чергового сегмента.

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

Клас link: аспекти реалізації

Клас, що моделює канал, одержує значення пропускної здатності й затримки передачі при ініціалізації. Структура даних класу реалізується однозв'язною динамічною чергою, у яку містяться сегменти, прийняті на обслуговування. Після початку обслуговування сегмента канал блокує наступні запити протягом часу $t=s/R$, де s – розмір сегмента, а R – швидкість каналу, тобто часу прийому сегмента розміру s у канал. У черзі каналу сегмент затримується

протягом установленого часу затримки передачі. Після закінчення цього часу, об'єкт каналу викликає метод наступного об'єкта в топологічній схемі мережі. Якщо наступний об'єкт відмовляє в обслуговуванні сегмента, то цей сегмент відкидається. Для моделювання дуплексного зв'язку застосовуються 2 екземпляри каналу. Параметри каналів можуть бути різними для моделювання асиметричних систем.

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

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

Клас router: аспекти реалізації

Структура класу router є складною. У його склад входять кілька екземплярів класу interface. При ініціалізації класу router йому передаються два параметри: кількість інтерфейсів і об'єм буферного простору (максимальна довжина черги), однаковий для всіх інтерфейсів. Об'єкт маршрутизатора створює задана кількість екземплярів класу interface. Покажчики на кожний інтерфейс розташовуються в спеціальному масиві, проіндексованому один по одному створення інтерфейсів. Після цього на конкретний інтерфейс можна посилатися як router.interface(X), де X – номер потрібного інтерфейсу. Кожний інтерфейс при ініціалізації одержує покажчик на його об'єкт, що створив, класу router, по якому він може посилатися на методи класу router для передачі сегментів у матрицю комутації (позначена в схемах як "поле комутації").

При прийманні сегмента інтерфейс негайно передає його об'єкту маршрутизатора. Блокування при цьому відбутися не може, тобто моделюється

не комутаційна матриця, що блокує. Об'єкт маршрутизатора вносить запис, що складається із двох полів: {номер інтерфейсу, адреса відправника} у таблицю маршрутизації. Після цього зіставлення адреси призначення сегмента із другим полем таблиці маршрутизації дає номер вихідного інтерфейсу для даного сегмента. При відсутності збігу із записами таблиці маршрутизації, сегмент передається для розсилання всім інтерфейсам маршрутизатора, крім того, з якого він був отриманий. Отриманий від комутуючої матриці, сегмент міститься у вихідну чергу інтерфейсу, якщо вільний простір у ній $Q_{max} - Q \geq s$, у противному випадку сегмент відкидається. Черга моделюється списком подвійної зв'язності, вишиковується динамічно в пам'яті [63]. Черга обслуговується зі швидкістю каналу, до якого підключений інтерфейс.

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

- середня швидкість;
- лічильники пропущених/відкинутих сегментів;
- таблиця маршрутизації;
- середня довжина черги.

Таким чином, об'єкт маршрутизатора моделює сучасний міжмережний пристрій з не комутаційною матрицею, що блокує, і буферизацією на виході [67]. Можливо також розширення класу маршрутизатора алгоритмами RED і WFQ або CBQ [65,66].

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

Клас host: аспекти реалізації

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

При відправленні сегмента метод екземпляра класу host викликається одним із протоколів. У випадку прийому сегмента на обслуговування об'єктом каналу передачі, позитивне підтвердження вертається об'єкту протоколу, що викликав метод, і покажчик на область пам'яті, що зберігає сегмент передається об'єкту канального рівня. У випадку відсутності можливості передати сегмент на канальному рівні, метод об'єкта host повертає негативне підтвердження. При ініціалізації екземпляра класу host він одержує мережну адресу. Окремо встановлюється мережна адреса призначення.

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

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

Об'єкт протоколу CBR

Протокол постійної швидкості передачі прямо відповідає реальному режиму передачі даних з постійною бітовою швидкістю, наприклад CBR в ATM [68] або (Circuit emulation services) CES [69]в IP мережах. Коректна взаємодія з таким режимом передачі даних є актуальною задачею будь-якого адаптивного транспортного протоколу в сучасних мережах з інтеграцією сервісів [30]. Саме тому протокол CBR включений у модель середовища виконання ARTCP. Протокол CBR генерує сегменти й відправляє їх у мережу із передвстановленою швидкістю R_{CBR} , тобто часові проміжки між моментами початку послідовних

трансляцій становлять $\tau_{CBR} = s / R_{CBR}$. Підтвердження й, отже, ретрансляція сегментів і контроль швидкості не реалізується протоколом CBR.

Об'єкт протоколу ARTCP

Модель ARTCP реалізує:

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

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

– швидку ретрансляцію сегментів і стандартну ретрансляцію по спрацьовуванню ТПП.

Основні призначення методів класу ARTCP такі:

– застосовуються для початкової синхронізації з'єднання;

– обчислюють значення області компенсації;

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

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

Клас Queue

Даний клас реалізує схему стандартного керування потоком по методу ковзного вікна. Клас містить динамічний список подвійної зв’язності, у який записуються покажчики на сегменти, поставлені в чергу разом з екземпляром класу таймера, що реалізує ТПП. Методи класу Queue дозволяють здійснювати вставку сегмента із сортуванням, сканування списку, знаходження чергового сегмента готового до передачі. Ретрансляція реалізована за допомогою зміни черговості готових до відправлення сегментів. Взаємодія черг із об'єктом протоколу й схема обробки сегментів наведені на рисунку 4.4.

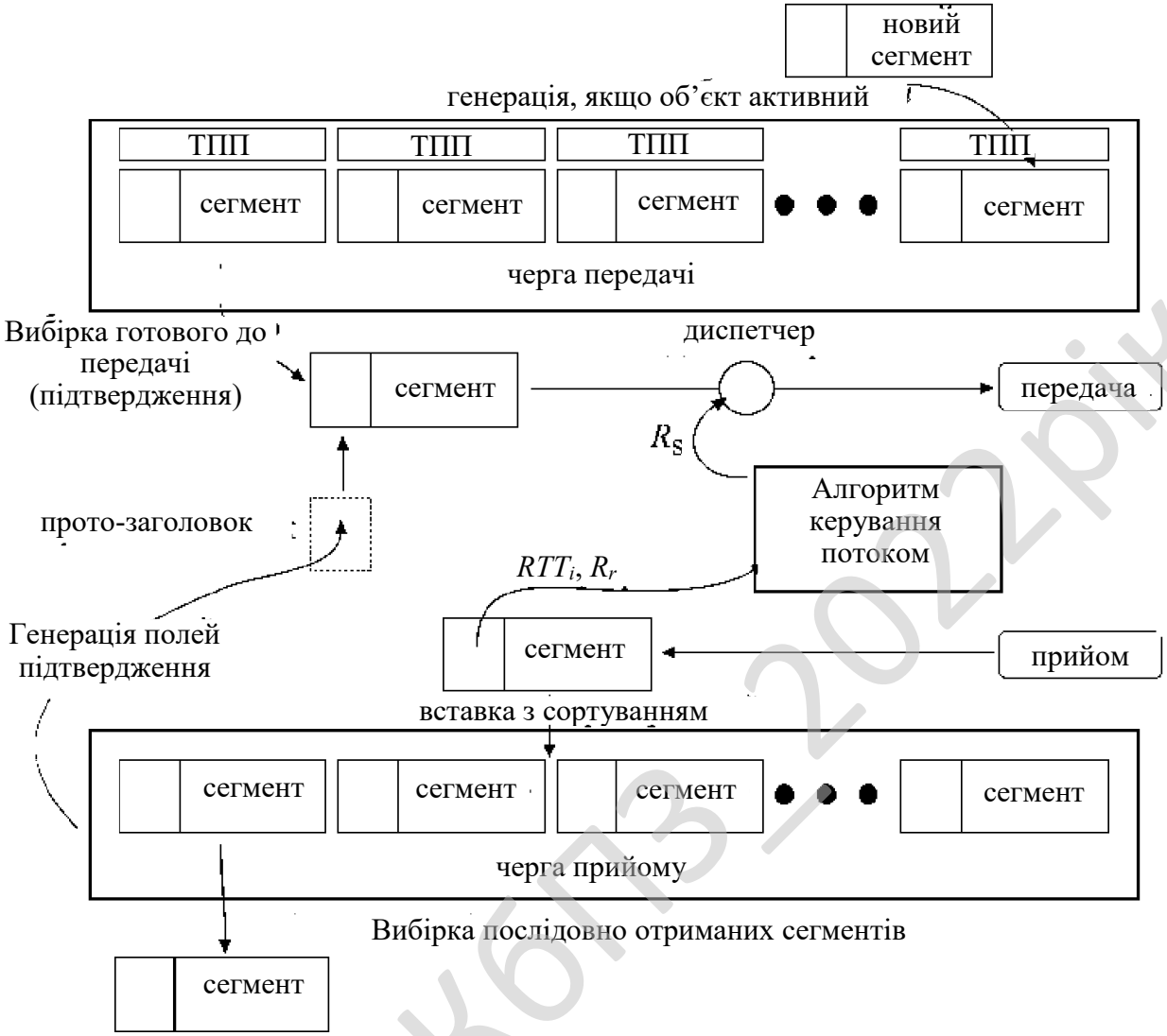

Рисунок 4.4 – Концептуальна схема взаємодії черг прийому й передачі з алгоритмом керування потоком в об'єкті протоколу ARTCP

Прямі лінії вказують напрямок потоків даних, криві – керуючої інформації.

Обробка прийому

Сегмент, прийнятий вузлом, передається об'єкту ARTCP. Метод приймає сегмент і обробляє його як підтвердження, якщо сегмент містить поле підтвердження. Параметри ARTCP обчислюються, якщо встановлено поле "TI" сегмента. Якщо номер підтверджуваного сегмента утворить безперервну послідовність із порядковими номерами сегментів, що перебувають у черзі

передачі, то вся послідовність, крім підтверджуваного сегмента віддаляється із черги. Наприклад, сегменти в черзі передачі: {*k*, *k*+1, *k*+2, *k*+3, *k*+4, ...} і вузол одержує підтвердження {k+3}, із черги віддаляються сегменти *k*, *k*+1, *k*+2.

Якщо сегмент містить дані, то він міститься в прийомну чергу, після чого прийомна черга сканується, з її убираються (передаються користувачеві) сегменти, отримані в безперервній послідовності порядкових номерів. Потім генерується підтвердження наступного очікуваного одержувачем сегмента, у поля якого заносяться наступні значення: ack – кумулятивне підтвердження, ack-trig – номер останнього отриманого сегмента, advwnd – вільний простір у черзі прийому. Наприклад, у черзі прийому перебувають сегменти {*i*+1, *i*+2, *i*+3}, після одержання *i-го* сегмента і його запису в чергу, з її віддаляються сегменти i, i+1, *i*+2, *i*+3 і значення кумулятивного підтвердження стає рівним *i*+4. Прототип контрольного сегмента з підтвердженням ставиться на чергу до передачі.

Обробка переривання

Метод активного протоколу викликається з методу утримуючого його екземпляра класу вузла. Даний метод обновляє значення внутрішнього лічильника часу екземпляра об'єкта протоколу, обчислює поточну швидкість передачі потоку в стані **SS**; у стані **REC**; у стані **FT**.

Обчисливши значення міжсегментного інтервалу, метод класу протоколу ARTCP визначає, чи пройшов цей час із моменту відправлення попереднього сегмента. Якщо так, то запитується екземпляр черги передачі на предмет наявності в ній готового до відправлення сегмента. Сегмент для відправлення береться із черги або створюється новий (у випадку відсутності готового сегмента в черзі), після чого в поля ack, acktrig, advwnd заносяться значення із прототипу контрольного сегмента, й сегмент передається об'єкту вузла для передачі. Якщо передача успішна, то вузол, викликаючи метод, повідомляє про це об'єкт протоколу, що відзначає відповідний сегмент як відправлений і запускає ТПП для нього.

Прискорена ретрансляція

Підтримка прискореної ретрансляції сегментів також включена в ПМ. Для цього об'єкт протоколу ARTCP відслідковує надходження дублюючих підтверджень. Якщо послідовно приходять два підтвердження того самого і-го сегмента, то ARTCP здійснює ретрансляцію даного сегмента поза залежністю від значення його ТПП. Після здійснення ретрансляції алгоритм переходить у стан, у якому прискорена ретрансляція не дозволена й залишається в цьому стані поки не прийде хоча б одне підтвердження наступні: (i+1)-го сегмента.

Початкова синхронізація

Початкове значення RTT необхідно для обчислення початкової (мінімальної) швидкості роботи з'єднання, рівної s байт за час RTT. Для здійснення цього виміру при відкритті нового з'єднання об'єкт протоколу ARTCP реалізує обмін сегментом спеціального типу із протилежною стороною. При активації об'єкт ARTCP попадає в "не синхронізований" стан. Сегмент, позначений як SYN (по наявності відповідного прапора), відправляється від ініціюючої сторони обміну до одержувача. Сегмент SYN несе порядковий номер, що не впливає на порядкові номери при обміні даними. Одержувач сегмента SYN реагує відправленням сегмента із установленими прапорами SYN, ACK, де поле ack несе значення порядкового номера SYN сегмента. Одержання SYN, ACK сегмента дає можливість відправникові виміряти час RTT і переводить з'єднання в стан "синхронізації", після чого починається обмін даними. Наявність ідентифікуючого SYN сегмент порядкового номера дає можливість розрізнити їхні можливі копії й правильно виміряти час RTT. Синхронізація з'єднання може бути ініційована одночасно обома сторонами. Для протоколу TCP, наприклад початкова синхронізація має місце при відкритті з'єднання й має на меті встановити ідентичні початкові значення змінних у контрольних блоках обох сторін з'єднання.

Головний цикл

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

Дуплексний режим

Протокол ARTCP, як і TCP здатний здійснювати симетричний обмін інформацією з одного з'єднання. У такому режимі контрольна інформація одержувача транслюється не окремими сегментами, а записується у відповідні поля сегментів з даними, що випливають у протилежному напрямку. Програмна модель передбачає наявність буфера під прототип заголовка контрольного сегмента. Цей заголовок, що містить номер підтвердження, розмір вікна й іншу керуючу інформацію, формується після одержання чергового сегмента з даними. Однак замість негайного відправлення порожнього сегмента з підтвердженням, створеним із прототипу, модель перевіряє наявність сегмента з даними чекаючого передачі і якщо сегмент такий сегмент є, то підтвердження передається разом з ним. Інакше із прототипу контрольного сегмента створюється окремий сегмент не несучий даних, а лише передавальне підтвердження.

Трасування моделі

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

Формат цього запису такий:

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

psk ack trig adv wnd link seq/syn ack id

де:

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

– time – час у секундах;

– elem – ідентифікатор об'єкта створившего дію;

– src – адреса відправника сегмента;

– dst – адреса одержувача сегмента;

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

– size – розмір сегмента в байтах;

– seq – порядковий номер;

– ack – номер кумулятивного підтвердження;

– psn – поле "PS";

– psk – поле "TI";

– ack_trig – номер сегмента викликавшого відправлення підтвердження;

– adv_wnd – розмір вікна одержувача в байтах;

– link – лічильник посилань на область пам'яті;

– synack – номер підтверджуваного SYN сегмента;

– id – унікальний ідентифікатор кожного сегмента.

Наприклад, наступний фрагмент звіту представляє передачу одного сегмента з даними і його підтвердження:

```
s 0 6251  0   0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 2 0/-1  2
+ 0 6357  6   0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 2 0/-1  2
+ 0 6358  22  0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 3 0/-1  2
+ 0 8608  11  0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 2 0/-1  2
+ 0 8609  24  0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 3 0/-1  2
r 0 8717  1   0- >1 D--  1000  0/-1 . | . 1 -   1  -1 65536 1 0/-1  2
+ 0 8717  25  1- >0 -A-  40    0/1  . | . -1  -1  0 65536 1 0/1   4
s 0 8718  1   1- >0 -A-  40    0/1  . | . -1  -1  0 65536 1 0/1   4
+ 0 8817  11  1- >0 -A-  40    0/1  . | . -1  -1  0 65536 1 0/1   4
+ 0 8818  23  1- >0 -A-  40    0/1  . | . -1  -1  0 65536 2 0/1   4
+ 0 9868  6   1- >0 -A-  40    0/1  . | . -1  -1  0 65536 1 0/1   4
+ 0 9869  17  1- >0 -A-  40    0/1  . | . -1  -1  0 65536 2 0/1   4
r 0 9970  0   1- >0 -A-  40    0/1  . | . -1  -1  0 65536 0 0/1   4
```

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

ретрансльований. Дослідження моделюємої системи виробляється за допомогою візуалізації інформації звіту.

Крім подій, що відбуваються із сегментом, у звіт записуються також значення змінних протоколу ARTCP і характеристики, що знімаються з мережних пристроїв (з певних інтерфейсів). На відміну від подій із сегментами, змінні ARTCP і дані з інтерфейсів маршрутизаторів знімаються із заданою періодичністю. По кожному з подій звіт може виводитися як у короткій формі (наведеній вище для операцій із сегментами) так і в розгорнутому виді з метою налагодження.

Візуалізація даних

Для візуалізації отриманих при моделюванні даних застосовувалася програма gnuplot версії 3.7 для Os UNIX. Як правило, візуалізація роботи протоколу TCP виробляється за допомогою графіків залежності розміру вікна CWND від часу, залежності послідовності передачі сегментів від часу, що дозволяє візуально визначити різні фази роботи протоколу. У переважній більшості робіт в області транспортних протоколів застосовується саме така схема [87].

У деяких роботах приводяться більш складні системи візуалізації, наприклад в [14,15,19]. У цій роботі результати моделювання найбільше наочно представляються такими способами:

– залежність швидкості потоку;
– порядкового номера переданого сегмента;
– часу RTT, середньої довжини черги від часу.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використати фіналіста конкурсу AES – шифр Rijndael. Він є нетрадиційним блоковим шифром, оскільки не використовує мережу Фейштеля для

криптоперетворень. Алгоритм представляє кожний блок кодуємих даних у вигляді двовимірного масиву байт розміром 4x4, 4x6 або 4x8 залежно від установленої довжини блоку. Далі на відповідних етапах перетворення відбуваються або над незалежними стовпцями, або над незалежними рядками, або взагалі над окремими байтами в таблиці.

Всі перетворення в шифрі мають строге математичне обґрунтування. Сама структура й послідовність операцій дозволяють виконувати даний алгоритм ефективно як на 16-бітних так і на 64-бітних процесорах. У структурі алгоритму закладена можливість паралельного виконання деяких операцій, що на багатопроцесорних робочих станціях може ще підняти швидкість шифрування в 4 рази.

Алгоритм складається з деякої кількості раундів (від 10 до 14 – це залежить від розміру блоку й довжини ключа), у яких послідовно виконуються наступні операції :

ByteSub – Таблична підстановка 8x8 біт (рисунок 4.4).

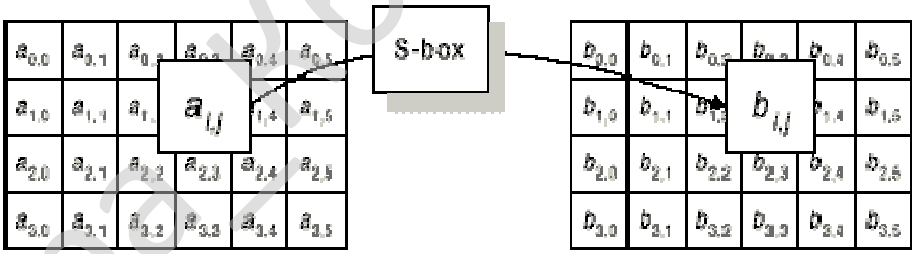

Рисунок 4.4 – Таблична підстановка 8x8 біт

ShiftRow – зрушення рядків у двовимірному масиві на різні зсуви (рисунок 4.5).

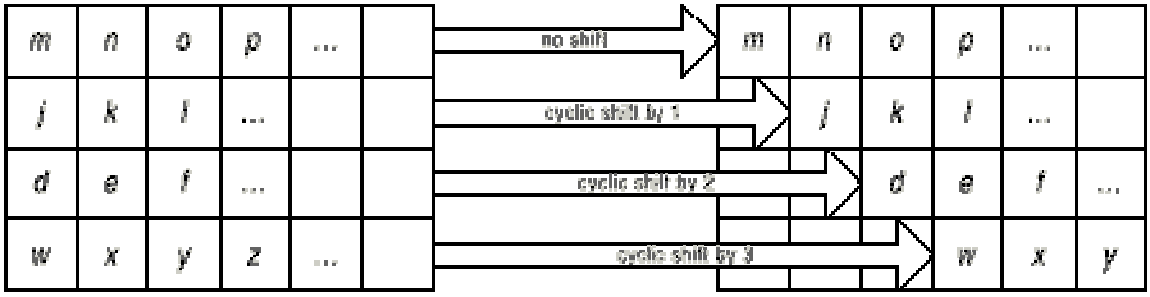

Рисунок 4.5 – Зрушення рядків у двовимірному масиві на різні зсуви

MixColumn – математичне перетворення, що перемішує дані усередині стовпця (рисунок 4.6).

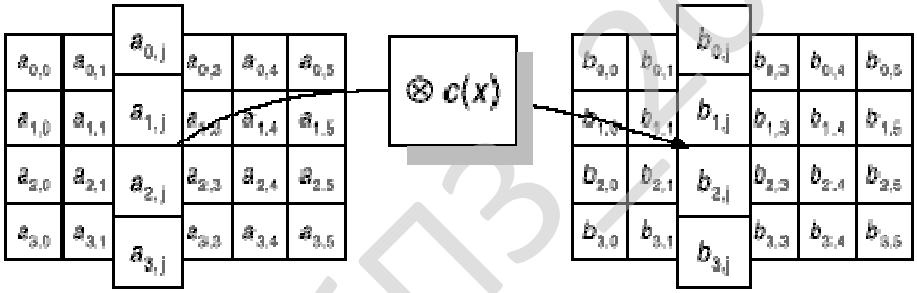

Рисунок 4.6 – Математичне перетворення, що перемішує дані усередині стовпця

AddRoundKey – додавання матеріалу ключа операцією XOR (рисунок 4.7).

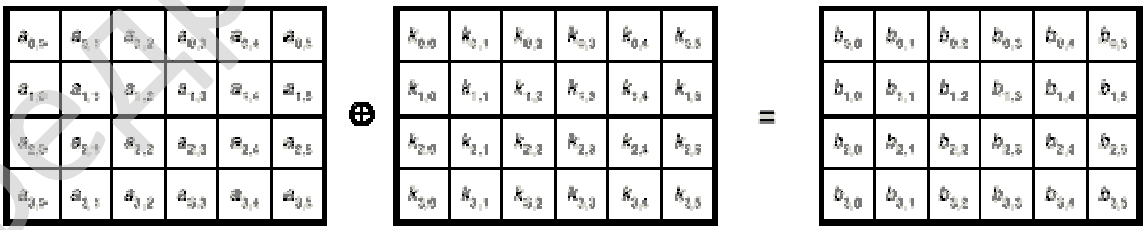

Рисунок 4.7 – Додавання матеріалу ключа операцією XOR

В останньому раунді операція перемішування стовпців відсутня, що робить всю послідовність операцій симетричною.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує адаптивну схеми управління потоком для TCP протоколу.

Програмно-апаратні вимоги:

– Загальний обсяг ОЗП: 256 Мбайт.
– Вільний простір на жорсткому диску: 5 Мбайт.
– Операційна система Microsoft Windows 10/11.
– Мережна бібліотека Winsock.

Головне вікно програми зображене на рисунку 5.1.

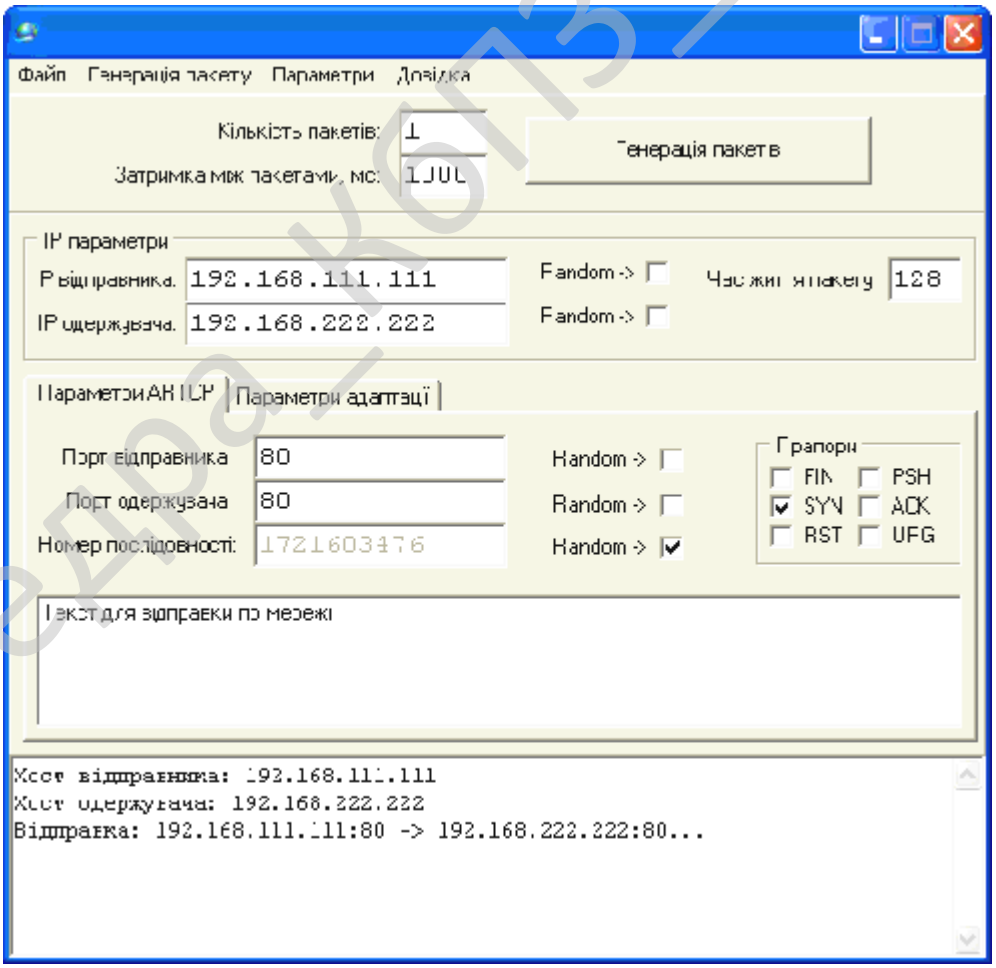

Рисунок 5.1 – Головне вікно програми

Для того, щоб згенерувати ARTCP-пакети, необхідно вказати наступні параметри:

– Кількість пакетів.

– Час затримки між пакетами у мілісекундах.

– IP відправника.

– IP одержувача.

– Час життя пакету.

– Порт відправника.

– Порт одержувача.

– Номер послідовності.

Далі треба встановити необхідні пратори з наступних:

– FIN – прапор, будучи встановлений, указує на завершення з'єднання.

– SYN – синхронізація номерів послідовності.

– RST – обірвати з'єднання, скинути буфер (очищення буфера).

– PSH – інструктує одержувача проштовхнути дані, що нагромадилися в прийомному буфері, у додаток користувача.

– ACK – *"Номер підтвердження"* задіяно.

– URG – Поле *"Покажчик важливості"* задіяно.

Потім слід вказати параметри адаптації:

– Значення пропускної здатності каналу зв'язку.

– Площа області компенсації.

– Параметр експонентного росту R_S у режимі SS.

– Часовий проміжок між моментом відправлення сегмента й моментом приходу його підтвердження.

– Зважене ковзне середнє RTT.

– Зважене ковзне середнє R_e.

– Максимальне значення згладженого RTT.

– Мінімальне значення згладженого RTT.

– Коефіцієнт, використовуваний при мультиплікативному зниженні $R_S(t)$.

– Значення швидкості відправлення даних безпосередньо на виході режиму MD1.

– Значення швидкості відправлення даних безпосередньо перед входом у режим MD1.

– Значення оцінки доступної пропускної здатності в режимі MD1.

– Коефіцієнт, що визначає ймовірність збільшення швидкості в режимі FT.

– Коефіцієнт, що визначає ймовірність зниження швидкості в режимі FT.

– Точка відліку швидкості на кожному кроці в режимі FT.

– Зважене ковзне середнє значення Rs.

– Коефіцієнт критерію здійснення переходу.

– Bit error ratio – резидентна ймовірність інвертування біта на лінії.

– Початкове мінімальне значення швидкості, з якого починається ріст у стані SS.

У поле для набору тексту слід ввести повідомлення, яке треба передати по мережі за допомогою ARTCP .

Після цього слід натиснути кнопку «Генерувати пакети».

Внизу вікна буде відображатися стан мережного з'єднання та передачі пакетів.

Коротку доводку про розроблену програму та її автора можна переглянути вибравши підпункт «Про програму…» з меню «Довідка», після чого з'явиться вікно зображене на рисунку 5.2.

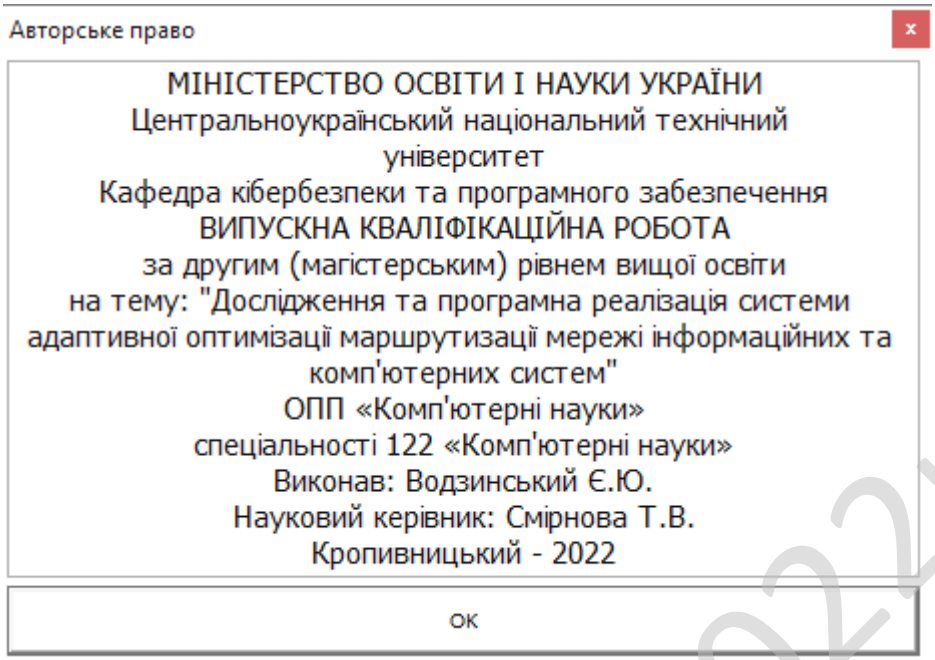

Рисунок 5.2 – Довідка про програму

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Метою розробки є дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Об'єктом дослідження є процес адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Предметом дослідження є методи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Методи дослідження базуються на методах побудови комп'ютерних мереж, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– Розроблено вітчизняний продукт адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 60 днів (три місяці).

В магістерській роботі було проведене дослідження та виконана програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	22
3. Запланований термін розробки, днів	Fpq	60 (3 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	3
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	22000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}85 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 80 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \qquad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{60 - 5} = 3,8 \text{ ставки.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	9	810	13,5
Монітор	60	9	540	9
Клавіатура	30	9	270	4,5
Маніпулятор «мишка»	10	9	90	1,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	2	40	0,67
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛВС на 1 м. п.	2,5	250	625	10,42
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	48,42

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{48 \cdot 3}{1,2} = 120 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$Ч_{ел}$ = 120/(60·8) = 0,25 ставки.

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,25
	Верстка друкованих видань	0,5	
	Додрукова підготовка макетів	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	12450	37350
Продакт-менеджер	0,25	12000	9000
Інженер-програміст	3,8	12000	136800
Інженер-електронщик	0,25	11500	8625
Інженер-системотехнік	0,25	11500	8625
Адміністратор мережі	0,5	11500	17250
Системний програміст	0,25	11500	8625
Дизайнер WEB	0,25	12000	9000
Інженер-верстальник	0,25	11700	8775
Бухгалтер-економіст	0,5	12500	18750
Всього за період розробки	$R_{сп}$ = 7,3	-	$Ф_{роб}$ = 262800

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{262800}{7,3 \cdot 60} = 600 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми HARD.kiev.ua за 1.11.22 – джерело https://hard.kiev.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		11186

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок		6490
Процесор	Intel Core i5-750 S-1156 (8M Cache, 2.6 GHz)	1638
Системна плата	MB Asrock H55M-LE s1156 (H55, s1156 DDR3 2600(OC)x, ntel(R) HD Graphics 1xPCI-E 16x, 4xSATA2, Lan 1000 Mb/s, SI 7.1) mATX	1050
Відеокарта	VC Manli GeForce GT220 PCIe-2.0 512M DDR2 128 bit HDMI DVI RTL	860
Жорсткий диск	HDD 640 Gb WD 7200 64Mb WD6400AAR SATAII	980
Оперативна пам’ять	DIMM 2048Mb DDR3 PC3-10600 Samsun 1333Mhz	468
DVD-привод	Super Multi LG SATA DVD±RW R+22x/ 22x, RW+8x/-6x, DL+16x/-12x, RAM 12x SecurDisc, black (GH22NS40RBB)	432
Корпус	ASUS TA-861 500W FSP ATX-500V (Black/Silver panel) ATX (90-PL861AF5C4 53CZ)	822
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	240
інше	Клавіатура, мишка	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3200
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37 Photo	2970
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	UPS APC BACK-UPS ES 525VA 230V RUSSIA (BE525-RS)	1496

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни. Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	11186	8948,8	98436,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2970	8948,8	98436,8
Копіюв. апарат	1	5965	540	5940
Всього	–	–	–	214803,6

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	214804	-	-
Всього по групі	214804	50	107402
Група 5, 6			
4. Вимірювальні пристрої	5190	25	-
5. Транспортні засоби	0	20	-
6. Господарський інвентар	28000	25	-
Всього по групі	33190	-	8297,5
Нематеріальні активи			
7. Нематеріальні активи	22000	10	2200
Разом	$К_р$ = 1677994		$А_р$ = 188299,5

Примітка: вартість автомобіля приймаємо рівною нулю.

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 600 \cdot 209/22 = 5700 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 5700 \cdot 10 \cdot 0{,}01 = 570 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(5700+570) = 1379 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 5700 \cdot 15 \cdot 0{,}01 = 855 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,75 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

$$З_{М1} = Ц_п \cdot n_{вит}. \quad (7.16)$$

$$З_{М1} = 200 \cdot 0,75 = 150 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо вісім):

$$З_{М2} = \sum Ц_д\,, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD – 34,50 грн./шт.

$$З_{М2} = 34,5 \cdot 8 = 276 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з.., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (150 + 276 + 1702)/22 = 97 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0,01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 5700 \cdot 15 \cdot 0,01 = 855 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 22 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 188300 \cdot 3/(22 \cdot 12) = 2140 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + C_{оц} + Г_{осп} + З_м + О_п + A_м. \quad (7.21)$$

$$C_п = 5700+570+1379+855+97+855+2140 = 11596 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 11596 = 5798 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	5700
2. Додаткова зарплата виконавців	$З_д$	570
3. Відрахування на соціальні потреби	$C_{оц}$	1379
4. Загальногосподарські витрати	$Г_{осп}$	855
5. Витрати на матеріали	$З_М$	97
6. Освоєння нових операційних систем, мов програмування	$О_п$	855

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	2140
8. Повна собівартість програмного забезпечення	$С_п$	11596
9. Плановий прибуток	$П_р$	5798
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	17394
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3478,8
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	20872,8

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	20873
Всього капітальних витрат	–	20873

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на обслуговування	$З_р$	30195	12078
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	5218
Всього витрат за рік	$І$	30195	17296

Витрати на обслуговування:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин, що витрачається на обслуговування на рік, *год.* (приймаємо 750 год.);

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

$$З_{р\,баз} = 750 \cdot 30 \cdot 1{,}1 \cdot 1{,}22 = 30195 \ грн,$$

до:

$$З_{р\,нов} = 300 \cdot 30 \cdot 1{,}1 \cdot 1{,}22 = 12078 \ грн.$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\,баз} = З_{ел\,нов}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	20873	–	5218,25
Всього відрахувань	-	–	20873	–	5218,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Ев = (17394 - 11596) \cdot 22 - (0,05 \cdot 1408000 + 0,5 \cdot 214804 +$$
$$+0,25 \cdot 33190 + 0,1 \cdot 22000) \cdot 3/12 = 80481 \; грн.$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_в = \frac{1677994}{(17394 - 11596) \cdot 21 \cdot 12 / 3} = 3 \; роки.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	22
2. Повна собівартість розробленої програми	Грн.	11596
3. Ціна розробленої програми	Грн.	17394
4. Плановий прибуток від реалізації одної програми	Грн.	5798
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1677994
7. Загальний прибуток від реалізації програмної продукції	Грн.	127556
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	80481
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	3
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	20873
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7681
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	1,6

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (30195\text{-}17296)\text{-}0{,}25 \cdot 20873 = 7681 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н}, \tag{7.28}$$

$$T_{сп} = \frac{20873}{40260 - 17296} = 1,6 \ року .$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [3] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207, який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» [5], яким затверджено нормативно-правовий акт з охорони праці НПАОП 0.00-7.15-18, «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98.

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням. До шкідливих факторів, які впливають на робитників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

Розглянемо шкідливі чинники роботи програмістів керуючись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [5], та «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» НПАОП 0.00-7.15-18.

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення негативного впливу комп'ютера на організм людини визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста.

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, параметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98.

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат;

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,38
Довжина	5,95
Висота	2,8

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	8
Об'єм, V	$м^3$	не менше 20.0	22,4

*Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин).

У зазначеному приміщенні працюють четверо людей. За даними, які наведено у табл. 8.1, та табл. 8.2, можна зробити висновок, що площа та об'єм приміщення у розрахунку на одно робоче місце програміста не відповідають нормативним вимогам ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [5], але відповідають нормативним вимогам Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» [5] та НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»). Таним чином можна зробити висновок, що санітарно-гігієнічні умови праці на робочому місці програміста відповідають вимогам.

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови № 42 від 01.12.1999 Головного державного санітарного лікаря України, робота, виконувана в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря в приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура, °С	Воло-гість,%	Швидкість повітря, м/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	40-55	0,1
Тепла	23-25	50-70	0,1	24-25	50-65	0,11

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP 1100, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

З 2019 року діють Державні будівельні норми України "Природне і штучне освітлення" – ДБН В.2.5-28:2018 [1], у яких прописані вимоги до використання всіх освітлювальних приладів, у т.ч. світлодіодних.

Працю працівника, який постійно працює за комп'ютером, згідно ДБН В.2.5-28:2018 [1], можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою

контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 Лк. [1], Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до

колективного договору мінімально можливого вмісту аптечок з обов'язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору ланцюга).

Регулярна наочне знайомство персоналу із шляхами для евакуації людей із приміщення відповідно до плану евакуації, забезпечення розподільних щитів спеціальними розетками з заземлюючими контактами; організація заземлення всіх приладів і пристроїв, які працюють при напрузі вище 36 В.

Так як при ураженні електричним струмом у людини може статися фібриляція шлуночків серця, в організації бажано мати дефібрилятор і підготовлений персонал для роботи з ним.

8.5 Розрахункова частина

Початкові дані для розрахунку захисного штучного заземлення: опір заземлювача, який нормується: $R_{ЗН} = 4$ *Ом*.

Для захисного штучного заземлення застосовуються вертикальні електроди з металевого прутка діаметром 30 мм. (*D=30 мм.=0,03 м.)* довжиною *L*=2,5 *м.* та горизонтальний електрод – металева полоса з перетином 40*4 мм. тип грунта – глина (питомий опір 40 Ом*м). Відстань між вертикальними заземлювачами (електродами) *A*=3 *м*.

Глибина закладення горизонтального контура заземлення *t= 0,8 м*.

Умовна товщина верхнього шару грунта: *H=0,4 м*. Напруга – 220/380 В. Розрахункова схема розташування заземлюючих електродів – у ряд.

Розрахунок проводиться за допустимим опором розтіканню струму заземлювача.

Необхідно визначити необхідну кількість вертикальних заземлювачів та довжину полоси (горизонтального заземлювача).

Розрахунок захисного заземлення можна автоматизувати за допомогою програми, сирцевий код якої опублікован на стр. 13-16 [6], або аналогічної.

Розрахунок.

Відстань від центра вертикального заземлювача до поверхні землі:

$$T=t+L/2=0,8+2,5/2=2,045 \text{ м.}$$

Розрахунковий питомий опір грунта (з врахуванням того, що фактично вся конструкція заземлювача розташовується у нижньому шарі грунта):

$$\rho = \psi \rho_2 = 1,36*40=54,55 \text{ Ом*м.}$$

де:

$\psi = 1,36$ – табличне значення коефіцієнта сезонності для відповідної климатичної зони у багатошаровому грунті [8];

$\rho_1 = 50$ Ом*м. – табличне значення питомого опору верхнього шару грунта [6];

$\rho_2 = 40$ Ом*м. – табличне значення питомого опору нижнього шару грунта [8].

Опір розтіканню електричного струму одного електрода вертикального заземлювача [8]:

$$R_o = 0,366\frac{\rho}{L}\left(\lg\frac{2L}{D}+\frac{1}{2}\lg\frac{4T+L}{4T-L}\right) = 0,366\frac{54,5}{2,5}\left(\lg\frac{2\cdot 2,5}{0,03}+\frac{1}{2}\lg\frac{4\cdot 2,05+2,5}{4\cdot 2,05-2,5}\right) = 20,1 \text{ Ом.}$$

Відношення $A/L=3/2,5=1,2$.

Визначаємо коефіцієнт екранування вертикальних електродів $Кев=0,8$ при попередній (орієтовній) кількості вертикальних електродів, яке дорівнює 4 [8].

Визначаємо необхідну кількість вертикальних заземлювачів (без врахування горизонтального заземлювача), при $R_{ЗН} = 4$ Ом :

$$N=R_O /(Кев\ R_{ЗН})= 20,1 / (0,8*4)=5,87 \approx 6 \text{ шт.}$$

Визначаємо довжину з'єднуючої полоси [8]:

$$L_П =1,05*A*N= 1,05*3*6=18,8\approx 19 \text{ м.}$$

Опір розтіканню електричного струму з’єднуючої полоси [8]:

$$R_П = 0,366(\rho_2 * K_П / L_П) lg(2(L_П * L_П)/(К * t)) =$$

$$= 0,366(40*5/16) * [lg(2*16*16)/(0,04*0,8)] = 16,5 \text{ Ом}.$$

де $K_П = 5$ – табличне значення коефіцієнта сезонності для відповідної климатичной зони з’єднуючої полоси: [8].

Загальний опір розтіканню електричного струму заземлювача [8]:

$$R = (R_О * R_П)/(R_О * \eta_П + N * R_П * Кев) =$$

$$= (20,1 * 16,5)/(20,1 * 0,75 + 6 * 16,5 * 0,8) = 3,32 \text{ Ом}.$$

де $\eta_П = 0,75$ – табличне значення коефіцієнта екранування з’єднуючої полоси [8].

Умова $R \leq R_{ЗН}$ виконується ($3,32 \leq 4$).

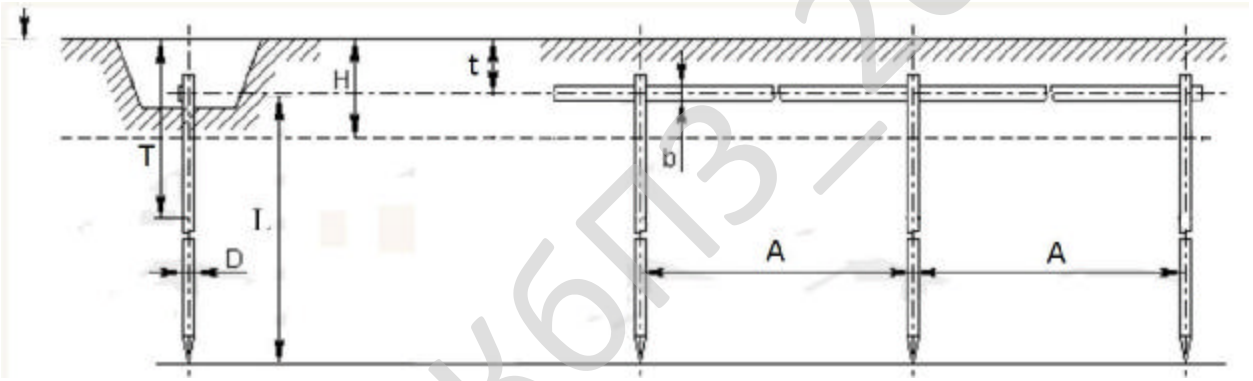

Рисунок 8.1 – Схема штучного заземлення

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров’я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного заземлення. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– Досліджена система адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

– На основі отриманих результатів досліджень створена програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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 Embarcadero Delphi. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 AES.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 7681 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,6 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Водзинський Є.Ю. Дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. 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, pp. 1-12. **(Scopus).**

3. 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. (**Scopus**).

4. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». *In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies*, vol 131. 2023. **Springer**, Singapore. pp. 21-34. **(Scopus).**

5. Smirnov O.A., Al-Oraiqat A.M., Ulichev O.S., Meleshko Ye.V., Al-Rawashdeh H.S., Polishchuk L.I. «Modeling strategies for information influence dissemination in social networks». *Journal of Ambient Intelligence and Humanized Computing* Volume 13, Issue 5. **Springer**, Cham. 2022, pp. 2463-2477. **(Scopus).**

6. 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 **(Scopus).**

7. Smirnov O., Kovalenko O., Kovalenko A., Kavun S. «Quantitative Risk Assessment Method Development in the Context of the SDLC-model». *2021 IEEE 8th*

International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), 2021, pp. 203-208, doi: 10.1109/PICST54195.2021.9772143 **(Scopus).**

8. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». *CEUR Workshop Proceedings* Volume 3101, 2021, Pages 192-207. **(Scopus).**

9. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova K. «Data hiding scheme based on spread sequence addressing». *CEUR Workshop Proceedings* Volume 2805, 2020, Pages *44-58.* **(Scopus)**.

10. 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. **(Scopus)**.

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

12. Smirnov O.A., 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. **(Scopus)**.

13. Smirnov O., Kuznetsov A., Arischenko A., Chepurko I., Onikiychuk A., Kuznetsova T. «Pseudorandom sequences for spread spectrum image steganography». *CEUR Workshop Proceedings* Volume 2654, 2020, Pages 122-131. **(Scopus)**.

14. Tanenbaum A.S. Computer Networks. Third edition, Prentice-Hall, New Jersey, 1996.

15. Jacobson V. Congestion Avoidance and Control. // ACM SIGCOMM'88. 1988.

16. Holzmann G. Design and Validation of Computer Protocols. Prentice Hall, New Jersey, 1991.

17. Postel J. Transmission Control Protocol. // RFC793 (STD7). 1981.

18. Braden R. T. Requirements for Internet Hosts – Communication Layers. // RFC1122. 1989.

19. Jacobson V., Braden R., Borman D. TCP Extensions for High Performance. // RFC1323. 1992.

20. Karn P., Partridge C. Estimating Round-trip Times in Reliable Transport Protocols. // ACM SIGCOMM'87. 1987.

8. Nagle J. Congestion Control in IP/TCP Networks. // ARPANET Working Group Requests for Comment (RFC-896), DDN Network Information Center, SRI International, Menlo Park, CA. 1984.

21. Бертсекас Д., Галлагер Р. Сети передачи данных. Пер. с англ. М., Мир. 1989.

22. George F., Young G. SNA Flow Control: Architecture and Implementation. // IBM System Journal, vol. 21, no. 2. – 1982. – pp. 179-210.

23. Digital Equipment Corporation. DECnet Digital Network Architecture [phase IV] General Description. // Order AA-N149A-TC, Digital Equipment Corporation. 1982.

24. Postel J. B., Sunshine C. A., and Cohen D. The ARPA Internet Protocol. // Computer Networks, vol. 5, no. 4. – 1981. – pp. 171-261.

25. Yang C., Reddy A. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks. // IEEE Network Magazine. Vol. 9, Number 5. – 1995.

26. Brakmo L., O'Malley S., Peterson L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. // ACM SIGCOMM. -1994.– pp. 24-35.

27. Brakmo L., Peterson L. TCP Vegas: End to End Congestion Avoidance on a Global Internet. // IEEE Journal on Selected Areas in Communications, 13(8). -1995.

28. Lefelhocz C., Lyles B., Shenker S., Zhang L. Congestion Control for Best-Effort Service: Why We Need a New Paradigm. // IEEE Network, Vol. 10, N. 1. -1996.

29. Braden. B., Clark D., Crowfort J., Deering S., Estrin D. Recommendations in Queue Management and Congestion Avoidance in the Internet. // RFC 2309 -1998.

30. Floyd S., Jacobson V. Random Early Detection gateways for Congestion Avoidance. // IEEE/ACM Transactions on Networking. Vol.1, N.4. -1993.– p. 397-413.

31. Brakmo L., Peterson L. Performance Problems in BSD4.4 TCP. // ACM Computer Communications Review. 25(5). -1995.– p. 69-86.

32. Caceres R., Danzig P., Jamin S., Mitzel D. Characteristics of Wide-Area TCP/IP Conversations. // ACM SIGCOMM'91. -1991.

33. Fall K., Floyd S. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. // Computer Communications Review. July -1996.

34. Tomey C. Rate-Based Congestion Control Framework for Connectionless Packet-Switched Networks. // Doctor of Philosophy Thesis. University of New South Wales Australian Defence Force Academy. -1997.

35. Benmohamed L., Meerkov S. M. Feedback Control of Congestion in Packet Switched Networks: The Case of a Single Congested Node. // IEEE Transactions on Networking. Vol. 1, No. 6. -1993.– p. 693-707.

36. Charney A. An Algorithm for Rate Allocation in a Packet-Switched Network with Feedback. // M.Sc. thesis. Department of EECS. MIT. -1994.

37. Ramakrishnan K., Jain R. A Binary Feedback Scheme for Congestion Avoidance in Computer Networks. // ACM Transactions on Computer Systems. Vol. 8, No. 2. -1990.– p. 158-181.

38. Keshav S. The Packet Pair Flow Control Protocol. // ICSI Tech. Rept. TR-91-028. Computer Science Division, Department of EECS, University of

California, Berkeley and International Computer Science Institute. Berkeley, CA. May 1991.

39. Bennett J., Zhang H. Hierarchical Packet Fair Queueing Algorithms. // ACM SIGCOMM'96. Aug 1996.

40. Demers A., Keshav S., Shenker S. Analysis and Simulation of a Fair Queueing Algorithm. // ACM SIGCOMM'89. -1989.– p. 3-12.

41. Floyd S., Jacobson V. Link Sharing and Resource Management Models for Packet Networks. // IEEE/ACM Transactions on Networking. 3(4). -1995.

42. Алексеев И.В. Интегрированные услуги нового поколения Internet. // Сети. № 10. -1999.– с. 102-108.

43. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

44. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

45. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

46. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

47. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

48. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

49. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

50. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

51. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

52. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 18-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи адаптивної оптимізації маршрутизації мережі інформаційних та комп'ютерних систем;

– 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 архітектури IBM PC, 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 Embarcadero Delphi.

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 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 131 аркуш.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 20.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнова Т.В.

Дослідження та програмна реалізація системи адаптивної оптимізації маршрутизації мережі інформаційних та комп’ютерних систем

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 27

Літера: РП

Кропивницький – 2022 року

Файл PKT_GEN.DPR – головний файл проекту

```
program Pkt_Gen;

uses
  Forms,
  UPkt_Gen in 'UPKT_GEN.PAS' {Form1},
  artcpiphlp in 'artcpiphlp.pas',
  UPkt_Snd in 'UPKT_SND.PAS',
  About in About.PAS'; {Form2}

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(Tform2, Form2);
  Application.Run;
end.
```

Файл PKT_GEN.PAS – генерація ARTCP-пакетів

```
unit UPkt_Gen;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, ComCtrls, UPkt_Snd, ArtcpIpHlp,
  Menus, About;

type
  TForm1 = class(TForm)
    TopPanel: TPanel;
    MainPanel: TPanel;
    GroupBox2: TGroupBox;
    DstIpEdit: TEdit;
    SrcIpEdit: TEdit;
    TtlEdit: TEdit;
    RandomSrcIP: TCheckBox;
    RandomDstIP: TCheckBox;
    SrcPortEdit: TEdit;
    DstPortEdit: TEdit;
    SequenceEdit: TEdit;
    FlagsGroupBox: TGroupBox;
    FinFlag: TCheckBox;
    SynFlag: TCheckBox;
    RstFlag: TCheckBox;
    PshFlag: TCheckBox;
    AckFlag: TCheckBox;
    UrgFlag: TCheckBox;
    RandomSrcPort: TCheckBox;
    RandomDstPort: TCheckBox;
    UseRandomSeq: TCheckBox;
    DataMemo: TMemo;
    NumPacketsEdit: TEdit;
    DelayEdit: TEdit;
    ResultMemo: TMemo;
    GoButton: TButton;
    TabControl: TTabControl;
    SrcIPLabel: TLabel;
    DstIPLabel: TLabel;
    TTLLabel: TLabel;
    SrcPortLabel: TLabel;
    DstPortLabel: TLabel;
    SeqNoLabel: TLabel;
    Label1: TLabel;
    Label2: TLabel;
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    procedure RandomSrcIPClick(Sender: TObject);
    procedure RandomDstIPClick(Sender: TObject);
    procedure RandomSrcPortClick(Sender: TObject);
    procedure RandomDstPortClick(Sender: TObject);
    procedure UseRandomSeqClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure GoButtonClick(Sender: TObject);
    procedure TabControlChange(Sender: TObject);
  private
    { Private declarations }
    FRandomSrcARTCP, { save checkmarks for ARTCP,UDP and Ping }
    FRandomDstARTCP,
    FRandomSrcUDP,
    FRandomDstUDP,
```

```
    FRandomIdPing,
    FRandomSqPing: Boolean;
    FProtocolType: TProtocolType; { protocol tab }

    function  AcceptUserInput(Sender: TSenderIP; x: TProtocolType): Boolean;
    function  AcceptIPSettings(ip: TSenderIP): Boolean;
    function  AcceptArtcpSettings(artcp: TSenderARTCP): Boolean;
    function  AcceptUdpSettings(udp: TSenderUDP): Boolean;
    function  AcceptPingSettings(ping: TSenderICMP): Boolean;
    procedure PrintLine(s: String);

  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

var WarnedAboutW2k: Boolean = FALSE;

procedure WarnAboutW2k;
begin
  if NOT WarnedAboutW2k then
  begin
    WarnedAboutW2k := TRUE;
    if NOT Win2KDetected then
      ShowMessage('Попередження: Цей додаток розроблено під Windows 2000/XP, '
                  +'на вашому комп"ютері встановлена інша операційна система. '
                  +'Тому можуть виникати помилки у роботі з сокетами, '
                  +'так як для їх коректної роботи потрібний MS Winsock');
  end
end;

procedure TForm1.PrintLine(s: String);
// вивід рядка тексту
begin
  Form1.ResultMemo.Lines.Add(s)
end;

function TForm1.AcceptIPSettings(ip: TSenderIP): Boolean;
// Перевірка установочних параметрів IP заданих користувачем
begin
  Result := FALSE;

  ip.UseRandomSrcIP   := RandomSrcIP.Checked;
  ip.UseRandomDstIP   := RandomDstIP.Checked;
  ip.Data             := DataMemo.Text;

  ip.DelayBetweenPackets := StrToIntDef(DelayEdit.Text, 1000);
  DelayEdit.Text := IntToStr(ip.DelayBetweenPackets);

  ip.NumPackets := StrToIntDef(NumPacketsEdit.Text, 1);
  NumPacketsEdit.Text := IntToStr(ip.NumPackets);

  ip.TimeToLive := StrToIntDef(TtlEdit.Text, 1);
  TtlEdit.Text := IntToStr(ip.TimeToLive);

  if NOT ip.UseRandomSrcIP then
  begin
    try
      ip.SourceHost := SrcIpEdit.Text;
    except
      ShowMessage('Невідомий хост відправника: '+ SrcIpEdit.Text);
      Exit;
    end;
```

```
  end;

  if NOT ip.UseRandomDstIP then
  begin
    try
      ip.DestinationHost := DstIpEdit.Text;
    except
      ShowMessage('Невідомий хост одержувача: '+ DstIpEdit.Text);
      Exit;
    end;
  end;

  Result := TRUE;
end;

function TForm1.AcceptArtcpSettings(artcp: TSenderARTCP): Boolean;
// Перевірка установочних параметрів ARTCP заданих користувачем
VAR x: DWORD;
begin
  Result := AcceptIPSettings(artcp);
  if Result then
  begin
    artcp.UseRandomSrcPort := RandomSrcPort.Checked;
    artcp.UseRandomDstPort := RandomDstPort.Checked;

    artcp.FinFlag := FinFlag.Checked;
    artcp.SynFlag := SynFlag.Checked;
    artcp.RstFlag := RstFlag.Checked;
    artcp.PshFlag := PshFlag.Checked;
    artcp.AckFlag := AckFlag.Checked;
    artcp.UrgFlag := UrgFlag.Checked;

    if NOT RandomSrcPort.Checked then
    begin
      x := StrToIntDef(DstPortEdit.Text, 80);
      if x < 1 then
      begin
        x := 80;
        SrcPortEdit.Text := '80';
      end;
      artcp.SourcePort := x;
    end;


    if NOT RandomDstPort.Checked then
    begin
      x := StrToIntDef(DstPortEdit.Text, 80);
      if x < 1 then
      begin
        x := 80;
        DstPortEdit.Text := '80';
      end;
      artcp.DestinationPort := x;
    end;

    x := DWORD(StrToIntDef(SequenceEdit.Text, 12345678));
    if (x < 1) OR UseRandomSeq.Checked then
      x := DWORD(Random(MaxInt-1)+1);
    SequenceEdit.Text := IntToStr(x);
    artcp.SequenceNumber := x;
  end;
end;

function TForm1.AcceptUdpSettings(udp: TSenderUDP): Boolean;
// Перевірка установочних параметрів UDP заданих користувачем
begin
  Result := AcceptIPSettings(udp);
  if Result then
  begin
```

```
    udp.UseRandomSrcPort := RandomSrcPort.Checked;
    udp.UseRandomDstPort := RandomDstPort.Checked;

    if NOT RandomSrcPort.Checked then
      udp.SourcePort := StrToIntDef(DstPortEdit.Text, 0);

    if NOT RandomDstPort.Checked then
      udp.DestinationPort := StrToIntDef(DstPortEdit.Text, 0);
  end;
end;

function TForm1.AcceptPingSettings(ping: TSenderICMP): Boolean;
// Перевірка установочних параметрів ICMP/Ping заданих користувачем
begin
  Result := AcceptIPSettings(ping);
  if Result then
  begin
    ping.UseRandomPingID := RandomSrcPort.Checked;
    ping.UseRandomPingSequence := RandomDstPort.Checked;

    if NOT RandomSrcPort.Checked then
      ping.PingID := StrToIntDef(DstPortEdit.Text, $1234);
    if NOT RandomDstPort.Checked then
      ping.PingSequence := StrToIntDef(DstPortEdit.Text, $4321);
  end;
end;

function  TForm1.AcceptUserInput(Sender: TSenderIP; x: TProtocolType): Boolean;
begin
  ResultMemo.Clear;  { clear the results field }

  Sender.OnPrintLine := PrintLine; { progress feedback }

  // Перевірка установочних параметрів протоколів заданих користувачем
  // в залежності від обраного типу протоколу
  //
  case x of
    ptUDP:  Result := AcceptUdpSettings(TSenderUDP(Sender));
    ptICMP: Result := AcceptPingSettings(TSenderICMP(Sender));
    else    Result := AcceptArtcpSettings(TSenderARTCP(Sender));
  end
end;

procedure TForm1.RandomSrcIPClick(Sender: TObject);
begin
  SrcIpEdit.Enabled := NOT RandomSrcIP.Checked
end;

procedure TForm1.RandomDstIPClick(Sender: TObject);
begin
  DstIpEdit.Enabled := NOT RandomDstIP.Checked
end;

procedure TForm1.RandomSrcPortClick(Sender: TObject);
begin
  SrcPortEdit.Enabled := NOT RandomSrcPort.Checked;
  case FProtocolType of
    ptUDP:  FRandomSrcUDP := RandomSrcPort.Checked;
    ptICMP: FRandomIdPing := RandomSrcPort.Checked;
    else    FRandomSrcARTCP := RandomSrcPort.Checked;
  end;
end;

procedure TForm1.RandomDstPortClick(Sender: TObject);
begin
  DstPortEdit.Enabled := NOT RandomDstPort.Checked;
  case FProtocolType of
    ptUDP:  FRandomDstUDP := RandomDstPort.Checked;
    ptICMP: FRandomSqPing := RandomDstPort.Checked;
```

```
    else    FRandomDstARTCP := RandomDstPort.Checked;
  end;
end;

procedure TForm1.UseRandomSeqClick(Sender: TObject);
begin
  SequenceEdit.Enabled := NOT UseRandomSeq.Checked
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  FRandomSrcARTCP := TRUE;
  UseRandomSeq.Checked := TRUE;
  TabControl.OnChange(Sender); { init tab control w/ARTCP settings }
end;

procedure TForm1.GoButtonClick(Sender: TObject);
VAR x: TSenderIP;
begin


  // Заборона вводу користувача
  //
  TopPanel.Enabled := FALSE;
  MainPanel.Enabled := FALSE;
  Application.ProcessMessages;

  try

    WarnAboutW2k;      // для Win2K

    // створюємо специфічний обєкт користувача (ARTCP, UDP
    // або ICMP/Ping) в залежності від обраного протоколу


    case FProtocolType of
      ptUDP:  x := TSenderUDP.Create;
      ptICMP: x := TSenderICMP.Create;
      else begin
        FProtocolType := ptARTCP;
        x := TSenderARTCP.Create;
      end;
    end;
    x.OnExecute := AcceptUserInput;

    try

      // перевіряємо та приймаємо введення користувача
      // та відправляємо один або декілька пакетів у мережу.
      //
      x.Execute;
    finally
      x.Free;
    end
  finally
    MainPanel.Enabled := TRUE;
    TopPanel.Enabled := TRUE;
  end
end;

procedure TForm1.TabControlChange(Sender: TObject);
begin
  // якщо протокол змінюєжться, коректується заголовок
  //
  case TabControl.TabIndex of
    1:   { UDP }
      begin
        FProtocolType := ptUDP;
        GoButton.Caption := 'Генерація пакетів';
```

```
        SrcPortLabel.Caption  := 'Порт відправника: ';
        DstPortLabel.Caption  := 'Порт одержувача: ';
        RandomSrcPort.Checked := FRandomSrcUDP; { restore checkmarks }
        RandomDstPort.Checked := FRandomDstUDP;
        SeqNoLabel.Visible    := FALSE;
        SequenceEdit.Visible  := FALSE;
        UseRandomSeq.Visible  := FALSE;
        FlagsGroupBox.Visible := FALSE;
      end;
    2:   { ICMP/Ping }
      begin
        FProtocolType := ptICMP;
        GoButton.Caption := 'Генерація пакетів';
        SrcPortLabel.Caption  := 'ID: ';
        DstPortLabel.Caption  := 'Послідовність: ';
        RandomSrcPort.Checked := FRandomIdPing;
        RandomDstPort.Checked := FRandomSqPing;
        SeqNoLabel.Visible    := FALSE;
        SequenceEdit.Visible  := FALSE;
        UseRandomSeq.Visible  := FALSE;
        FlagsGroupBox.Visible := FALSE;
      end;
    else { ARTCP }
      begin
        FProtocolType := ptARTCP;
        GoButton.Caption := 'Генерація пакетів';
        SrcPortLabel.Caption  := 'Порт відправника: ';
        DstPortLabel.Caption  := 'Порт одержувача: ';
        RandomSrcPort.Checked := FRandomSrcARTCP;
        RandomDstPort.Checked := FRandomDstARTCP;
        SeqNoLabel.Visible    := TRUE;
        SequenceEdit.Visible  := TRUE;
        UseRandomSeq.Visible  := TRUE;
        FlagsGroupBox.Visible := TRUE;
      end;
  end
end;

end.
```

Файл UPKT_SND.PAS – відправка ARTCP-пакетів

```
unit UPkt_Snd;

interface

uses
  Windows, SysUtils, Forms, Winsock, ArtcpIpHlp;

type
  TSenderIP = class;
  TProtocolType = (ptARTCP, ptUDP, ptICMP);
  TOnExecuteEvent = function (Sender: TSenderIP; x: TProtocolType): Boolean of
Object;
  TOnPrintLineEvent = procedure (s: String) of Object;

  TSenderIP = class(TObject)  // IP клас пакета передавача
  private
    FProtocol: Word;
    FSocket: TSocket;

    FNumPackets: Word;
    FMaxPackets: Word;
    FSrcIP, FDstIP: DWORD;
    FSourceHost: String;
    FDestinationHost: String;
    FDelay: DWORD;

    FTimeToLive: Byte;

    FRandomSrcIP,
    FRandomDstIP: Boolean;

    FData: String;
    FDataLen: Integer;

    FOnExecute: TOnExecuteEvent;
    FOnPrintLine: TOnPrintLineEvent;

    function  GetLength: Word; virtual; abstract;
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
virtual; abstract;
    procedure UpdateSequence; virtual;

    procedure PrintLine(s: String);
    procedure ThrowException(msg: String; eCode: Integer);

    procedure SetNumPackets(value: Word);
    procedure SetSrcIP(value: String);
    procedure SetDstIP(value: String);
    procedure SetDelay(value: DWORD);
    procedure SetData(value: String);

    procedure SendPackets;
    procedure SendOnePacket;
    function  SendToRemote(addr, port: Integer;
                           buffer: PChar; len: Integer): Integer;

   {$ifdef DUMP_MODE}
    procedure Dump(data: PByte; len: Integer);
   {$endif}

  protected
    FRandomSrcPort,
    FRandomDstPort: Boolean;
    FSrcPort, FDstPort: Word;
    FIpBuffer: Array [1..$FFFF] of Char; // IP пакет не більше 64K
```

```
  public
    constructor Create; virtual;
    destructor Destroy; override;
    procedure Execute;

    property  NumPackets: Word read FNumPackets write SetNumPackets;
    property  SourceHost: String read FSourceHost write SetSrcIP;
    property  DestinationHost: String read FDestinationHost write SetDstIP;
    property  DelayBetweenPackets: DWORD read FDelay write SetDelay;
    property  TimeToLive: Byte read FTimeToLive write FTimeToLive;

    property  UseRandomSrcIP: Boolean read FRandomSrcIP write FRandomSrcIP;
    property  UseRandomDstIP: Boolean read FRandomDstIP write FRandomDstIP;

    property  Data: String read FData write SetData;

    property  OnExecute: TOnExecuteEvent read FOnExecute write FOnExecute;
    property  OnPrintLine: TOnPrintLineEvent read FOnPrintLine write
FOnPrintLine;
  end;

  TSenderARTCP = class(TSenderIP)  // ARTCP пакет передавача
  private
    FSequence: DWORD;
    FFinFlag, { ARTCP flags }
    FSynFlag,
    FRstFlag,
    FPshFlag,
    FAckFlag,
    FUrgFlag: Boolean;

    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
    procedure UpdateSequence; override;
  public
    constructor Create; override;

    property  SequenceNumber: DWORD read FSequence write FSequence;
    property  FinFlag: Boolean read FFinFlag write FFinFlag;
    property  SynFlag: Boolean read FSynFlag write FSynFlag;
    property  RstFlag: Boolean read FRstFlag write FRstFlag;
    property  PshFlag: Boolean read FPshFlag write FPshFlag;
    property  AckFlag: Boolean read FAckFlag write FAckFlag;
    property  UrgFlag: Boolean read FUrgFlag write FUrgFlag;

    property  SourcePort: Word read FSrcPort write FSrcPort;
    property  DestinationPort: Word read FDstPort write FDstPort;
    property  UseRandomSrcPort: Boolean read FRandomSrcPort write
FRandomSrcPort;
    property  UseRandomDstPort: Boolean read FRandomDstPort write
FRandomDstPort;
  end;

  TSenderUDP = class(TSenderIP)  // UDP пакет передавача
  private
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
  public
    constructor Create; override;

    property  SourcePort: Word read FSrcPort write FSrcPort;
    property  DestinationPort: Word read FDstPort write FDstPort;
    property  UseRandomSrcPort: Boolean read FRandomSrcPort write
FRandomSrcPort;
    property  UseRandomDstPort: Boolean read FRandomDstPort write
FRandomDstPort;
```

```
  end;

  TSenderICMP = class(TSenderIP)  // Ping Пакет передавача
  private
    function  SendDatagram(ih: THdrIP; data: PChar; dlen: Integer): Integer;
override;
    function  GetLength: Word; override;
  public
    constructor Create; override;

    property  PingID: Word read FSrcPort write FSrcPort;
    property  PingSequence: Word read FDstPort write FDstPort;
    property  UseRandomPingID: Boolean read FRandomSrcPort write FRandomSrcPort;
    property  UseRandomPingSequence: Boolean read FRandomDstPort write
FRandomDstPort;
  end;

implementation

type
  // ARTCP/UDP псевдозаголовок
  // (використовується для підрахунку контрольної суми)
  //
  TPseudoHeader = packed record
    saddr, daddr: DWORD;
    zero, protocol: BYTE;
    packetlen: WORD;
  end;

procedure DelayMS(ms: Integer);
//
// DelayMS() - затримка в мілісекундах.
// можливо використання стандартної функції sleep(),
// але вона дуже довго працює й складається враження
// що додаток підвис.
// написане DelayMS() викликає Application.ProcessMessages.
//
var StartTime: LongInt;
begin
  StartTime := GetTickCount;
  repeat
    Application.ProcessMessages;
  until (LongInt(GetTickCount)-StartTime >= ms) or (Application.Terminated)
end;

procedure RaiseException(msg: String; eCode: Integer);
//
// форматуємо повідомлення для передачі по протоколу
//
  function AdditionalMessage: String;
  begin
    Result := SysErrorMessage(eCode);
    if Result <> '' then Result := ': ' + Result
  end;
begin
  if eCode = 0 then
    raise Exception.Create(msg)
  else
    raise Exception.Create('Помилка: '+msg+' [SocketError '+IntToStr(eCode)
                           +AdditionalMessage+']')
end;

{  TSenderIP }

constructor TSenderIP.Create;
begin
  FSocket := INVALID_SOCKET;
  Randomize  { використовуємо стандартний генератор випадкових чисел }
end;
```

```
destructor TSenderIP.Destroy;
begin
  CleanupWinsock(FSocket);
  inherited
end;

procedure TSenderIP.ThrowException(msg: String; eCode: Integer);
begin
  if eCode <> 0 then CleanupWinsock(FSocket);
  RaiseException(msg, eCode)
end;

procedure TSenderIP.Execute;
VAR opt: Integer;
    errStr: String;
    x: Boolean;
begin
  // ініціалізуємо WinSock. Працюємо з Winsock version 2.
  //
  errStr := InitWinsock(2,2);
  if errStr <> '' then  ThrowException(errStr, 0);

  if Assigned(FOnExecute) then
  begin
    case FProtocol of
       IPPROTO_UDP:   x := FOnExecute(Self, ptUDP);
       IPPROTO_ICMP:  x := FOnExecute(Self, ptICMP);
       else           x := FOnExecute(Self, ptARTCP);
    end;
    if NOT x then
    begin
      CleanupWinsock;
      Exit
    end
  end;

  FSocket := socket(AF_INET, SOCK_RAW, IPPROTO_RAW);

  if (FSocket = INVALID_SOCKET) then
    ThrowException('Failed to open a RAW socket', WSAGetLastError());

  // З Winsock 2 IP_HDRINCL закриваємо сокети
  //
  // IP_HDRINCL працює на IPPROTO_IP рівні
  //
  opt := 1;
  if (setsockopt(FSocket, IPPROTO_IP, IP_HDRINCL, PChar(@opt), sizeof(opt)) =
SOCKET_ERROR) then
    ThrowException('Помилка встановленя опцій сокетів ', WSAGetLastError())
  else begin
    SendPackets; {... }
    CleanupWinsock(FSocket);
    PrintLine('Done.');
  end
end;

procedure TSenderIP.SetNumPackets(value: Word);
begin
  FNumPackets := value;
  if FNumPackets = 0 then  //
    FMaxPackets := MaxWord
  else
    FMaxPackets := FNumPackets
end;

procedure TSenderIP.SetData(value: String);
begin
  FData := value;
```

```
  FDataLen := Length(FData)
end;

procedure TSenderIP.SetDelay(value: DWORD);
begin
  // Встановлюємо мінімальну затримку 50 ms
  //
  if value < 50 then  value := 50;
  FDelay := value
end;

procedure TSenderIP.SetSrcIP(value: String);
begin
  FSrcIP := ResolveHostAddress(value);
  if FSrcIP = DWORD(-1) then
  begin
    FSourceHost := '';
    ThrowException('Невідомий хост відправника: '+ value, 0)
  end
  else begin
    FSourceHost := value;
    PrintLine('Невідомий хост відправника: '+value+'...');
  end;
end;

procedure TSenderIP.SetDstIP(value: String);
begin
  FDstIP := ResolveHostAddress(value);
  if FDstIP = DWORD(-1) then
  begin
    FDestinationHost := '';
    ThrowException('Невідомий хост одержувача: '+ value, 0)
  end
  else begin
    FDestinationHost := value;
    PrintLine('Невідомий хост одержувача: '+value+'...');
  end;
end;

procedure TSenderIP.SendPackets;
  function GenerateFakeIP: String;
  VAR a, b, c, d: Integer;
  begin
    a := random(239)+1; // a >= 240 викликає помилку сокета
    b := random(255);
    c := random(255);
    d := random(255);
    Result := Format('%d.%d.%d.%d', [a, b, c, d]);
  end;
VAR srcIP, dstIP: String;
    i: Integer;
begin
  for i := 1 to FMaxPackets do
  begin
    if FRandomSrcPort then { використовуємо вищі вихідні порти }
      FSrcPort := random(5000-1024) + 1024;

    if FRandomDstPort then { використовуємо порти }
      FDstPort := random($FFFF);

    srcIP := FSourceHost;
    dstIP := FDestinationHost;

    if FRandomSrcIP then
    begin
      srcIP := GenerateFakeIP;
      FSrcIP := ResolveHostAddress(srcIP);
    end;
```

```
    if FRandomDstIP then
    begin
      dstIP := GenerateFakeIP;
      FDstIP := ResolveHostAddress(dstIP);
    end;

    PrintLine('Відправка: '+ srcIP+ ':'+ IntToStr(FSrcPort)+ ' -> '
                          + dstIP+ ':'+ IntToStr(FDstPort));

    Application.ProcessMessages;
    SendOnePacket;
    if i < FMaxPackets then  DelayMS(FDelay);
  end;
end;

procedure TSenderIP.SendOnePacket;
VAR errCode: Integer;
    ih: THdrIP;
    id: WORD;
begin
  // Створюємо IP заголовок
  id := random($FFFF);

  FillChar(ih, sizeof(ih), 0);// очищуємо
  SetIHver(ih, 4);            // IP v 4
  SetIHlen(ih, sizeof(ih));   // IP довжина заголовка

  ih.tot_len  := htons(GetIHlen(ih) + GetLength); // загальна довжина
  ih.id       := htons(id);   // ID
  ih.ttl      := FTimeToLive; // час передачі
  ih.protocol := FProtocol;   // тип протоколу
  ih.saddr    := FSrcIP;      // джерело   IP
  ih.daddr    := FDstIP;      // розташування IP
  ih.check    := CalculateChecksum(@ih, GetIHlen(ih)); // підраховуємо IP
контрольну суму

  errCode     := SendDatagram(ih, PChar(FData), FDataLen);

  if errCode = 0 then
    UpdateSequence
  else
    ThrowException('Помилка відправки датаграми',  errCode)
end;

procedure TSenderIP.UpdateSequence;
begin
  { нічого не робить }
end;

function  TSenderIP.SendToRemote(addr, port: Integer;
                                 buffer: PChar;
                                 len: Integer): Integer;
VAR remote: TSockAddrIn;
begin
  Result := 0;

 {$ifdef DUMP_MODE}
  // dump() - допомогоа при дебазі...
  dump(buffer, len);
 {$endif}

  FillChar(remote, sizeof(remote), 0);
  remote.sin_family      := AF_INET;    { Internet домен }
  remote.sin_port        := port;
  remote.sin_addr.s_addr := addr;

  // відправляємо датаграму
  if (sendto(FSocket, buffer^, len, 0, TSockAddr(remote), sizeof(remote)) =
SOCKET_ERROR) then
```

```
    Result := WSAGetLastError()
end;

procedure TSenderIP.PrintLine(s: String);
begin
  if Assigned(FOnPrintLine) then FOnPrintLine(s)
end;

{$ifdef DUMP_MODE}
VAR howmany: Integer = 1;   // тільки дамп
procedure TSenderIP.Dump(data: PByte; len: Integer);
VAR i: Integer;
    s: String;
begin
  if howmany = 0 then Exit;
  Dec(howmany);

  PrintLine('');
  PrintLine('Дамп даних: загальна довжина: '+IntToStr(len));

  s := '';
  for i := 0 to len-1 do
  begin
    s := s + Format('%02X ', [data^]);
    if ((i+1) mod 16 = 0) then  PrintLine(s);
    Inc(data);
  end;
  PrintLine(s);
  PrintLine('');
end;
{$endif} {DUMP_MODE}


{  TSenderARTCP }

constructor TSenderARTCP.Create;
begin
  inherited;
  FProtocol := IPPROTO_ARTCP
end;

function TSenderARTCP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR ph: TPseudoHeader;
    th: THdrARTCP;
    len: Integer;
    p: PChar;
begin
  // створення ARTCP заголовку
  FillChar(th, sizeof(th), 0);    // очищення
  th.source       := htons(FSrcPort); // порт джерела
  th.dest     := htons(FDstPort); // порт розташування
  th.seq      := htonl(FSequence);// номер послідовності
  th.ack_seq  := htonl(0);        // номер запиту

  SetTHdoff(th, sizeof(th));      // прапорці $0050
  SetTHflag(th, ftFIN, FFinFlag);
  SetTHflag(th, ftSYN, FSynFlag); // прапорці $0250
  SetTHflag(th, ftRST, FRstFlag);
  SetTHflag(th, ftPSH, FPshFlag);
  SetTHflag(th, ftACK, FAckFlag);
  SetTHflag(th, ftURG, FUrgFlag);
  th.window   := htons($FFFF);

  // розрахування ARTCP контрольну суму  ...
  { ARTCP контрольна сума   включає в себе 96 біт псевдо заголовку
    Прописані в ARTCP заголовку.  Цей псевдо заголовок
    Включає Адресу Джерела, Адресу розташування, Протокол (займає октет), та
довжину ARTCP пакету. Використовується для захисту ARTCP від втрати сегментів.
```

```
  }
  ph.saddr     := ih.saddr;
  ph.daddr     := ih.daddr;
  ph.zero      := 0;
  ph.protocol  := FProtocol;
  ph.packetlen := htons(sizeof(th) + dlen);

  p := @FIpBuffer;
  Move(ph, p^, sizeof(ph));
  Inc(p, sizeof(ph));
  Move(th, p^, sizeof(th));
  Inc(p, sizeof(th));
  Move(data^, p^, dlen);

  th.check := CalculateChecksum(@FIpBuffer, sizeof(ph) + sizeof(th) + dlen);

  // заповнюється буфер посилання
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка в байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(th, p^, sizeof(th));
  Inc(p, sizeof(th));
  Move(data^, p^, dlen);
  len := len + sizeof(th)+ dlen;

  Result := SendToRemote(ih.daddr, th.dest, @FIpBuffer, len);
end;

function  TSenderARTCP.GetLength: Word;
begin
  Result := sizeof(THdrARTCP) + FDataLen
end;

procedure TSenderARTCP.UpdateSequence;
begin
  { номер послідовності задається октетом даних в датаграмі }
  Inc(FSequence, FDataLen)
end;


{  TSenderUDP }

constructor TSenderUDP.Create;
begin
  inherited;
  FProtocol := IPPROTO_UDP
end;

function  TSenderUDP.GetLength: Word;
begin
  Result := sizeof(THdrUDP) + FDataLen
end;

function TSenderUDP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR ph: TPseudoHeader;
    uh: THdrUDP;
    len: Integer;
    p: PChar;
begin
  // створюємо заголовок UDP
  FillChar(uh, sizeof(uh), 0);    // очищуємо
  uh.src_port := htons(FSrcPort); // порт джерела
  uh.dst_port := htons(FDstPort); // порт розташування
  uh.length   := htons(sizeof(uh)+dlen);

  // розраховуємо UDP контрольну суму  ...
```

```
  { UDP контрольна сума   включається в  96 бітовий псевдо заголовок який
прописаний в UDP заголовку. Розписано в  RFC 768
    for details
  }
  ph.saddr     := ih.saddr;
  ph.daddr     := ih.daddr;
  ph.zero      := 0;
  ph.protocol  := IPPROTO_UDP;
  ph.packetlen := htons(sizeof(uh)+dlen);

  p := @FIpBuffer;
  Move(ph, p^, sizeof(ph));
  Inc(p, sizeof(ph));
  Move(uh, p^, sizeof(uh));
  Inc(p, sizeof(uh));
  Move(data^, p^, dlen);

  uh.checksum   := CalculateChecksum(@FIpBuffer, sizeof(ph) + sizeof(uh) +
dlen);
  if uh.checksum   = 0 then  uh.checksum   := $FFFF;

  // заповнення буферу відправки
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка у байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(uh, p^, sizeof(uh));
  Inc(p, sizeof(uh));
  Move(data^, p^, dlen);
  len := len + sizeof(uh)+ dlen;

  Result := SendToRemote(ih.daddr, uh.dst_port, @FIpBuffer, len);
end;


{  TSenderICMP  }

type
  THdrECHO = packed record  // ICMP exo (Ping) заголовок
    IcmpType: Byte;
    IcmpCode: Byte;
    контрольну суму  : WORD;
    id      : WORD;
    sequence: WORD;
  end;

constructor TSenderICMP.Create;
begin
  inherited;
  FProtocol := IPPROTO_ICMP
end;

function  TSenderICMP.GetLength: Word;
begin
  Result := sizeof(THdrEcho) + FDataLen
end;

function TSenderICMP.SendDatagram(ih: THdrIP; data: PChar; dlen: Integer):
Integer;
VAR eh: THdrEcho;
    len: Integer;
    p: PChar;
begin
  // створює ICMP/Ping заголовок
  FillChar(eh, sizeof(eh), 0);    // очищення
  eh.IcmpType := 8;               // ICMP тип ECHO
  eh.IcmpCode := 0;
  eh.ID       := htons(FSrcPort);
  eh.Sequence := htons(FDstPort);
```

```
  // розраховуємо ICMP контрольну суму  ...
  p := @FIpBuffer;
  Move(eh, p^, sizeof(eh));
  Inc(p, sizeof(eh));
  Move(data^, p^, dlen);

  eh.checksum   := CalculateChecksum(@FIpBuffer, sizeof(eh) + dlen);

  // заповнення буферу відправки
  p := @FIpBuffer;
  len := GetIHlen(ih); // IP довжина заголовка у байтах
  Move(ih, p^, len);
  Inc(p, len);
  Move(eh, p^, sizeof(eh));
  Inc(p, sizeof(eh));
  Move(data^, p^, dlen);
  len := len + sizeof(eh)+ dlen;

  Result := SendToRemote(ih.daddr, eh.id, @FIpBuffer, len);
end;

procedure TForm1.N5Click(Sender: TObject);
begin
Form2.Show;
end;

end.
```

Файл ARTCPIPHLP.PAS – функції роботи зі стеком протоколів ARTCP/IP

```
unit ArtcpIpHlp; { ARTCP/IP хелпер }

interface

uses Windows, Winsock, SysUtils;

CONST SIO_RCVALL = $98000001;

type
  THdrIP = packed record   // IP заголовок (RFC 791)
    ihl_ver : BYTE;        // комбінована область:
                           //   ihl:4 - IP довжина заголовка divided by 4
                           //   version:4 - IP версія
    tos     : BYTE;        // IP type-of-service поле
    tot_len : WORD;        // загальна довжина
    id      : WORD;        // унікальний ID
    frag_off: WORD;        // Fragment Offset + fragmentation flags (3 bits)
    ttl     : BYTE;        // час передачі
    protocol: BYTE;        // тип протоколу
    check   : WORD;        // IP заголовок контрольна сума
    saddr   : DWORD;       // джерело   IP
    daddr   : DWORD;       // розташування IP
   {початкові опції...}
  end;
  PHdrIP = ^THdrIP;

  (* Розписуємо IP заголовок (RFC 791):

    -ih.ihl довжина заголовка у байтах поділена на 4
     Internet Довжина заголовка –довжина інтернет заголовка
     в 32 бітових словах, мінімальна довжина для правильного заголовка 5.

    -ih.tos - IP type-of-service область забезпечення якості бажаємої послуги.

    -ih.id  - визначається передавачем, для допомоги при трансляції фрагментів
датаграми.

    -ih.frag_off включає 3 бітовий флаг (RFC815).
       Bit 0: зарезервовано, дорівнює нулю
       Bit 1: (DF) 0 = може фрагментуватися,  1 = не може.
       Bit 2: (MF) 0 = останій фрагмент, 1 = поточний фрагмент.
       Bits?: показують розташування цього фрагмента в датаграмі

    -ih.protocol повідомляє IP при посилці датаграми
   *)

  THdrARTCP = packed record     // ARTCP заголовок
    джерело   : WORD;  // порт джерела
    dest    : WORD;  // порт розташування
    seq     : DWORD; // номер послідовності
    ack_seq: DWORD; // наступний номер послідовності
    flags   : WORD;  // комбіноване поле:
                     //   res1:4 – зарезервовано.дорівнює 0
                     //   doff:4 - ARTCP довжина заголовка поділена на 4
                     //   fin:1  - FIN
                     //   syn:1  - SYN
                     //   rst:1  - Reset
                     //   psh:1  - Push
                     //   ack:1  - ACK
                     //   urg:1  - Urgent
                     //   res2:2 – зарезервовано. дорівнює 0
    window : WORD;  // розмір вікна
    check  : WORD;  // контрольна сума
    urg_ptr: WORD;  // використовується для повідомлень
  end;
```

```
  PHdrARTCP = ^THdrARTCP;
  (* Деталі ARTCP заголовку

    -th.seq - номер послідовності – перший октет у сегменті

    -th.doff – зміщення даних - число 32 бітових слов в заголовку ARTCP.


    -th.ack_seq використовується, коли ACK флаг встановлений


    -th.window використовується для управління даними при передачі у любий
проміжок часу

    -th.urgent поле у яке записується конкретний октет, який буде оброблятися.
*)

  THdrUDP = packed record  // UDP заголовок (RFC 768)
    src_port: WORD;        // порт джерела
    dst_port: WORD;        // порт розташування
    length  : WORD;        // довжина, включаючи заголовок
    контрольну суму  : WORD;        // UDP контрольна сума
  end;
  PHdrUDP = ^THdrUDP;

CONST
 { Опції використовують[gs]etsockopt для рівня IPPROTO_IP }

 IP_OPTIONS                 =  1; { встановити/прочитати IP опції }
 IP_HDRINCL                 =  2; { заголовок, включаючий дані }
 IP_TOS                     =  3; { IP тип послуги}
 IP_TTL                     =  4; { IP час передачі  }
 IP_MULTICAST_IF            =  9; { встановити/прочитати IP груповий інтерфейс}
 IP_MULTICAST_TTL           = 10; { встановити/прочитати IP груповий  ttl }
 IP_MULTICAST_LOOP          = 11; { встановити/прочитати IP груповий  loopback }
 IP_ADD_MEMBERSHIP          = 12; { додати IP групу }
 IP_DROP_MEMBERSHIP         = 13; { частина IP групи }
 IP_DONTFRAGMENT            = 14; { не фрагмент IP датаграми }
 IP_ADD_SOURCE_MEMBERSHIP   = 15; { приєднатися до IP грпи/джерела  }
 IP_DROP_SOURCE_MEMBERSHIP = 16; { залишити IP групу/джерело   }
 IP_BLOCK_SOURCE            = 17; { блокувати IP групу/джерело   }
 IP_UNBLOCK_SOURCE          = 18; { розблокувати IP групу/джерело }
 IP_PKTINFO                 = 19; { приймає інформацію пакету для ipv4}

 { мережні інтерфейсні типи}
 IFF_UP            = $00000001; { інтерфейс запуску }
 IFF_BROADCAST     = $00000002; { підтримання широковіщання}
 IFF_LOOPBACK      = $00000004; { інтерфейс loopback }
 IFF_POINTTOPOINT  = $00000008; { інтерфейс включаючий point-to-point link}
 IFF_MULTICAST     = $00000010; { групова  характеристика підтримується }

type
  TArtcpFlagType = (ftFIN, ftSYN, ftRST, ftPSH, ftACK, ftURG);
  TEnumInterfacesEvent = procedure (value: String; iff_type: Integer) of Object;


// присвоюємо імені число
//
function  GetIPProtoName(protocol: Byte): String;
function  GetServiceName(s_port, d_port: Integer): String;
function  GetICMPType(x: Byte): String;

// вказує загальний шлях
//
procedure CleanupWinsock; overload;
procedure CleanupWinsock(VAR socket: TSocket); overload;
function  Win2KDetected: Boolean;
procedure EnumInterfaces(cb: TEnumInterfacesEvent; iff_types: Integer);
function  InitWinsock(hi_ver, lo_ver: Byte): String;
```

```
function  ResolveHostAddress(name: String): u_long;

//
procedure SetTHdoff(VAR th: THdrARTCP; value: Byte);
function  GetTHdoff(th: THdrARTCP): Word;
procedure SetTHflag(VAR th: THdrARTCP; flag: TArtcpFlagType; on: Boolean);
function  GetTHflag(th: THdrARTCP; flag: TArtcpFlagType): Boolean;
procedure SetIHver(VAR ih: THdrIP; value: Byte);
function  GetIHver(ih: THdrIP): Byte;
procedure SetIHlen(VAR ih: THdrIP; value: Byte);
function  GetIHlen(ih: THdrIP): Word;

// контрольна сума, яка використовується в IP заголовку, ARTCP, UDP, й т.д.
//
function  CalculateChecksum(addr: PWord; len: Integer): Word;

// Winsock 2 функції для імпорту
//
function getsockopt(s: TSocket; level, optname: Integer; optval: PChar; var
optlen: Integer): Integer; stdcall;
function setsockopt(s: TSocket; level, optname: Integer; optval: PChar; optlen:
Integer): Integer; stdcall;

function WSAIoctl(s: TSocket;
  dwIoControlCode: DWORD;
  lpvInBuffer: Pointer;
  cbInBuffer: DWORD;
  lpvOutBuffer: Pointer;
  cbOutBuffer: DWORD;
  lpcbBytesReturned: LPDWORD;
  lpOverlapped: Pointer;
  lpCompletionRoutine: Pointer): Integer; stdcall;

implementation

function getsockopt;   far; external 'ws2_32.dll' name 'getsockopt';
function setsockopt;   far; external 'ws2_32.dll' name 'setsockopt';
function WSAIoctl;     far; external 'ws2_32.dll' name 'WSAIoctl';

type
  TIPProto = record
    iType: word;
    iName: String;
  end;

  TWellKnownSvc = record
    port: Integer;
    svc: string[20];
  end;

CONST
  // Тип протоколу  s
  //
  IpProto: Array[1..5] Of TIPProto = (
    (iType: IPPROTO_IP;   iName: 'IP'),
    (iType: IPPROTO_ICMP; iName: 'ICMP'),
    (iType: IPPROTO_IGMP; iName: 'IGMP'),
    (iType: IPPROTO_ARTCP;  iName: 'ARTCP'),
    (iType: IPPROTO_UDP;  iName: 'UDP')
  );

  // сервіси – яким портам. Які сервіси та протоколи відповідають
  //
  WellKnownSvcs: array[1..43] of TWellKnownSvc = (
    ( port:   0; svc: 'LOOPBACK'),
    ( port:   1; svc: 'ARTCPMUX  '),   { ARTCP сервіс  порту }
    ( port:   7; svc: 'ECHO    ' ),  { exo                        }
    ( port:   9; svc: 'DISCARD ' ),  { Заборона                     }
    ( port:  13; svc: 'DAYTIME ' ),  { День та час               }
```

```
    ( port:  17; svc: 'QOTD    ' ),  { }
    ( port:  19; svc: 'CHARGEN ' ),  { генерація символів       }
    ( port:  20; svc: 'FTP_DATA' ),  { протокол Ftp                              }
    ( port:  21; svc: 'FTP_CTL ' ),  { протокол File Transfer Control Protocol}
    ( port:  22; svc: 'SSH     ' ),  { протокол SSH Remote Login Protocol     }
    ( port:  23; svc: 'TELNET  ' ),  { протокол TelNet                         }
    ( port:  25; svc: 'SMTP    ' ),  { протокол Simple Mail Transfer Protocol }
    ( port:  37; svc: 'TIME    ' ),
    ( port:  42; svc: 'NAME    ' ),  { Host Name Server              }
    ( port:  43; svc: 'WHOIS   ' ),  { WHO IS сервіс                }
    ( port:  53; svc: 'DNS     ' ),  { Domain Name Service           }
    ( port:  66; svc: 'SQL*NET ' ),  { Oracle SQL*NET                 }
    ( port:  67; svc: 'BOOTPS  ' ),  { BOOTP Server                   }
    ( port:  68; svc: 'BOOTPC  ' ),  { BOOTP Client                   }
    ( port:  69; svc: 'TFTP    ' ),  { протокол Trivial FTP                    }
    ( port:  70; svc: 'GOPHER  ' ),  { Gopher                         }
    ( port:  79; svc: 'FINGER  ' ),  { Finger                         }
    ( port:  80; svc: 'HTTP    ' ),  { протокол HTTP                           }
    ( port:  88; svc: 'KERBEROS' ),  { протокол Kerberos               }
    ( port:  92; svc: 'NPP     ' ),  { протокол Network Printing Protocol     }
    ( port:  93; svc: 'DCP     ' ),  { протокол Device Control Protocol       }
    ( port: 109; svc: 'POP2    ' ),  { Post Office Protocol Version 2}
    ( port: 110; svc: 'POP3    ' ),  { протокол Post Office Protocol Version 3}
    ( port: 111; svc: 'SUNRPC  ' ),  { протокол SUN Remote Procedure Call     }
    ( port: 119; svc: 'NNTP    ' ),  { протокол Network News Transfer Protocol}
    ( port: 123; svc: 'NTP     ' ),  { протокол Network Time protocol         }
    ( port: 135; svc: 'LOCSVC  ' ),  { Location Service               }
    ( port: 137; svc: 'NTBNAME ' ),  { NETBIOS Name service           }
    ( port: 138; svc: 'NTBDGRAM' ),  { NETBIOS Datagram Service       }
    ( port: 139; svc: 'NTBSESSN' ),  { NETBIOS Session Service        }
    ( port: 161; svc: 'SNMP    ' ),  { протокол Simple Netw. Mgmt Protocol    }
    ( port: 162; svc: 'SNMPTRAP' ),  { протокол SNMP TRAP                     }
    ( port: 220; svc: 'IMAP3   ' ),  { протокол Interactive Mail Access Protocol
v3 }
    ( port: 443; svc: 'HTTPS   ' ),  { протокол HTTPS                         }
    ( port: 445; svc: 'MS-DS   ' ),  { Microsoft-DS                   }
    ( port:1433; svc: 'MSSQL   ' ),  { MSSQL                          }
    ( port:3306; svc: 'MYSQL   ' ),  { MySQL                          }
    ( port:5900; svc: 'VNC     ' )   { VNC - similar to PC Anywhere   }
  );

function GetIPProtoName(protocol: Byte): String;
VAR i: Integer;
begin
  Result := IntToStr(protocol);
  for i := 1 To sizeof(IPPROTO) div sizeof(TIPProto) do
    if protocol = IPPROTO[i].itype then
      Result := IPPROTO[i].iName;
end;

function GetServiceName(s_port, d_port: Integer): String;
VAR i: Integer;
begin
  Result := '';
  for i := 1 to sizeof(WellKnownSvcs) div sizeof(TWellKnownSvc) do
    if (s_port = WellKnownSvcs[i].port) OR (d_port = WellKnownSvcs[i].port) then
    begin
      Result := WellKnownSvcs[i].svc;
      break;
    end;
end;

function  GetICMPType(x: Byte): String;
begin
  Result := 'UNKNOWN';
  case x of
     0: Result := 'ECHO_R'; // повтор exa
     3: Result := 'DSTUNR'; // Розташування досягаємого
     4: Result := 'SRC_Q';  // блокування джерела повідомлень
```

```
     5: Result := 'REDIR';  // перепосилання
     8: Result := 'ECHO';   // exo
    11: Result := 'TTLX';   // перевищення часу
    12: Result := 'BADPAR'; // параметр помилки
    13: Result := 'TIME';   // часова мітка
    14: Result := 'TIME_R'; // повтор часової мітки
    15: Result := 'INFO';   // запит інформації
    16: Result := 'INFO_R'; // повтор інформації
  end
end;

function Win2KDetected: Boolean;
VAR IsNT: Boolean;
    ver: DWORD;
begin
  ver := GetVersion();

  IsNT  := ver < $80000000;
  Result := IsNT AND ((ver AND $FF) >= 5)
end;

function InitWinsock(hi_ver, lo_ver: Byte): String;
VAR versionRequested: Word;
    errorStatus: Integer;
    wd: TWSADATA;
begin
  Result := '';

  versionRequested := MAKEWORD(lo_ver, hi_ver);

  errorStatus := WSAStartup(versionRequested, wd);
  if (errorStatus <> 0) then
  begin
    Result := ' Помилка ініціалізації Winsock '+ IntToStr(errorStatus);
    Exit;
  end;

  if (LOBYTE(wd.wVersion) <> LOBYTE(versionRequested)) OR
     (HIBYTE(wd.wVersion) <> HIBYTE(versionRequested)) then
  begin
    Result := Format('Неспівпадіння версії Winsock (required: %d.%d, found:
%d.%d)',
                     [HIBYTE(versionRequested),
                     LOBYTE(versionRequested),
                     HIBYTE(wd.wVersion),
                     LOBYTE(wd.wVersion)]);
    Exit;
  end;
end;

procedure CleanupWinsock(VAR socket: TSocket);
begin
  if (socket <> 0) AND (socket <> INVALID_SOCKET) then
    CloseSocket(socket);
  socket := INVALID_SOCKET;
  WSACleanup()
end;

procedure CleanupWinsock;
begin
  WSACleanup()
end;

procedure EnumInterfaces(cb: TEnumInterfacesEvent; iff_types: Integer);
type
  TSockAddrGen = packed record
    AddressIn: sockaddr_in;
    { This record must be big enough
      to hold struct sockaddr_in6,
```

```
      which is 24 bytes long... }
    dummy: array[0..7] of char;
  end;

  TInterface_Info = packed record { see Q181520 }
    iiFlags: u_long;         // інтерфейс флагів
    iiAddress,               // інтерфейс адрес
    iiBroadcastAddress,      // передача адреси
    iiNetmask: TSockAddrGen; // маска мережі
  end;

CONST SIO_GET_INTERFACE_LIST = $4004747F;
VAR
  InterfaceList: array[0..$20] of TInterface_Info;
  NumInterfaces: Integer;
  BytesReturned: DWORD;
  AddrIn: TSockAddrIn;
  pAddr: PChar;
  flags: u_long;
  s: TSocket;
  i: Integer;
begin
  if InitWinsock(2,2) <> '' then Exit;

  s := Socket(AF_INET, SOCK_STREAM, IPPROTO_IP); { artcp сокет }
  if (s <> INVALID_SOCKET) then
  begin
    try
      if WSAIoctl(s,
                  SIO_GET_INTERFACE_LIST,
                  Nil,
                  0,
                  @InterfaceList,
                  sizeof(InterfaceList),
                  @BytesReturned,
                  Nil,
                  Nil) <> SOCKET_ERROR then
      begin
        NumInterfaces := BytesReturned div SizeOf(TInterface_Info);

        for i := 0 to NumInterfaces - 1 do
        begin
          AddrIn := InterfaceList[i].iiAddress.AddressIn;
          pAddr  := inet_ntoa(AddrIn.sin_addr);

          flags := InterfaceList[i].iiFlags;

          if (flags AND iff_types) = flags then
            if Assigned(cb) then  cb(pAddr, flags)
        end;
      end;
    except
      { do nothing }
    end;
  end;

  CleanupWinsock(s);
end;

function ResolveHostAddress(name: String): u_long;
//
// Ця функція записує заголовок імені хосту або IP адреси
//
VAR hep: PHostEnt;
    addr: u_long;
begin
  addr := inet_addr(PChar(name));

  if (addr = u_long(-1)) then
```

```
  begin
    hep := gethostbyname(PChar(name));
    if (hep <> Nil) then
    begin
      with TInAddr(addr), hep^ do
      begin
        S_un_b.s_b1 := h_addr^[0];
        S_un_b.s_b2 := h_addr^[1];
        S_un_b.s_b3 := h_addr^[2];
        S_un_b.s_b4 := h_addr^[3];
      end
    end
  end;

  Result := addr;
end;


function GetIHlen(ih: THdrIP): Word;  // IP довжина заголовка
begin

  // повертає довжину у байтах
  Result := (ih.ihl_ver AND $0F) SHL 2
end;

procedure SetIHlen(VAR ih: THdrIP; value: Byte);
begin
  // ділить значення на 4
  value := value SHR 2;
  ih.ihl_ver := value OR (ih.ihl_ver AND $F0)
end;

function GetIHver(ih: THdrIP): Byte;  // IP версія
begin
  // записує вищу частину
  Result := ih.ihl_ver SHR 4
end;

procedure SetIHver(VAR ih: THdrIP; value: Byte);
begin
  // встановлює вищу частину
  ih.ihl_ver := (value SHL 4) OR (ih.ihl_ver AND $0F)
end;

(* ARTCP заголовок містить наступні флаги
       res1:4  - зарезервовано.дорівнює 0
       doff:4 - ARTCP довжина заголовка ділиться на 4
       fin:1   - FIN
       syn:1   - SYN
       rst:1   - Reset
       psh:1   - Push
       ack:1   - ACK
       urg:1   - Urgent
       res2:2 - зарезервовано,дорівнює 0
     MSB
*)

CONST flagMask: Array[ftFIN..ftURG] of Integer = ($100, $200, $400, $800, $1000,
$2000);

function GetTHflag(th: THdrARTCP; flag: TArtcpFlagType): Boolean;
begin
  Result := Boolean(th.flags AND flagMask[flag])
end;

procedure SetTHflag(VAR th: THdrARTCP; flag: TArtcpFlagType; on: Boolean);
begin
  if on then
    th.flags := th.flags OR flagMask[flag]
```

```
  else
    th.flags := th.flags AND NOT flagMask[flag]
end;

function GetTHdoff(th: THdrARTCP): Word;
begin

  Result := (($00F0 AND th.flags) SHR 4) SHL 2;
end;

procedure SetTHdoff(VAR th: THdrARTCP; value: Byte);
VAR x: Integer;
begin
  x := value SHR 2; //
  th.flags := (x SHL 4) OR (th.flags AND $FF0F)
end;

function CalculateChecksum(addr: PWord; len: Integer): Word;
// алгоритм отримання  контрольну суму    RFCs 791,793
//
VAR sum: DWORD;
begin
  Result := 0;
  sum := 0;

  while (len > 1) do
  begin
    Inc(sum, addr^);
    Inc(addr);
    Dec(len, 2);
  end;


  if (len = 1) then
  begin
    PByte(@Result)^ := PByte(addr)^;
    Inc(sum, Result);
  end;

  sum := (sum SHR 16) + (sum AND $FFFF);{ }
  Inc(sum, sum SHR 16);                 { }
  sum := NOT sum;                       { }
  Result := Word(sum);                  { }
end;

end.
```

Файл About.PAS – довідка про програму

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
Dialogs, StdCtrls, ExtCtrls;

type
  TForm2 = class(TForm)
    Image1: TImage;
    Memo1: TMemo;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var  Form2: TForm2;

implementation
     uses Mainunit;
{$R *.dfm}

procedure TForm2.Button1Click(Sender: TObject);
begin
Form2.Close;
end;

procedure TForm2.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Дослідження та програмна реалізація системи адаптивної
оптимізації маршрутизації мережі інформаційних та комп'ютерних систем ');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Водзинський Євген Юрійович ');
Memo1.Lines.Add('                    гр. КН-21М-1,4');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Смірнова Т.В.');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2022');
end;
end.
```