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
“____” _____ 2026 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC”

Виконав здобувач вищої освіти
IV курсу, групи КН-22
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Дорошок О.О.
«____» _____ 2026 р.

Керівник проекту
кандидат технічних наук
_____ Лисенко І.А.
«____» _____ 2026 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет
Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 15 » січня 2026 року

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

Дорошок Олені Олегівні
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC*

2. Керівник роботи *Лисенко Ірина Анатоліївна, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 116-02 від 06.02.2026 року

3. Строк подання студентом роботи до захисту *23.05.2026 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи реалізації інтелектуального контролера доставки додатків ADC*

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

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

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

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

7. Дата видачі завдання « 15 » січня 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2026 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2026 р.	
3.	Розробка моделі компонента	20.03.2026 р.	
4.	Розробка структур даних	25.03.2026 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2026 р.	
6.	Програмування алгоритмів	10.04.2026 р.	
7.	Оформлення ПЗ	17.04.2026 р.	
8.	Попередній захист роботи	23.05.2026 р.	

Дата видачі завдання
« 15 » січня 2026 р.

Підпис керівника
_____ Лисенко І.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 15 » січня 2026 р.

Підпис здобувача
_____ Дорошок О.О.
(прізвище та ініціали)

АНОТАЦІЯ

Дорошок О.О. Програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2026.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи реалізації інтелектуального контролера доставки додатків ADC.

Метою розробки є програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC.

Результат роботи – програмна реалізація системи реалізації інтелектуального контролера доставки додатків ADC.

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

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

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

Програму розроблено в середовищі Python.

Ключові слова: комп'ютерні науки, інтелектуальний контролер, доставка додатків, ADC

ABSTRACT

Doroshok O.O. Software for the implementation of the intelligent application delivery controller ADC. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2026.

In this final qualification work for the first (bachelor's) level of higher education, software has been developed that is intended for the implementation of the intelligent application delivery controller ADC.

The purpose of the development is the software for the implementation of the intelligent application delivery controller ADC.

The result of the work is the software implementation of the intelligent application delivery controller ADC.

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 the software are provided.

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

The program was developed in the Python environment.

Keywords: computer science, intelligent controller, application delivery, ADC

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

EOM	–	електронно-обчислювальна машина
ATM	–	Asynchronous Transfer Mode – асинхронний спосіб передачі даних
BER	–	Bit Error Rate – імовірність ушкодження одного біта
CBWFQ	–	class based weighted fair queueing
DiffServ	–	Differentiated Services – механізм який залежно від вимог до якості обслуговування записує в IP заголовки пакетів спеціальні мітки
DSCP	–	DiffServ CoSde Point
ICMP	–	Internet Control Message Protocol – міжмережний протокол управляючих повідомлень
ISP	–	Internet Service Provider – провайдер
LLQ	–	low latency queueing
MPLS-TE	–	Multiprotocol Label Switching Traffic Engineering
PQ	–	priority queueing
RIO	–	RED with Input Output
RTT	–	Round Trip Time – час між відправкою запиту та отриманням відповіді
STM	–	Synchronous Transfer Mode – синхронний спосіб передачі даних
QoS	–	Quality of Service – якість обслуговування
WFQ	–	Weighted Fair Queuing – механізм планування пакетних потоків даних
WRED	–	Weighted Random early detection – взвішане значення довжини черги, у якості фактора, визначаючого ймовірність відкидання пакета

ВСТУП

Актуальність теми. На тлі росту інтересу корпоративних замовників до хмарних технологій, а також поширення DevOps-процесів контролер доставки застосунків ADC стає усе більше затребуваним інструментом для розроблювачів застосунків.

Контролер доставки застосунків (Application Delivery Controller, ADC) уже давно став для IT-фахівців звичним терміном, під яким розуміють пристрою, що забезпечують різні мережні функції, такі як рівномірний розподіл навантаження, оптимізація роботи застосунків і завантаження мережного каналу, а також різні функції з безпеки, включаючи обробку шифруємих з'єднань браузера.

Контролер доставки застосунків може похвастатися завидною «кар'єрою»: будучи спочатку звичайним балансувальником навантаження, воно трансформувався в мультифункціональну платформу, чий самий продуктивний представник розвиває швидкість передачі даних у додатку до 200 Гбіт/с і здатний обробляти до 5 млн HTTP-запитів одночасно.

Для віртуалізованих і приватних хмарних середовищ уже давно існують віртуальні ADC, які по функціональності мало чим відрізняються від своїх фізичних побратимів. Декілька віртуальних ADC можуть працювати паралельно на базі центральної потужної ADC-платформи, такий як Citrix NetScaler SDX. При наявності широкого спектра віртуальних і апаратних пристроїв це забезпечує гнучкі можливості для масштабування.

Однак наступна еволюція оптимізатора застосунків не менш захоплююча. Поширення практик DevOps для забезпечення тісної інтеграції команд розробки, тестування й експлуатації застосунків, а також для контролю якості й IT-операцій веде до того, що ADC трансформуються із традиційного мережного пристрою в елемент виробничого середовища розроблювача програмного забезпечення.

Команди розроблювачів можуть також просто вмонтувати ADC-пристрою у свої архітектури застосунків, як і будь-який інший віртуальний сервер. До того ж завдяки NetScaler CPX тепер доступний повний діапазон ADC-функцій у формі-форматі Linux-контейнера.

Контейнерна технологія, подібна Docker, сьогодні особливо популярна серед програмістів при розробці застосунків, орієнтованих на хмарні середовища. Типові хмарні мікросервіси – або навіть цілі хмарні архітектури застосунків – можна легко впровадити в публічні хмарні середовища, протестувати, а при необхідності знову деініціалізувати або передати в працюючий процес завдяки Docker і аналогічним рішенням.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC.

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

– Огляд існуючих систем реалізації інтелектуального контролера доставки додатків ADC.

– Дослідження системи реалізації інтелектуального контролера доставки додатків ADC.

– Програмна реалізація системи реалізації інтелектуального контролера доставки додатків ADC.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі реалізації інтелектуального контролера доставки додатків ADC.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Використовуючи ADC у форматі контейнера, програмісти можуть заздалегідь протестувати хмарні додатки в рамках циклу розробки відповідно до реального робочого сценарію, що включає в себе такі типові ADC-функції оптимізації продуктивності й доступності сервера застосунків, як балансування навантаження, перемикання контенту (Content Switching), кешування контенту, специфічне для передплатника сервісів керування трафіком і оптимізація транспортного протоколу TCP. До перерахованого також додаються такі функції безпеки для ADC-контейнера, як термінування сеансів SSL, відбиття DoS- і DDoS-атак, міжмережевий екран для захисту Web-застосунків, а також організація багатофакторної автентифікації.

При виборі підходящого ADC для DevOps важлива не тільки підтримка рішень для керування контейнерами й оркестрації, таких як Docker Swarm, Google Kubernetes і Apache Mesos. ADC на базі контейнера повинен насамперед підтримувати виявлення необхідних сервісів і автоматичну реконфігурацію, щоб швидко реагувати на зміни в середовищі застосунків. Крім того, він повинен мати інтегровану функціональність керування й аналітики, щоб команди розроблювачів DevOps у динамічному середовищі застосунку завжди могли одержати інформацію про копії, що з'явилися в інфраструктурі, ADC і їхньої продуктивності.

1.2 Область застосування

В 2009 році виник DevOps. На перший погляд це розроблювачі з навичками сісадмінів і сісадміни з навичками розроблювачів. Але насправді це аж ніяк не так. Дане підхід має чіткі цілі, філософію, інструменти й методи, які тільки деякі російськомовні компанії починають використовувати.

Терміном «DevOps» звичайно називають виниклий професійний рух, що виступає за спільні робочі відносини між розроблювачами й IT-підрозділом, у результаті одержуючи більше швидке виконання планованих робіт (наприклад, високі темпи розгортання), одночасно збільшуючи надійність, стабільність, стійкість і безпеку production-середовища. Чому розроблювачі й IT-підрозділу (далі для стислості адміни)? Тому що, як правило, потік цінностей (value stream) перебуває між бізнесом (де визначаються вимоги) і замовником (куди поставляється результат).

Вважається, що рух DevOps зародилося в 2009 році, як об'єднання численних суміжних і взаємодоповнюючих співтовариств: Velocity Conference, на якій особливо яскравими був виступ «10 Deploys A Day» John Allspaw і Paul Hammond, підхід «інфраструктура як код»(Mark Burgess і Luke Kanies)," Agile infrastructure "(Andrew Shafer) і системне адміністрування в Agile (Patrick DeBois), підхід Lean Startup Ерика Райса з його безперервної інтеграції, а так само широка доступність хмарних і PaaS технологій.

Одним із принципів Agile є надання робочого програмного забезпечення меншими й більше частими релізами, на відміну від підходу «великого вибуху», що властивий водоспадної методології. Так що Agile прагнуть мати потенційно готовий продукт наприкінці кожного спринту (як правило, раз у два тижні).

Високі темпи розгортання приводять до того, що перед адмінами накопичується більша гора завдань. Clyde Logue, засновник StreamStep, говорить про це так: «Agile зіграв важливу роль у розробці для відновленню довіри в

бізнесу, але він ненавмисно залишив IT Operations за. DevOps це спосіб відновлення довіри до всієї IT-організації в цілому ».

DevOps особливо добре доповнює Agile, тому що він розширює й доповнює процеси безперервної інтеграції й випуску продукту, даючи впевненість у тому, що код готовий до випуску й несе цінність для клієнта.

DevOps дозволяє сформувати набагато більше безперервний потік роботи в IT-підрозділ. Якщо код не поставляється в продукт, у тому виді як він був розроблений (наприклад, розроблювачі поставляють код кожні два тижні, але розгортається він тільки один раз у два місяці), операції по установці будуть накопичуватися перед адмінами, клієнти не одержать важливого функціонала для себе й сам процес установки в такому випадку приведе до хаосу й дезорганізації.

DevOps, у тому числі, змінює культуру, так само як змінює процеси й метрику розроблювачів і адмінів.

Хоча багато з людей уважають, що DevOps це реакція на ITIL (IT Infrastructure Library) і ITSM (IT Service Management), у мене інша точка зору. ITIL і ITSM як і раніше є кращою кодифікацією бізнес-процесів, що лежать в основі IT підрозділу, тому що насправді описують багато речей, необхідні для того, щоб IT команда могла працювати у форматі DevOps.

Безперервна інтеграція й релізи є результатом роботи розроблювачів, що у свою чергу є матеріалом для роботи адмінів. Для того щоб укладеться в більше швидкий ритм релізів, що існує в DevOps, багато процесів ITIL вимагають автоматизації, зокрема, пов'язані зі зміною конфігурації й релізів у цілому.

Ціль DevOps не стільки збільшення швидкості видачі нового функціонала, скільки розгортання цього функціонала у виробництві, без хаосу й порушення роботи вже запущеного додаток, а так само швидке виявлення й виправлення проблем, якщо вони все-таки відбуваються. Це додає в ITIL такі пункти як настроювання сервісів, керування інцидентами й проблемами.

Visible Ops є посібником з досягнення трансформації до високопродуктивних IT підрозділами по шляху „от гарного до великого“, і одним

із ключових понять є поняття по обмеженню й скороченню незапланованих робіт. DevOps у свою чергу орієнтується, у першу чергу на те, як створити швидкий і стабільний потік планових робіт з розробки й адміністрування. Проте, DevOps також має більш цілісний підхід до систематичного викорінювання незапланованої роботи, звертаючись до принципів відповідального керування й скорочення технічного боргу.

Всі DevOps паттерни можуть бути отримані за допомогою підходу «Три шляхи». Вони описують цінності й філософію, які є основою процесів, процедур, практик, а також обов'язкових кроків.

Перший Шлях підкреслює продуктивність всієї системи в цілому, на відміну від продуктивності окремої ланки або відділу – це може бути як великий підрозділ (наприклад, розробка або IT відділ) так і окремі люди (наприклад, розроблювач, системний адміністратор).

У центрі уваги перебувають всі бізнес-потоки по створенню цінності, які включені в IT. Інакше кажучи, він починається тоді, коли визначаються основні вимоги (наприклад, для бізнес або IT), вони закінчені в розробці, а потім перейшли в IT-відділ, у яких цінність сервісу потім і доставляється замовникові у вигляді сервісу.

Результати проходження Першому Шляху на практиці полягають у тому, що відомі баги ніколи не передаються на наступний етап робіт, ніколи не розвивається локальна оптимізація, що приводить до створення глобальної деградації, відбувається безперервне поліпшення й прагнення досягти глибокого розуміння системи.

Другий Шлях полягає в створенні петлі зворотного зв'язка який йде справо наліво. Метою практично будь-якої ініціативи по вдосконалювання процесу є скорочення й посилення зворотний зв'язок, щоб необхідні виправлення могли впроваджуватися постійно. Підсумки Другого Шляху: розуміння й реакція на всіх клієнтів, як внутрішніх так і зовнішніх, скорочення й посилення всіх

петель зворотного зв'язка, і поглиблення знань про середовище там, де це потрібно.

Третій шлях полягає в створенні культури, що впливає на дві речі: постійне експериментування, що вимагає прийняття ризиків і добування уроків з успіхів і невдач, а також розуміння того, що повторення й практики є передумовою до майстерності.

Нам потрібні обоє цих принципи рівною мірою. Експерименти й прийняття ризиків, є тим, що гарантує, що ми продовжимо поліпшення, навіть якщо це означає, що ми можемо зайти в занадто небезпечні нетрі. І ми повинні вчитися навичками, які можуть допомогти нам вийти з небезпечної зони, коли ми зайшли занадто далеко.

Підсумки Третього Шляху включають виділення часу для поліпшення повсякденної роботи, створення ритуалів, які заохочують команду в прийнятті ризиків і можливому створенню несправностей у системі з метою підвищення стійкості в перспективі.

Виділено 4 моделей впровадження DevOps:

Модель 1: Поглиблення процесів розробки в поставку: це включає розширення безперервної інтеграції й випуску на бойові сервера, інтеграція тестування й інформзахисту в робочі процеси, що дає готовий до поставки код, настроєні середовища, і так далі.

Модель 2: Створення зворотного зв'язка від продукту до розробки: включає створення повної хронології подій у розробці й адмініструванні, з метою допомоги в дозволі проблем, а так само надання доступу команді розробки до аналізу проблем на продукті, одночасно зі створенням розроблювачами сервісів самообслуговування, скрізь де це можливо, і створення інформаційних радіаторів, що показують зміну в поводженні системи при внесенні змін.

Модель 3: Об'єднання розробки й адміністрування: складається у включенні команди розробки в ланцюжок дозволу проблем, призначення

розроблювачів на дозвіл проблем на продукті, а так само взаємні тренінги між розроблювачами й адміністраторами, щоб зменшити кількість ескалацій.

Модель 4: Включення IT команди в розробку: складається у включенні або тісному зв'язку між IT і розробкою, створення багатоетапних користувальницьких історій (включаючи розгортання, керування кодом у виробництві й т.д.), і визначення нефункціональні вимоги, які можуть бути використані у всіх проектах.

Я думаю, що є три бізнес переваги, які організації одержують від переходу на DevOps: швидкий вихід на ринок (наприклад, скорочення часу циклу й більше високі темпи розгортання), підвищення якості (наприклад, підвищення доступності, менше збоїв і т.д.), і збільшення організаційної ефективності (наприклад, більше часу витрачається на діяльність пов'язану зі збільшенням цінності продукту в порівнянні із втратами, збільшення кількості функціонала, переданого замовникові).

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Radware VADI: віртуалізація доставки застосунків

Віртуалізація є потужним і головним на сьогодні IT-нововведенням, елементи якого ми знаходимо в багатьох мережах – від невеликих і великих підприємств до дуже більших хостингових компаній і ЦОД операторів зв'язку й постачальників різних послуг.

Тенденції в області віртуальних ЦОД

Основні завдання віртуалізації, поза залежністю від розміру підприємства – створення консолідованих, економічних, надійних, доступних і продуктивних центрів обробки даних.

Для того, щоб повністю скористатися перевагами віртуалізації, IT-менеджери прагнуть перешикувати всі рівні ЦОД.

Перший етап стосується серверної архітектури й сховищ даних, для яких застосовуються технології VMware vSphere™ або Microsoft Hyper-V™. Швидкість процесу віртуалізації залежить від загальної кількості й кількості критично важливих застосунків у ЦОД, оскільки IT-менеджери природно прагнуть віртуалізувати менш важливі додатки, і тільки потім переходити до критично важливого. Дослідження, проведені компаніями CDW1 і EMC2, показують, що більшість організацій створює гетерогенне середовище, тобто користується сполученням спеціалізованих і віртуальних серверів.

Другий етап перебудови, що набирає силу в багатьох ЦОД – це консолідація й віртуалізація мережної інфраструктури (рівень доступу, агрегації й так далі) і перехід від вертикальної архітектури мережі до більше плоского, з

меншої ієрархічністю. Ця перебудова в основному підтримується великими виробниками мережного встаткування, такими, як Juniper і Cisco.

Розглянемо, як віртуалізація й консолідація мережної інфраструктури впливає на процеси на рівні доставки застосунків і, зокрема, на роль контролерів доставки застосунків ADC (Application Delivery Controllers).

Віртуальна інфраструктура доставки застосунків Radware VADI

Перебудова мережної інфраструктури з вертикальної ієрархії на горизонтальну змінює роль і місце блоків ADC. Пристрою ADC, які раніше були тісно зв'язані кожний з певним додатком, тепер повинні обслуговувати цілий шар віртуалізованих застосунків на загальній серверній інфраструктурі.

Для того, щоб сервіси доставки застосунків відповідали новій архітектурі віртуальних ЦОД, необхідно впровадити гнучкий віртуалізований рівень доставки застосунків так, щоб забезпечити на ньому виконання сервісів доставки застосунків відповідно до угоди про рівень обслуговування (SLA) і з передбачуваною продуктивністю.

Справжня віртуалізація доставки застосунків вимагає відділення сервісів ADC від фізичних обчислювальних ресурсів у такий же спосіб, як застосунку й керуюча система їхньої експлуатації відділяються від обчислювальних ресурсів при віртуалізації серверів.

Radware уперше в галузі пропонує рішення віртуальної інфраструктури доставки застосунків VADI (Virtual Application Delivery Infrastructure), що дозволяє перетворити обчислювальні ресурси, засоби виконання застосунків і послуги віртуалізації в єдину надійно працюючу й нарощувану віртуальну інфраструктуру.

Radware VADI дозволяє використовувати об'єднані апаратні ресурси ЦОД для виконання різних застосунків у віртуальному середовищі з гарантією SLA і з надійністю й передбачуваністю, властивим фізичному середовищу. Таким чином, знімаються багато ризиків, що виникають при консолідації ADC, і досягається характерна для віртуального середовища можливість динамічного перерозподілу

обчислювальних потужностей. Стандартна інфраструктура доставки застосунків перетвориться у віртуальну площину керування доставкою застосунків, що забезпечує сервіси всьому ЦОД.

Рішення Radware VADI підходить для віртуального ЦОД будь-якого розміру, будь це повністю віртуалізований або ЦОД змішаного типу.

Radware VADI дозволяє здійснити консолідацію й віртуалізацію сервісів доставки застосунків як інтегральна частина архітектури віртуального ЦОД і його систем автоматизації керування й надання послуг. Рішення інтегрується у віртуальну інфраструктуру Vmware, включає програмний модуль vDirect для автоматизації керування доставкою трафіку застосунків і інші елементи, які інтегруються з VMware vCenter і vCenter Orchestrator у віртуальних ЦОД.

Блоки віртуального ADC

В основі Radware VADI перебуває віртуальний блок ADC (vADC), що забезпечує всього функціонала стандартного пристрою ADC, у такий спосіб трансформуючи контролер доставки застосунків у сервіси контролера доставки застосунків (ADC). Функції ADC можуть бути реалізовані в 3 варіантах на вибір, залежно від потреб застосунків. Radware поставляє:

– Спеціалізований ADC на фізичній платформі з одним блоком vADC, що надає нарощувану «на вимогу» пропускну здатність і сервіси;

– Численні vADC на спеціалізованій платформі Radware OnDemand Switch® під керуванням першого в галузі гіпервізору доставки застосунків ADC-VX™;

– Програмний ADC Alteon® VA™: vADC на основі будь-якої віртуальної серверної інфраструктури, що працює, як віртуальний пристрій.

Кожний блок vADC, незалежно від того, на яких обчислювальних ресурсах він реалізований, надає всі можливості доставки застосунків, включаючи такі розвинені послуги, як балансування навантаження серверів по локальній і глобальній мережі, послуги Layer 7, акселерацію трафіку застосунків, убудовану захист, BWM і багато чого іншого. Так досягається основна мета

Radware VADI – відділення сервісів ADC від фізичних обчислювальних ресурсів. Служби IT підприємства або ЦОД можуть застосовувати будь-яку комбінацію зазначених типів ADC залежно від конкретних умов і цілей: вимог SLA, що існують потужностей, обмежень у приміщеннях ЦОД і обраної моделі реалізації застосунку (хостинг, використання «хмарних» послуг і так далі).

У чому переваги рішення VADI

Radware пропонує унікальну інтеграцію сервісів віртуалізації серверної інфраструктури, нових сервісів VADI і 3 різних типів ADC, що дає більшу гнучкість при автоматизації робочих процесів у віртуальних ЦОД. Концепція Radware VADI підтримує повне забезпечення віртуальних сервісів (сервера, сховища, послуги ADC, ресурси й так далі), адаптацію ресурсів для виконання SLA для кожного застосунку й міграцію сервісів усередині ЦОД або між декількома ЦОД.

Завдяки Radware VADI постачальникам «хмарних» послуг і хостингу, віртуальним ЦОД великих підприємств, операторам зв'язку, а також невеликим компаніям, що користуються віртуальними серверами, стають доступні наступні переваги віртуалізації:

– Істотна економія витрат завдяки консолідації ADC.

– Спрощення переходу до віртуальних ЦОД.

– Більша динамічність у віртуальному ЦОД.

– Підвищення ефективності завдяки автоматизації процесів у ЦОД.

– Повна гнучкість ресурсів доставки застосунків відповідно до бізнес-вимогами застосунків.

– Масштабування пропускної здатності, кількості блоків vADC і застосування розвинених сервісів «на вимогу».

– Прискорене повернення інвестицій, кардинальне зниження капітальних витрат і їхній повний захист.

Це перше на ринку рішення, що дозволяє повністю автоматизувати всі дії у віртуальному ЦОД, включаючи доставку застосунків, і реально домогтися динамічності бізнесу за допомогою «хмарних» послуг.

Citrix ADC

Організації здійснюють цифрову трансформацію, а це означає, що розробка застосунків повинна мінятися відповідно до вимог часу. За допомогою Citrix ADC розроблювачі застосунків у вашім підрозділі можуть здійснювати розгортання власних ресурсів для доставки застосунків будь-яким обраним способом – у своєму середовищі розробки, у хмарі або на базі «платформи як послуги» (Platform as a Service, PaaS). За допомогою в IT-відділ звертатися не потрібно, тому що Citrix ADC забезпечує швидкість і контроль, необхідні вам і вашим розроблювачам.

Довідник StyleBook по Citrix Application Delivery Management визначає всі конфігурації Citrix ADC у структурованій формі YAML, доступ до яких можна одержати через простий графічний інтерфейс користувача або через API StyleBook. Ваші IT-відділи можуть вибирати необхідні їм StyleBook і співвідносити їх з конфігурацією для забезпечення розроблювачів вашої сфери діяльності. Здійснюйте масштабування своєї діяльності, надавши своїм розроблювачам модель самообслуговування для віртуального провіжинінгу IP при розгортанні застосунку в середовищах розробки, тестування й виробництва.

Citrix ADC підтримує доставку застосунків ПЗ й апаратного забезпечення, надаючи можливість вибрати форм-фактор і підхід відповідно до ваших потреб. Citrix Networking VPX підтримує горизонтальне й вертикальне масштабування, відновлення без переривання робочого процесу й збір докладних телеметричних даних, що допомагають розроблювачам і членам операційної групи швидко усувати проблеми з застосунками. Citrix Networking CPX надає такі ж можливості у форм-факторі контейнера Docker.

Розроблювачі одержують повний доступ до ADC, а ваші IT-відділи зберігають можливість контролю

За допомогою Citrix ADC розроблювачі можуть використовувати можливості ADC під час розробки, застосовуючи політики й конфігурації, які можна транспортувати або переміщати в робочі пристрої Citrix Networking

Appliances під контролем членів вашої групи по хмарних технологіях. Завдяки повному доступу до власних ресурсів Citrix ADC розроблювачі можуть переглядати статистичні дані про роботу додатки й усувати несправності навіть тоді, коли інфраструктуру одночасно використовують кілька застосунків.

Citrix Networking Appliances використовується підприємствами й постачальниками підтримки безпеки вже більше десяти років. У багатьох продуктів з відкритим кодом немає платформи керування, аналітики й взаємодії. А в більшості основних постачальників ADC немає контейнерного ADC зі спрощеною архітектурою, що здатний функціонувати незалежно від апаратного забезпечення. На відміну від конкурентів, Citrix ADC пропонує повний комплект.

Продукти Citrix

Citrix ADC

– Citrix ADC забезпечує виняткову комфортність роботи з застосунками в гібридному багатохмарнойу середовищу з постійною оптимізацією й доступністю ваших застосунків у будь-який час.

– Шаблони Citrix ADC Application використовують засоби автоматизації хмарних сервісів для швидкого розгортання інфраструктури доставки застосунків.

– Вони пропонуються в різних форм-факторах, включаючи стандартні галузеві гіпервізори й хмарний, зручний для використання з DevOps контейнерний форм-фактор, провіжинінг якого ваші розроблювачі можуть легко здійснити на хосте Docker.

Citrix Application Delivery Management

– Ви можете управляти всією інфраструктурою Citrix ADC, здійснювати моніторинг і усувати проблеми з єдиної уніфікованої консолі.

– Ви можете переглядати, автоматизувати мережні сервіси для масштабування архітектур застосунків і управляти цими сервісами.

– Одержите наскрізну аналітику й відомості про продуктивність застосунків і автоматизоване усунення несправностей.

Citrix ADC CPX Express

– Citrix ADC CPX Express забезпечує балансування навантаження Інтернету й застосунків Citrix ADC, прискорення, безпеку й розвантаження в простому, зручному для установки контейнері.

– Використовуйте можливості механізму Docker, а також функції балансування навантаження й керування трафіком Citrix ADC для контейнерних застосунків.

Контролер доставки застосунків Fortinet FortiADC

Пристрій для балансування навантаження й прискорення доставки застосунків. FortiADC – це сімейство фізичних і віртуальних контролерів (Application Delivery Controller, ADC), які забезпечують високу продуктивність, безпеку й масштабованість доставки застосунків у корпоративних дата центрах. Використання пристроїв дозволяє збільшити ефективність серверів і знизити сукупну вартість володіння (TCO).

ADC нового покоління

FortiADC є контролером доставки застосунків нового покоління, що сполучить у собі такі функції, як балансування навантаження (load balancer), SSL розвантаження й прискорення, оптимізація роботи застосунків, керування трафіком, Web Application Firewall і ряд інших.

FortiADC виконує розвантаження SSL з підтримкою 4 096-бітних ключів, керування TCP з'єднаннями, стиск і кешування даних, прискорену обробку HTTP запитів від серверів. Пристрій забезпечує uptime застосунків до 99.999%, багаторазово збільшуючи продуктивність. Це значно скорочує витрати на інфраструктуру й дозволяє обслуговувати більше користувачів.

Функція глобального балансування навантаження (GSLB) дозволяє розподілити трафік між декількома географічними локаціями, забезпечуючи аварійне відновлення й скорочення часу відгуку. Адміністратор може встановлювати політики, автоматично перенапрямні трафік на основі доступності сайту, продуктивності дата центра й рівня затримок у мережі. Завдяки цьому

FortiADC гарантує високі показники надійності й доступності застосунків. Контролер здатний виконувати функції Web Application Firewall, Web Filtering і підтримує технологію Reputation IP Filter, що дозволяє забезпечити захист від DoS / DDoS атак, фішингу, спаму, шкідливого програмного забезпечення й ботнетів.

Модельний ряд Fortinet FortiADC

Самими продуктивними в сімействі FortiADC є контролери 1000F, 2000F і 4000F. Вони призначені для використання у великих дата центрах з більшим обсягом застосунків. FortiADC 200F, 300D і 400D будуть ідеальним рішенням для великих і середніх компаній, а моделі 60F і 100F зможуть забезпечити потреби невеликих організацій.

Для віртуальних середовищ Fortinet пропонує FortiADC VM01, 02, 04,08, 16 і 32.

Сучасний ADC: стан ринку

На закінчення приведемо деякі приклади пропонованих вендорами продуктів.

F5 нарощує обсяги продажів, регулярно випускаючи нові версії ПЗ й апаратних платформ. У липні вийшла BIG-IP 11.4 з поліпшеною масштабованістю й новими функціями SDN. У лютому F5 придбала LineRate – розроблювача технології для мережних сервісів рівня застосунків для середовища SDN. У числі інших недавніх придбань – Scale. Її досвід у багатоарендній віртуалізації буде, безсумнівно, корисний для поліпшення горизонтального й вертикального масштабування контролерів доставки застосунків від F5. Недавно F5 обновила свою модель ліцензування з розрахунком на провайдерів хмарних сервісів. Крім того, тепер контролери F5 поряд з VMware підтримують всі основні гіпервізори, включаючи KVM.

Контролер доставки застосунків Citrix NetScaler оптимізований з урахуванням вимог сучасного ЦОД. На думку аналітиків, у ньому застосовуються одні із самих зроблених у галузі засобів балансування

навантаження, кешування й стиску даних. У результаті досягається висока продуктивність. Убудована технологія TriScale дозволяє створювати масштабовані відказостійкі кластерні конфігурації – до 32 активних вузлів. TriScale означає можливість масштабування в «трьох вимірах»: нарощування потужності одного пристрою при покупці додаткових ліцензій, побудова кластера, де кожний вузол бере на себе частина навантаження, а весь кластер функціонує як один ADC, а також запуск до 40 незалежних віртуальних пристроїв ADC в архітектурі NetScaler SDX.

Поділюваний контролер SDX забезпечує ефективний захист даних за рахунок повної ізоляції віртуальних пристроїв. Віртуальні контролери в NetScaler SDX прискорюють розгортання застосунків і сервісів, а розроблювачі можуть тестувати нові продукти, не побоюючись завдати шкоди робітничому середовищу. Віртуальний ADC поводиться точно так само, як фізичний. Версія NetScaler SDX з віртуальними контролерами дає можливість консолідувати інфраструктуру.

NetScaler Cloud-Bridge поширює функції ADC із ЦОД на зовнішню хмару, підтримуючи на належному рівні продуктивність при використанні внутрішніх і зовнішніх ресурсів, спрощує розгортання гібридних хмар. Як і ADC від F5, NetScaler передбачає убудовану інтеграцію із сервісами IaaS від Amazon Web Services, дозволяючи замовникам додавати або видаляти потужності в міру необхідності.

Не забули розроблювачі й про мобільні обчислення. Зокрема, NetScaler може динамічно доставляти контент на різні мобільні пристрої, а підтримка протоколів класу Multipath TCP забезпечує постійний доступ при перемиканні між мережами 3G/4G і бездротовою корпоративною мережею, тобто з'єднання не переривається. Для керування мобільними пристроями й створення бібліотек зі схваленими адміністратором мобільними застосунками NetScaler можна інтегрувати з Citrix XenMobile. Програмне забезпечення NetScaler Insight Center і NetScaler Action Analytics забезпечує моніторинг трафіку і його аналіз у

реальному часі, динамічне виконання заданих правил. По відкликаннях замовників, NetScaler досить просто настроїти й запустити в роботу. Усе робиться за допомогою завдання правил – ніякого програмування й написання сценаріїв не буде потрібно.

У червні Citrix анонсувала програму міграції ACE Migration Program (AMP), що буде діяти до грудня 2019 року. Представлена у квітні 2012 року технологія Citrix TriScale для NetScaler передбачає кілька моделей масштабування ADC: оплата в міру росту, кластеризація або консолідація на єдиній багатоарендній платформі. Щоб справлятися з піковими навантаженнями, організації можуть використовувати пакети NetScaler Burst Pack для тимчасового (на період до 90 днів) нарощування потужності.

Radware продовжує додавати у свою продуктову лінійку нові засоби, в основному дотичній безпеці й протидії атакам, особливо в області SDN. Разом з Juniper компанія Radware повністю інтегрувала свій ADC з операційною системою Juniper Networks JunOS. А користувачі IBM SmartCloud можуть застосовувати віртуальні ADC Radware у приватній, публічній або гібридній хмарі. Radware також працює з VMware і Red Hat над інтеграцією своїх продуктів з технологіями віртуалізації. Контролери Radware уже підтримують не тільки гіпервізор VMware, але й KVM, Open Xen.

Компанія A10 Networks стрімко нарощує в останні роки свій бізнес. Раніше віртуальний контролер доставки застосунків A10 SoftAX підтримував тільки VMware ESX, але тепер працює в середовищі Citrix XenServer і KVM. В області SDN цей вендор співробітничає з NEC: SoftAX-2 можна використовувати в комбінації з NEC ProgrammableFlow Controller.

Компанія Brocade поставляє свій ADX ADC ряду вендорів як OEM-рішення. Її лінійка контролерів доставки застосунків поповнилася новим програмним продуктом Brocade Virtual ADX. Віртуальний контролер трафіку застосунків Brocade vADX являє собою віртуальну машину з повним функціоналом, властивим апаратним контролерам трафіку застосунків. Він

дозволяє здійснювати балансування трафіку віртуальних машин прямо на рівні сервера, замовникові немає необхідності в цьому випадку створювати додатковий рівень мережної інфраструктури.

Розроблювачі ще пари років тому усвідомили необхідність розширення функцій за рамки оптимізації WAN. Компанія вийшла на ринок ADC, придбавши британську Zeus Technologies. Програмне забезпечення останньої було використано при створенні лінійки ADC і поповненні продуктів для керування продуктивністю застосунків. Воно є також ключовим компонентом рішень AWS і допомагає розвивати партнерство з такими великими виробниками застосунків, як Oracle. Недавно її ADC Traffic Manager був сертифікований SAP відразу по трьох категоріях: надійність, доступність і безпека.

Array Networks, Barracuda і Kemp Technologies знайшли на ринку ADC свої власні ніші. Array Networks зміцнилася на азіатському ринку ADC і має там значну частку, але розширює своя присутність і в США. Поряд з ADC вона пропонує також продукти оптимізації WAN і шлюзи безпеки. Рік назад Array Networks випустила віртуальний пристрій vAPV, що функціонує на її апаратній платформі AVX і в середовищі віртуалізації VMware або Citrix.

Barracuda Networks пропонує за конкурентними цінами продукти безпеки, зберігання даних і доставки застосунків. Аналітики відзначали такі достоїнства продуктів Barracuda, як гарні функції балансування навантаження, але в цілому вважали її платформу ADC недостатньо функціональної, однак у квітні Barracuda нарешті випустила повноцінний пристрій ADC.

Продукти KEMP Technologies також уважаються відмінними балансувальниками навантаження, до того ж з дуже привабливим співвідношенням ціни й продуктивності, хоча й не називаються ADC. KEMP доповнює їхніми засобами автентифікації й SSO, пропонує уніфіковане рішення для розгортання на декількох площадках. Ця компанія має стратегічне партнерство з Cisco і Dell, що дозволяє їй розширювати клієнтську базу.

Продукти й технології компанії Coyote Point, що також раніше вважалася нишевим гравцем, після її придбання Fortinet доповнили портфель Fortinet Network Security.

Тим часом у галузі набирає популярність новий підхід – «ADC як послуга» (Application Delivery as-a-Service, ADCaaS). Так, півроку назад компанія анонсувала нову платформу для хмарних провайдерів, за допомогою якої вони можуть пропонувати послуги ADCaaS на базі ADC Services Controller. Як заявляється, динамічна й «еластична» інфраструктура доставки застосунків надає провайдерам інструменти для автоматичного виділення ресурсів, розгортання, ліцензування й виміри сервісів застосунків.

ADC Services Controller також передбачає можливість розгортання віртуальних контролерів ADC Traffic Manager на стандартних серверах. Віртуальні STM забезпечують масштабування сервісів ADC на вимогу відповідно до використовуваного в ЦОД застосунками.

Тим самим усуваються недоліки традиційних ADC, при використанні яких часто потрібно підтримувати дорогі й потужні платформи, коли в них ще немає необхідності, а також впроваджувати складні багатоарендні рішення, хоча в підсумку виходять обмежено масштабовані, погано адаптируемі й не «програмно обумовлені» системи, що утрудняють розгортання ADC і керування ними у віртуальному середовищі. ADCaaS називають новою парадигмою, однак по суті це все ті ж віртуальні контролери доставки застосунків.

Не виключено, що роль ADC буде мінятися й відповідні проблеми продуктивності будуть вирішуватися вже на рівні контролерів SDN – як у класичному варіанті з OpenFlow, так і в пропрієтарних рішеннях вендорів на зразок Cisco ACI. Та й маршрутизатори «обростають» всі новими функціями. Так, наприклад, маршрутизатори Cisco ISR 2911 можуть вирішувати завдання оптимізації доставки застосунків, забезпечення QoS, пріоритизації трафіку застосунків, однак роблять це на канальному рівні, а не на рівні застосунків, як ADC. У кожному разі в найближчому майбутньому ADC не залишаться без роботи.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Python – це об’єктно-орієнтована мова програмування високого рівня загального призначення з відкритим кодом. Це визначення може бути важким для новачків, тому розглянемо кожну характеристику окремо, щоб зрозуміти, що вона означає:

– Відкритий вихідний код: це безкоштовно та доступно для подальших покращень, таких як додавання корисних функцій або виправлення помилок.

– Об’єктно-орієнтована: заснована не на функціях, але в об’єктах з певними атрибутами й методами.

– Високий рівень: зручний для людини, а не для комп’ютера.

– Загальне призначення: можна використовувати для створення будь-яких програм.

Ця мова використовується в будь-якому програмному забезпеченні, про яке ви тільки можете подумати. Ви можете використовувати його для створення веб-сайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу та багато іншого. Також застосовується в науці про дані, аналізі даних, машинному навчанні, інженерії даних, веб-розробці, розробці програмного забезпечення та інших галузях.

Переваги та недоліки Python

Переваги:

– Її легко читати, вчити та писати. Це мова програмування високого рівня з англійським синтаксисом. Це полегшує читання та розуміння коду. Її дійсно легко зрозуміти і вивчити, тому багато людей рекомендують Python новачкам. Вам потрібно менше рядків коду для виконання того ж завдання в порівнянні з іншими основними мовами, такими як C/C++ та Java.

– Підвищує продуктивність. Це дуже продуктивна мова. Завдяки її простоті розробники можуть зосередитися на розв’язанні проблеми. Їм не

потрібно витрачати багато часу на розуміння синтаксису або поведінку мови програмування. Ви пишете менше коду та виконуєте більше завдань.

– Інтерпретована мова. Python мова, що інтерпретується, а це означає, що вона безпосередньо виконує код по рядку. Якщо сталася помилка, вона зупиняє подальше виконання та повідомляє про її виникнення. Вона показує лише одну помилку, навіть якщо у програмі їх кілька. Це спрощує налагодження.

– Динамічно типізована. Python не визначає тип змінної, доки ми не запустимо код. Вона автоматично надає тип даних, коли відбувається процес виконання. Фахівець може не турбуватися про оголошення змінних та типи даних.

– Безкоштовна та з відкритим вихідним кодом. Ця мова постачається під схваленою OSI ліцензією з відкритим вихідним кодом. Це робить його безкоштовним для використання та розповсюдження. Ви можете завантажити вихідний код, змінити його та навіть розповсюджувати свою версію. Це корисно для організацій, які хочуть використати свою версію для розробки.

– Підтримка великих бібліотек. Стандартна бібліотека Python є величезною, ви можете знайти майже всі функції, необхідні для вашого завдання. Таким чином ви не залежите від зовнішніх бібліотек.

– Портативність .У багатьох мовах, таких як C/C++, потрібно змінити свій код, щоб запустити програму на різних платформах. З Python все інакше. Ви тільки пишете один раз і запускаєте її будь-де.

Недоліки:

– Низька швидкість. Вище ми обговорювали, що це інтерпретована мова з динамічною типізацією. Порядкове виконання коду часто призводить до повільного виконання. Динамічна природа Python також є причиною її низької швидкості, оскільки їй доводиться виконувати додаткову роботу при виконанні коду. Тому вона не підходить для цілей, де швидкість важливий аспект проєкту.

– Неефективна для пам’яті. Ця мова програмування використовує великий обсяг пам’яті, це може бути недоліком при створенні програм, коли віддають перевагу оптимізації пам’яті.

– Слабка у мобільних обчисленнях. Python зазвичай використовується у серверному програмуванні. Ми не бачимо – її на стороні клієнта або в мобільних програмах з таких причин: вона не заощаджує пам’ять і має повільну обчислювальну потужність у порівнянні з іншими мовами.

– Доступ до бази даних. Програмувати на цій мові легко, але коли ми взаємодіємо з базою даних, її не вистачає. Рівень доступу до бази даних у Python примітивний та недостатньо розвинений у порівнянні з іншими популярними технологіями.

– Помилки виконання. Це мова з динамічною типізацією, тому тип даних змінної може змінюватись у будь-який час. Змінна, що містить ціле число, у майбутньому може містити рядок, що може призвести до помилок виконання.

Застосування Python:

– Для аналізу даних. Дані стали цінним активом у будь-якій сучасній галузі, і більшість компаній зацікавлені у збиранні, обробці та аналізі релевантних даних, щоб витягти з них цінну інформацію для бізнесу. І тут Python виходить за межі будь-якої конкуренції. Python особливо цінна тим, що крім великої стандартної бібліотеки надає величезний набір додаткових модулів, розроблених спеціально для аналітичних цілей. Найвідоміші бібліотеки Python для аналізу даних – це pandas і NumPy . Ці інструменти дозволяють робити з вашими даними майже все, наприклад, очищати і аналізувати їх, вивчати статистику або візуалізувати приховані тенденції у ваших даних.

– Для візуалізації даних. Візуалізація даних – це окрема частина аналізу даних, яка допомагає нам подавати інформацію, необроблену чи очищену, у більш змістовній формі. Тут Python знову входить у гру, пропонуючи широкий спектр інструментів візуалізації даних. Найпопулярніші з них – matplotlib і

заснований на ній seaborn. Використовуючи їх, ми можемо створювати буквально всі види візуалізації: від найпростіших до складніших.

– Для машинного навчання. Машинне навчання (ML) є основою більшості завдань науки даних. Він є областю штучного інтелекту, пов'язаною з використанням алгоритмів, що дозволяють машинам вивчати закономірності та тенденції на основі історичних даних, щоб робити прогнози на основі невідомих даних. – Використовуючи методи ML, ми можемо створювати моделі, які можуть точно передбачити швидкість відтоку клієнтів компанії, оцінити ризик виникнення у людини певного захворювання, визначити оптимальне розташування автомобілів таксі й т.д. За допомогою Python ми можемо побудувати модель ML, використовуючи лише три рядки коду.

– Для розробки програмного забезпечення. Крім свого багатостороннього застосування в галузях науки про дані, Python використовується на кожному етапі розробки програмного забезпечення, включаючи контроль складання, автоматичну безперервну компіляцію, прототипування, відстеження помилок, тестування та обслуговування програмного забезпечення. За допомогою цієї мови можемо створювати аудіо- або відеопрограми на основі методів штучного інтелекту, машинного навчання, API (інтерфейсів прикладного програмування), GUI (графічних інтерфейсів) або будь-якого іншого типу програмного забезпечення.

– Для веброзробки. У той час як для створення візуальної частини вебсайту ми переважно будемо використовувати такі мови, як HTML, CSS та JavaScript, для його невидимої частини ми часто вибираємо Python. Серед масштабних вебсайтів та програм, створених за допомогою цієї мови, варто згадати Google, Facebook, Instagram, YouTube, Dropbox та Reddit.

– Для автоматизації задач/скриптингу. Це відмінний інструмент для написання програм для автоматизації різних завдань, що повторюються. Цей процес називається скриптингом. Зокрема, можна робити скрипти для роботи з файлами та папками. Наприклад, можна створювати, перейменовувати,

перетворювати, розділяти, об’єднувати або видаляти файли, перевіряти їх на наявність помилок. Ви також можете використовувати автоматизацію Python для пошуку та завантаження інформації з Інтернету, заповнення та надсилання онлайн-форм та надсилання регулярних повідомлень або електронних листів.

Яким фахівцям потрібно володіти Python:

– Фахівець з даних.

– Аналітик даних.

– Інженер даних.

– Інженер з машинного навчання.

– Журналіст даних.

– Архітектор даних.

– Повний стек веб-розробника.

– Backend-розробник.

– DevOps-інженер.

– Інженер-програміст.

Можемо зробити висновок, що Python ще довго буде популярною мовою, хоч і має низку недоліків. Цю мову використовують для створення вебсайтів, штучного інтелекту, серверів, програмного забезпечення для бізнесу, аналізу даних, машинного навчання, інженерії даних та для багатьох інших областей. Це перспективна і затребувана навичка, яка необхідна у всіх галузях.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи реалізації інтелектуального контролера доставки додатків ADC.

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

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

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

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

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

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

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

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

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

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

У даній роботі представимо платформу для керування доставкою застосунків Application Delivery Control (ADC) як сервіс, що може обладнати будь-якого провайдера хмарних послуг інструментами й технологіями, необхідними для розгортання й керування динамічними й гнучкими інфраструктурами доставки застосунків.

Технологія надання ADC як послуги забезпечує:

– Скорочення часу підготовки до роботи: від декількох тижнів до негайного розгортання.

– 50%-я економія: гнучкі ADC платформи необхідного розміру.

– Підвищення зручності роботи користувачів: тонке настроювання окремих застосунків.

– Поліпшена безпека й продуктивність: забезпечення автономності й масштабованості

Stingray Services Controller

Контролер послуг Stingray забезпечить клієнтам автоматичне постачання, розгортання, ліцензування й керування пакетом з тисяч ADC при використанні сервісної моделі.

Рішення також надасть нову модель споживання для замовників, які використовують послуги ADC, за назвою «мікро» копія ADC Traffic Manager. Використовуючи таку “мікро” копію, ADC послуги тепер можуть гнучко масштабуватися на вимогу й мати підходящий розмір для відповідних застосунків у центрах обробки даних, пропонуючи при цьому високу щільність, повну автономність і мультиарендне масштабування.

Програмне забезпечення ADC Traffic Manager

ADC Traffic Manager забезпечує вашим кінцевим користувачам високу доступність і продуктивність застосунків. Цей повнофункціональний віртуальний контролер доставки застосунків ADC (application delivery controller) робить набагато більше, ніж просте балансування робочого навантаження веб сайтів і хмарних сервісів ваших підприємств. Він також управляє послугами й оптимізує їх для кінцевих користувачів шляхом огляду, перетворення, установлення пріоритетів і маршрутизації трафіку застосунків. Безкоштовна версія програмного забезпечення ADC Traffic Manager уже доступна для завантаження.

Завдання:

– Забезпечення надійної роботи для веб-сайтів, порталів і хмарних застосунків.

– Впровадження нових послуг і їхнього масштабу відповідно до бізнесів-потреб.

– Підтримка високої якості користування послугами під час стрибків трафіку.

Рішення:

– Зробити застосунки більше надійними, використовуючи локальну й глобальну балансування навантаження.

– Створення, керування й доставка послуг рівня застосунків швидше й з меншими витратами.

– Керування й оптимізація послуг для кінцевих користувачів шляхом диференціації й пріоритизації трафіку застосунків.

Програмне забезпечення ADC Application Firewall

Захистите ваші веб-застосунки від відомих і невідомих погроз, використовуючи програмне забезпечення – ADC Application Firewall. Цей динамічний інструмент безпеки забезпечує захист публічних, приватних і гібридних веб-сервісів на рівні застосунків. Його можна легко розгорнути у вигляді автономного брандмауера для веб-застосунків або як додатковий компонент до програмного забезпечення ADC Traffic Manager.

Проблеми:

– Збільшення кількості складних атак на сайти.

– Погрози розвиваються й міняються швидко, використовуючи нові слабкі місця ПЗ.

– Збільшення тягаря сумісності, включаючи стандарт PCI DSS.

Рішення:

– Виявлення, попередження, і блокування атак на рівні застосунків.

– Регулярне відновлення основних мір захисту.

– Використання розподіленої архітектури для високої продуктивності у всіх середовищах.

Програмне забезпечення ADC Aptimizer

Збільшення швидкості роботи веб-застосунків, задоволені користувачі, вільні розроблювачі – це трохи з багатьох переваг використання програмного забезпечення ADC Aptimizer. Складні методи оптимізації веб-контенту автоматично прискорюють роботу публічних веб-сайтів, інтрамереж, хмарні додатки й розгорнуті системи Microsoft SharePoint.

Проблеми:

– Низька задоволеність споживачів послугами, у зв'язку з повільним завантаженням сторінок

– Повільна реакція сторінок при роботі з корпоративними застосунками, такими як Microsoft SharePoint

– Розроблювачі витрачають занадто багато часу на оптимізацію сервісів для мобільних пристроїв і повільних мереж

Рішення:

– Регулювання продуктивності за допомогою злиття файлів, стиску, кешування, динамічної розмітки

– Зменшення кількості двосторонніх обігів браузера

– Автоматизація оптимізації застосунків для всіх пристроїв і браузерів і звільнення часу розроблювачів

Програмне забезпечення ADC Services Controller

Навіщо платити за потужності ADC, які ви не використовуєте? ADC Services Controller автоматизує розгортання, ліцензування й облік ваших послуг доставки застосунків. Він дає кожному з застосунків виділений екземпляр ADC, у мультиарендній платформі високої щільності. Нові послуги дозволяють використовувати бізнес-модель, у якій ви контролюєте свої витрати. Ви можете розподіляти витрати по кожному клієнтському додатку на основі погодинного обліку, щоб запропонувати ADC як послуги для ваших клієнтів і застосунків.

Завдання:

– Забезпечення надійної роботи для веб-сайтів, порталів і хмарних застосунків

– Впровадження нових послуг і їхнього масштабу відповідно до бізнес-потребами

– Підтримка високої якості користування послугами під час стрибків трафіку

Рішення:

– Зробити застосунку більше надійними, використовуючи локальну й глобальну балансування навантаження

– Створення, керування й доставка послуг рівня застосунків швидше й з меншими витратами

– Керування й оптимізація послуг для кінцевих користувачів шляхом диференціації й пріоритизації трафіку застосунків

3.2 Розробка структурної схеми

Контролери доставки застосунків (Application Delivery Controller, ADC) забезпечують прискорення роботи застосунків і балансування навантаження між серверами. Звичайно вони розміщаються в центрі обробки даних між міжмережевим екраном і серверами застосунків.

«Контролери доставки застосунків ведуть походження від балансувальників навантаження, базовою функцією яких є зниження навантаження на сервери шляхом рівномірного розподілу трафіку між ними. Їх можна вважати наступним поколінням балансувальників навантаження з такими розширеними функціями, як маршрутизація й керування трафіком на рівні застосунків, зниження навантаження при шифруванні SSL і стиску даних. Використання балансувальників і контролерів доставки застосунків дозволяє забезпечити безперервність роботи при відмові маршрутизаторів і серверів застосунків, а також компенсувати обмеженість ресурсів серверів, зокрема, звільнивши їх від шифрування SSL.

Віртуалізація, хмарні обчислення й BYOD вимагають трансформації ЦОД, і тут корисними виявляються ADC. Вони допомагають прискорити розгортання нових застосунків і сервісів і забезпечити високу доступність і продуктивність в умовах стрімкого зростання обсягів трафіку. При використанні мобільних пристроїв ADC дозволяють вилученим користувачам комфортно працювати зі своїми застосунками й даними – майже, як у себе в офісі. Для цього використовуються такі засоби, як кешування, стиск даних, керування сеансами й прозорою маршрутизацією до найближчого ЦОД. Нові покоління контролерів доставки застосунків застосовуються для більше широкого кола завдань, у тому числі для керування трафіком і розвантаження SSL, відіграють роль міжмережевого екрана для Web-застосунків (WAF) і т.д.

Завдання ADC

ADC вирішують завдання доставки застосунків кінцевим користувачам із прийнятною затримкою й швидкістю реакції на дії останніх (див. рисунок 3.1). Тим самим їхнє застосування дозволяє оптимізувати продуктивність на стороні ЦОД і мережної інфраструктури. Класичні ADC, називані контролерами трафіку застосунків, вирішують кілька основних завдань: забезпечення масштабування серверних застосунків і мережної інфраструктури замовника; підвищення рівня захисту застосунків і сервісів; оптимізація ресурсів серверів застосунків і створення для серверів застосунків конфігурацій високої доступності.

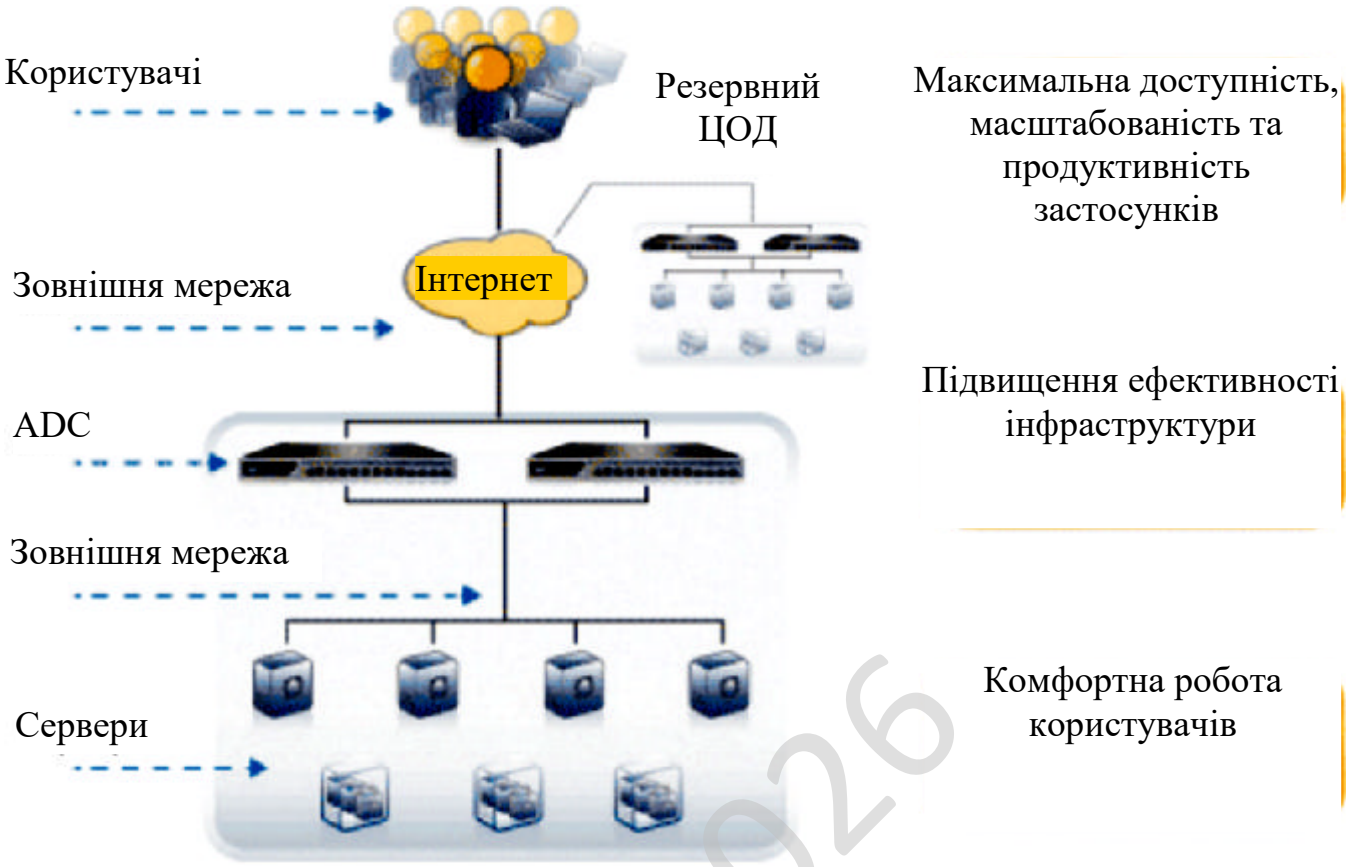

Рисунок 3.1 – Структурна схема системи

Розповсюдженя хмарних сервісів і засобів віртуалізації вкупі з підвищенням мобільності користувачів сприяє розвитку ринку й технологій контролерів доставки застосунків. Нова функціональність дозволяє забезпечити захист переданих даних і контроль пристроїв, засобу автоматизації й відказостійкости. Розвиток хмарних технологій диктує необхідність підтримки географічно розподілених кластерів з адаптивними функціями балансування трафіку. А збільшення продуктивності серверів уможливлює використання віртуальних контролерів доставки застосунків.

Контролери трафіку застосунків можуть мати різні області використання. От деякі із завдань, розв'язуваних ADC:

– балансування й комутація трафіку рівня L4-7 по моделі OSI;

– балансування й комутація поштового трафіку;

– прозора комутація трафіку кеш-серверів;

– балансування й комутація трафіку віртуальних машин, інтеграція з гіпервізорами VMware і Microsoft Hyper-V;

– балансування міжмережевих екранів, а також захист мережної інфраструктури ЦОД замовника від атак DoS.

Все більшу популярність здобуває використання контролерів трафіку застосунків для забезпечення відказостійкости територіально розподілених об'єктів замовника за допомогою протоколу Global Server Load Balancing (GSLB).

Інше важливе завдання – забезпечення надійного й постійного доступу при переміщенні користувачів між мережами операторів зв'язку й публічних крапок доступу. Вирішити її непросто. Для цього мережна інфраструктура повинна підтримувати високу продуктивність і безперебійну роботу в провідних і бездротових мережах незалежно від виду трафіку, масштабуватися з використанням внутрішніх ресурсів або публічних хмар, забезпечувати детальний контроль і керування трафіком.

Безпека мережі також має критичне значення, тому практично повсюдно використовується SSL, однак шифрування може негативно впливати на продуктивність. Нарешті, щоб знизити непродуктивні втрати, інфраструктуру ЦОД, як і розгортання й адміністрування застосунків, потрібно спростити, а гнучкість інфраструктури повинна сприяти її адаптації до мінливих вимог сервісів і застосунків.

ADC уважаються відмінним інструментом для досягнення перерахованих цілей. Підвищення продуктивності застосунків і ефективний захист – їхні базові завдання. Контролери застосунків абстрагують сервіси застосунків і дозволяють IT-підрозділам гнучко розподіляти ресурси. Однак у реальності більшість ADC не цілком відповідають вимогам віртуалізованих, хмарних ЦОД і інфраструктур, що обслуговують мобільних користувачів.

Одне з перешкод – схема ліцензування, при якій потрібно заздалегідь купувати встаткування з необхідною пропускною здатністю, навіть якщо його

потужності довгий час будуть задіятися не повністю. Крім того, більшість сучасних ADC не мають достатню продуктивність для впровадження повсюдного шифрування SSL, у них є лише обмежені засоби для підтримки хмарної архітектури й мобільних користувачів. Деякі успадковані ADC здатні підтримувати гібридні хмари. У найкращому разі вони трохи поліпшують продуктивність мобільних з'єднань або керування корпоративними й операторськими мережами. Більшість контролерів застосунків не передбачають також можливостей аналізу трафіку, вони можуть бути складними в інсталяції й обслуговуванні.

З огляду на прагнення замовників до зниження експлуатаційних витрат і підвищенню операційної ефективності, необхідно, щоб ADC підтримували одночасно кілька засобів керування, включаючи класичний командний рядок (CLI), Web-керування, SNMP, API-Інтерфейс, OpenStack, CloudStack.

Віртуалізація контролерів доставки застосунків

Протягом останнього року практично всі вендори ADC внесли в них досить серйозні зміни. На ринок ADC впливають кілька факторів. Насамперед це віртуалізація, програмно конфігуруємі мережі (SDN), хмарні обчислення й інформаційна безпека.

Перехід до хмарної моделі міняє традиційні моделі надання сервісів і доставки застосунків, при цьому він припускає високий рівень віртуалізації IT-Інфраструктури й більше ефективне використання мереж. Найбільше помітний вплив віртуалізації – це свого роду «побічний ефект» серверної віртуалізації. Віртуальні ADC заміняють фізичні пристрої, а управляти ними можна як єдиним пулом, перерозподіляючи ресурси між віртуальними контролерами застосунків.

Більшість організацій шукають шляхи зниження витрат і підвищення ефективності бізнесу. Для цих цілей широко використовуються засоби віртуалізації. У той же час контролери доставки застосунків самі по собі здатні істотно оптимізувати витрати на обслуговування серверів застосунків. При віртуалізації контролерів доставки застосунків досягається синергетичний ефект.

Це підвищення доступності застосунків до рівня 99,999% за рахунок усунення критичних крапок відмови, зниження витрат на підтримку інфраструктури й керування на 70%, зниження витрат на розміщення й забезпечення встаткування електроживленням на 60%, скорочення серверної інфраструктури на 40%, зниження вимог до необхідної пропускної здатності мережного встаткування на 30%.

У цілому використання віртуальних контролерів трафіку застосунків значно підвищує живучість і ефективність ЦОД. При цьому зменшуються капітальні витрати, знижується розмір витрат на абонента, поліпшуються характеристики доставки сервісу (SLA). А завдяки підтримці протоколів OpenStack і CloudStack віртуальним контролером трафіку застосунків Brocade vADX можна управляти за допомогою централізованої системи оркестрації стороннього виробника.

Якої ж проблеми виникають при віртуалізації контролерів доставки застосунків і як вони вирішуються? Основна проблема пов'язана із продуктивністю всього комплексу в цілому. При віртуалізації ресурси платформи віртуалізації використовуються всіма застосунками, які базуються на даному сервері. Тому дуже важливо контролювати коректний розподіл ресурсів і піклуватися про гарантоване виділення необхідних потужностей контролерам доставки застосунків (сучасні засоби віртуалізації дозволяють це робити). Ще одна проблема полягає в тім, що розроблювач контролерів доставки застосунків не може гарантувати продуктивність віртуальної системи, тому що цей показник залежить у першу чергу від характеристик базової серверної платформи.

Кластеризація ADC

Кластеризація ADC підвищує продуктивність і доступність комплексу, що здійснює балансування й доставку застосунків. Кластеризація потрібно в тих випадках, коли необхідно гарантувати високий рівень доступності застосунків. При цьому варто чітко розуміти різницю між двома можливими режимами кластеризації: рівномірним розподілом навантаження між вузлами кластера й

резервуванням вузлів, при якому активна тільки частина встаткування. Щоб виключити перевантаження вузлів кластера, рекомендується вибирати другий режим, при якому частина встаткування перебуває в гарячому резерві, а в пікові показники завантаження активної частини не перевищує 70%. У цьому випадку при відмові одного з вузлів кластера не виникне ситуація, коли інші вузли виявляться перевантажені й не здатні впоратися з покладеної на них завданням. Простий приклад: кластер складається із двох вузлів і працює в режимі балансування навантаження. Якщо кожний вузол завантажений на 50%, то при виході з ладу одного з контролерів на частину, що залишилася, ляже подвійне навантаження.

Дуже часто компанії, які мають потребу в контролерах доставки застосунків, використовують географічно рознесені ЦОД. У цьому випадку необхідний режим балансування ADC. Подібний варіант кластеризації часто називають Global Server Load Balancing. Географічно розподілені кластери ADC нерідко використовуються при хмарних обчисленнях. При цьому розподіл навантаження відбувається з урахуванням доступності ЦОД для оптимізації витрат на передачу трафіку.

Динаміка ринку

Сегмент ADC динамічно міняється, на ринку виникають нові альянси. Вендори, що раніше стримано ставилися до віртуалізації, більш уважно придивляються до хмарної моделі. Компанії, що займаються винятково ADC, прагнуть розширити свій бізнес за рахунок суміжних областей, зокрема, вбудовують у свої рішення функції забезпечення безпеки й захисти від погроз, моніторингу мережної інфраструктури й продуктивності.

Більше 80% прибутку виробники ADC поки що одержують від продажу фізичних платформ, але, за прогнозами, поставки віртуальних контролерів застосунків будуть рости щорічно на десятки відсотків. Що стосується безпеки, трохи виробників уже оснастили свої ADC функціями міжмережевого екрана.

По даним Infonetics Research, світовий ринок контролерів доставки застосунків виріс в II кварталі 2019 року на 4% у порівнянні з попереднім роком. Аналітики з оптимізмом оцінюють перспективи даного сегмента, відзначаючи поновлення відкладених раніше великих проектів і збільшення інвестицій в ADC.

У деяких компаній, включаючи A10 Networks, Array Networks, Barracuda, F5, Kemp Technologies і Radware, продажі ADC становлять більшу частину оберту. Контролери доставки застосунків поставляють також Brocade, Citrix, Riverbed/ADC і Coyote Point Systems (недавно придбана Fortinet), але в них це не основний продукт. Лідером світового ринку ADC залишається компанія F5. Її продукти середнього й початкового рівня – BIG-IP 4000 і BIG-IP 2000 – користуються високим попитом.

3.3 Розробка функціональної схеми

Функціональна схема розробленої системи зображена на рисунку 3.2. Існує не занадто багато способів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC. Найпростіший з них – збільшення смуги пропущення мережі за рахунок нарощування апаратних можливостей устаткування. Можна використовувати й такі прийоми, як завдання пріоритетів даних, організація черг, запобігання перевантажень і формування трафіку. Керування мережею за заданими правилами в перспективі повинне об'єднати всі ці способи в єдину автоматизовану систему, що буде гарантувати якість послуг абсолютно на всіх ділянках мережі.

Збільшення апаратної потужності, безсумнівно, є найбільш ефективним засобом балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC у корпоративній мережі. Тиск із боку конкурентів, необхідність підвищення ефективності виробництва, поява нових технологій, що дозволяють оснащувати спеціалізовані мікросхеми (ASIC) найрізноманітнішими функціями, – все це змушує постачальників комутаційного встаткування для

корпоративних мереж викидати на ринок усе більше швидкодіючі пристрої за цінами, порівнянним з вартістю моделей колишнього покоління.

Малоймовірно, що в доступному для огляду майбутньому даний підхід до балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC у корпоративних мережах перестане бути пріоритетним. Оскільки в корпоративних мережах вдається забезпечити гарантовану якість послуг, не прибігаючи до дорогої модернізації усього встаткування й серйозних змін у системі керування мережею, мережні адміністратори будуть звертати увагу й на програмні засоби, що дозволяють реалізувати балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC.

Отже, найбільше поширення, швидше за все, одержить комбінований підхід. Деякі виробники висловлюються на його користь, затверджуючи, що найкраще збільшувати пропускну здатність мережі не прямо, а за рахунок інтелектуальних можливостей устаткування, що має засоби забезпечення балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC. Правда, виробники мережних пристроїв навряд чи можуть бути об'єктивними в цьому питанні, тому що вони зацікавлені в збуті тих самих продуктів, які підтримують гарантовану якість послуг.

У глобальних мережах нарощування апаратних потужностей використовується рідше. Звичайно, зниження вартості смуги пропущення зробило б передачу даних по глобальних мережах доступною для більше широкого кола користувачів (і навіть трохи знизило б актуальність впровадження гарантованої якості послуг). Але в найближчому майбутньому вартість смуги пропущення в глобальних мережах буде залишатися досить високою, тому й нарощування апаратної потужності не стане настільки популярним, як у корпоративних мережах.

З рисунку видно, що розроблена система складається з наступних функціональних частин:

– Блок інтерфейсу користувача.

– Блок призначення пріоритетів.

– Блок організації та обслуговування черг.

– Блок управління навантаженням.

– Блок формування трафіку.

– Блок визначення параметрів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC.

– Блок примусового завдання параметрів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC.

– Блок моделювання завантаженості мережі.

–

– Блок моніторингу мережі.

– Блок дослідження можливостей механізмів WRED.

– Блок дослідження можливостей механізмів WFQ.

Розглянемо ці блоки більш детально.

Блок інтерфейсу користувача

Призначений для реалізації взаємодії користувача, або дослідника з системою.

Блок призначення пріоритетів

Нарівні з нарощуванням апаратного забезпечення мережі для реалізації балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC застосовуються й засобу типу завдання пріоритетів даних і організації черг. Маршрутизатори підтримують ці механізми протягом багатьох років, як і деякі з нових комутаторів для каналів Gigabit Ethernet. Однак ПЗ для керування мережею за заданими правилами, яких необхідно для практичного втілення цієї технології, поки не розроблено. Серед нових комутаторів такого класу можна назвати CoreBuilder 3500, CoreBuilder 9000 і SuperStack II компанії 3Com, пристрою серії Accelar фірми Bay Networks, SmartSwitch Router компанії Cabletron Systems, а також Catalyst 5000 і Catalyst 8000 виробництва Cisco.

Способи пріоритезації даних можна умовно підрозділити на явні й неявні.

При неявному призначенні пріоритетів маршрутизатор або комутатор автоматично привласнює послугам відповідні рівні, виходячи із заданих

адміністратором мережі критеріїв (наприклад, типу додатка для застосовуваного протоколу передачі або адреси джерела). Кожний вхідний пакет аналізується (фільтрується) на відповідність цим критеріям. Механізм неявної пріоритезації підтримують практично всі маршрутизатори.

Деякі комутатори теж здатні задавати пріоритети, але мають обмежений набір функцій. Так, комутатори можуть забезпечувати пріоритезацію даних по типу віртуальної корпоративної мережі, адресі джерела або адресата, але не використовують інформацію більш високого рівня (протокол передачі або тип додатка). Розроблювальні в цей час системи керування мережею за заданими правилами дозволять реалізувати більше зроблені схеми пріоритезації даних при роботі з такими комутаторами.

При явній пріоритезації даних користувач або додаток запитує певний рівень служби, а комутатор або маршрутизатор намагається задовольнити запит. Імовірно, самим популярним механізмом явної пріоритезації стане протокол IP Precedence (протокол старшинства), що одержав другу назву IP TOS (IP Type Of Service), – один з розділів четвертої версії протоколу IP.

IP TOS резервує спеціальне поле в заголовку пакета, де можуть бути зазначені ознаки балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC, що визначають час затримки, швидкість пропущення й рівень надійності передачі пакета. Однак найдеться небагато популярних додатків – за винятком мультимедійного ПЗ, – у які реалізована підтримка протоколу IP TOS.

Зараз розробляється протокол резервування ресурсів RSVP, що передбачає більше складний, чим в IP TOS, механізм передачі від додатка до машрутизатору запиту на гарантовану якість послуг. Як і IP TOS, протокол RSVP поки не одержав широкої підтримки розроблювачів – він реалізований лише в окремих типах маршрутизаторів. Поширення RSVP стримується через те, що не вирішені деякі питання, пов'язані із сумісністю різних мереж. До того ж застосування RSVP значно збільшує навантаження на маршрутизатори й може привести до зниження швидкодії цих пристроїв.

Видимо, у доступному для огляду майбутньому неявна пріоритезація, не потребуюча серйозних обчислювальних потужностей маршрутизатора, залишиться більше популярною, чим явна. Крім того, при явному завданні пріоритетів значно ускладнюється керування мережею. Кінцеві користувачі, швидше за все, будуть набудовувати своє програмне забезпечення на запит найвищого з можливих рівнів послуг. Відповідно, адміністраторові мережі прийдеться розробляти правила керування користувачами й, можливо, навіть побудовувати служби з гарантованою якістю для кожного користувача окремо.

Блок організації та обслуговування черг

Після того як переданим по мережі даним призначені відповідні пріоритети (за допомогою явних або неявних методів), потрібно визначити порядок передачі цих даних, задавши алгоритм обслуговування черг із необхідною якістю (рівнем балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC). По суті, черги являють собою області пам'яті комутатора або маршрутизатора, у яких групуються пакети з однаковими пріоритетами передачі. Алгоритм обслуговування черги визначає порядок, у якому відбувається передача пакетів, що зберігаються в ній. Зміст застосування всіх алгоритмів зводиться до того, щоб забезпечити найкраще обслуговування трафіку з більш високим пріоритетом за умови, що й пакету з низьким пріоритетом гарантується відповідна увага.

При використанні способів завдання явних і неявних пріоритетів алгоритм обробки черг визначає порядок їхнього обслуговування. Відповідно до цього алгоритму на кожні два пакети, переданих у мережу із черги 1 (з високим пріоритетом) доводиться по одному пакету із черг 2 і 3. Пакети з однаковими пріоритетами передаються за принципом FIFO (”першим прийшов – першим вийшов”).

Якщо в мережі виникає перевантаження, служба черг не гарантує своєчасного досягнення пункту призначення найбільш важливими даними.

Гарантується лише те, що ці пакети будуть передані раніше, ніж ті, що мають більш низький пріоритет.

Сучасні служби балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC вирішують таке завдання за рахунок резервування смуги пропущення. Кожній із черг (або їхніх груп) виділяється заздалегідь задана величина смуги пропущення, що гарантує певну смугу пропущення для черги з більш високим пріоритетом. Для критичних ситуацій, коли обсяг даних у черзі перевищує розміри смуги пропущення, в алгоритмах обслуговування звичайно передбачається передача трафіку з високим пріоритетом на смугу пропущення, “яка приналежить” чергам з низьким пріоритетом, і навпаки. Найпростіші алгоритми обслуговують кожну чергу за принципом FIFO. При цьому передача кадрів великого розміру, що мають високий пріоритет, може приводити до затримок трафіку іншого додатка з настільки ж високим пріоритетом, але меншим обсягом.

У більше складних алгоритмах уживає спроба “справедливої” обробки черг. Наприклад, алгоритм рівномірного пропорційного (або зваженого) обслуговування (WFQ – Weighted Fair Queuing), розроблений компанією Cisco, підрозділяє додатки на потребуючі великої й малої ширини смуги пропущення, а сама смуга пропущення розподіляється між всіма додатками нарівно. Слід зазначити, що основні виробники маршрутизаторів самі розробляють алгоритми обслуговування черг і використовують для їхнього опису власну термінологію.

Істотним недоліком сучасних маршрутизаторів і комутаторів є те, що вони підтримують мале число черг. Найчастіше виробники організують служби балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC, що використовують чотири черги, хоча чим більше черг, тим більше різних пріоритетів можна привласнити переданим пакетам і тим “більш справедливо” розподілити смугу пропущення між додатками. Наприклад, адміністратор у стані задати пріоритети таким чином, щоб перевага при передачі віддавалося пакетам, адресованим на більше віддалені вузли.

Блок управління балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC

Служба балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC дає можливість використовувати для керування мережею два важливих механізми – керування в умовах перевантаження й запобігання перевантажень. Перший з них дозволяє кінцевій станції відразу знижувати швидкість передачі даних, коли в мережі починається втрата пакетів. У протоколах TCP/IP і SNA цей механізм підтримується вже протягом декількох років. І хоча сам по собі він не гарантує якості передачі, при його використанні разом з механізмом запобігання перевантажень результати виявляються набагато кращими. У мережах TCP/IP механізм запобігання перевантажень застосовується досить давно, але лише в останні роки він стає стандартом “де-факто” для маршрутизаторів телекомунікаційних мереж і Internet.

Стандартним способом запобігання перевантажень у мережі стало застосування механізму випадкового виділення пакетів (Random Early Detection, RED). При заповненні черг вище певної критичної оцінки цей механізм змушує маршрутизатор вибирати із черги за випадковим законом деякі пакети й “втрачати” їх. Швидкість передачі даних станціями-відправниками знижується, що й дозволяє уникнути переповнення черги.

Механізм пропорційного випадкового виділення пакетів – WRED (Weighted RED) – можна вважати наступною, більше зробленою “версією” RED. Він передбачає, що вибір пакетів, які повинні “втратитися”, буде відбуватися з обліком їх пріоритезації згідно IP TOS.

Блок формування трафіку

Формування трафіку – це загальний термін, яким прийнято позначати різні способи маніпулювання даними для підвищення якості їхньої передачі. Один їхній таких способів – сегментація пакетів. У мережах ATM гарантовано високий рівень балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC досягається в тому числі й за рахунок малого розміру переданих пакетів (осередків – у термінології ATM). Максимальний час затримки при передачі будь-якого пакета мережі ATM – це час передачі одного осередку.

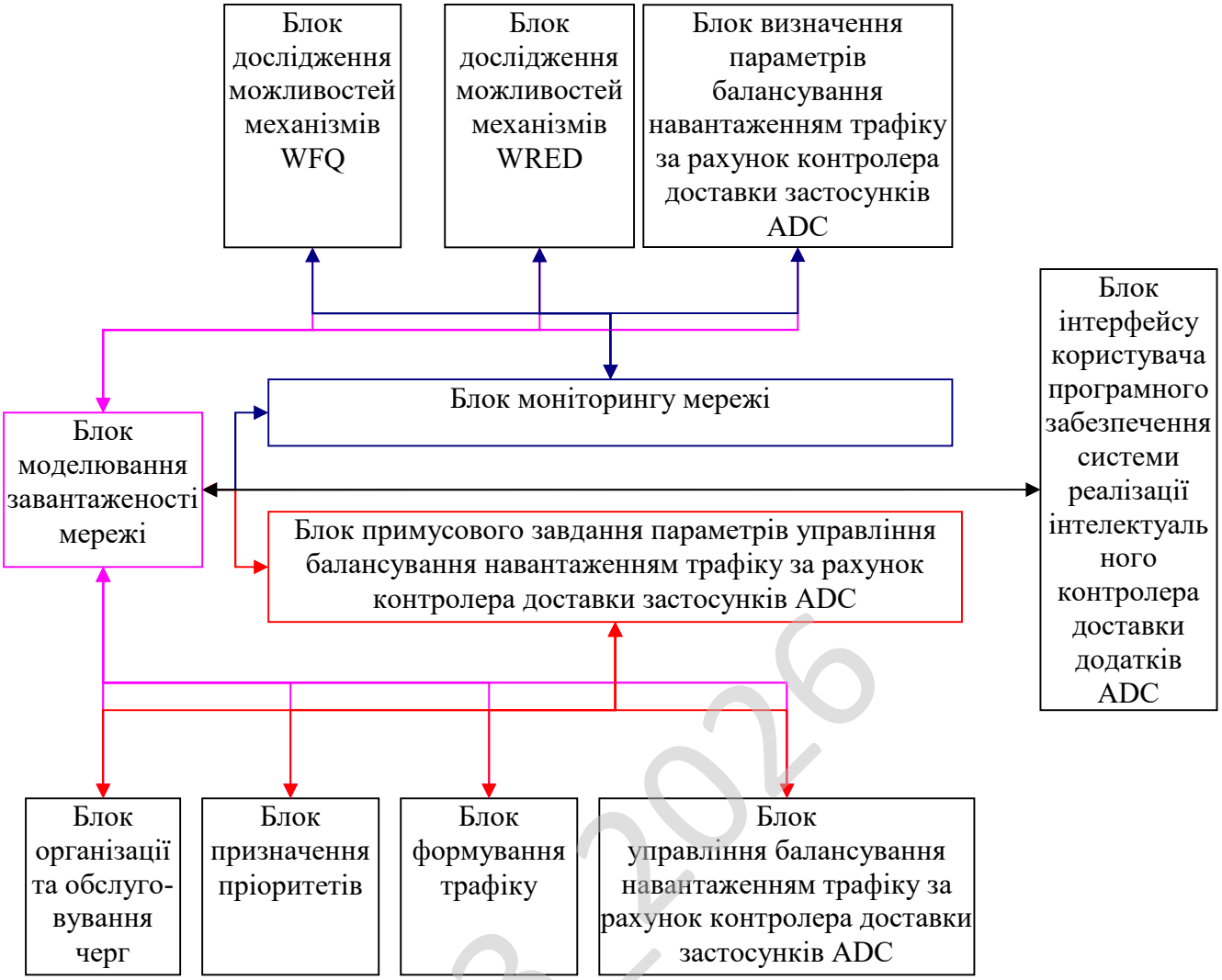

Рисунок 3.2 – Функціональна схема системи

Запозичаючи корисні механізми технології ATM, виробники маршрутизаторів і комутаторів починають забезпечувати у своїх продуктах можливість сегментації пакетів. Деякі пристрої, призначені для мереж frame relay, сегментують пакети, передані по каналах глобальних мереж, щоб гарантувати конкретний час передачі й мінімізувати затримки.

Ще один спосіб формування трафіку – його “вирівнювання”. Для таких протоколів, як наприклад, AppleTalk, характерна нерівномірна передача пакетів, що часом приводить до появи в мережі послідовностей або ланцюжків пакетів, а отже – до її перевантаження. Процедура вирівнювання трафіку дозволяє розчленувати ланцюжки шляхом розміщення пакетів у буфері перед їхньою

передачею в мережу. Для забезпечення більше рівномірної передачі даних можна також вирівнювати трафік кінцевих вузлів мережі.

Блок визначення параметрів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC

Призначений для визначення існуючих параметрів якості обслуговування (балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок примусового завдання параметрів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC

Призначений для примусового завдання одного, або декількох параметрів якості обслуговування (балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC). До них відносяться:

– Bandwidth (BW) – смуга пропущення, описує номінальну пропускну здатність середовища передачі інформації, визначає ширину каналу. Виміряється в bit/s (bps), kbit/s (kbps), mbit/s (mbps).

– Delay – затримка при передачі пакета.

– Jitter – коливання (варіація) затримки при передачі пакетів.

– Packet Loss – втрати пакетів. Визначає кількість пакетів, що відкидаються мережею під час передачі.

Блок моделювання завантаженості мережі

Призначений для моделювання завантаженості мережі з визначеним трафіком та заданими параметрами якості обслуговування (балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC).

Блок моніторингу мережі

Призначений для аналізу поточного стану мережі.

Блок дослідження можливостей механізмів WRED

Одним з методів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC, призначених для забезпечення необхідних вимог до різних потоків даних – запобігання перевантажень (congestion avoidance). Він заснований на обмеженні розмір черги, сигналізуючи джерелам даних про необхідність зменшити швидкість передачі інформації (WRED – Weighted Random early detection).

Блок дослідження можливостей механізмів WFQ

Другим з методів балансування навантаженням трафіку за рахунок контролера доставки застосунків ADC, призначених для забезпечення необхідних вимог до різних потоків даних – керування перевантаженням (congestion management). Він заснований на присвоєнні квот і пріоритетів потокам, і у випадку перевантаження, потоки одержують якість, обмежену їхньою квотою й пріоритетом (WFQ – Weighted Fair Queuing).

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

3.4 Розробка діаграми процесів

Розглянемо розроблену діаграму процесів яка зображена на рисунку 3.3. Основна будова діаграми процесів полягає у графічному представленні складу сукупностей даних, що характеризуються як співвідношення різних частин кожної з сукупностей. Склад статистичної сукупності графічно може бути представлений як за допомогою абсолютних, так і відносних показників. Графічне зображення складу сукупності по абсолютними і відносними показниками сприяє проведенню більш глибокого аналізу і дозволяє проводити аналіз системи.

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

систему що розробляється. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграми потоків даних містять чотири типи елементів:

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

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

– Сховища даних (репозиторії).

Рисунок 3.3 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

Розглянемо реалізацію бакалаврської дипломної роботи. Були проведені розрахунки і підібрані набори тестових даних для перевірки правильності реалізації проектних рішень.

Архітектура клієнт-сервер є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних програм і передбачає взаємодію та обмін даними між ними. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Сервери є незалежними один від одного. Клієнти також функціонують паралельно і незалежно один від одного. Немає жорсткої прив'язки клієнтів до серверів. Більш ніж типовою є ситуація, коли один сервер одночасно обробляє запити від різних клієнтів; з іншого боку, клієнт може звертатися то до одного сервера, то до іншого. Клієнти мають знати про доступні сервери, але можуть не мати жодного уявлення про існування інших клієнтів.

Дуже важливо ясно уявляти, хто або що розглядається як «клієнт». Можна говорити про клієнтський комп'ютер, з якого відбувається звернення до інших комп'ютерів.

Можна говорити про клієнтське та серверне програмне забезпечення. Нарешті, можна говорити про людей, які бажають за допомогою відповідного

програмного та апаратного забезпечення отримати доступ до тієї чи іншої інформації.

Загальноприйнятим є положення, що клієнти та сервери – це перш за все програмні модулі. Найчастіше вони знаходяться на різних комп'ютерах, але бувають ситуації, коли обидві програми – і клієнтська, і серверна, фізично розміщуються на одній машині; в такій ситуації сервер часто називається локальним.

Модель клієнт-серверної взаємодії визначається перш за все розподілом обов'язків між клієнтом та сервером. Логічно можна відокремити три рівні операцій:

– рівень представлення даних, який по суті являє собою інтерфейс користувача і відповідає за представлення даних користувачеві і введення від нього керуючих команд;

– прикладний рівень, який реалізує основну логіку ПЗ і на якому здійснюється необхідна обробка інформації;

– рівень управління даними, який забезпечує зберігання даних та доступ до них.

Дворівнева клієнт-серверна архітектура передбачає взаємодію двох програмних модулів – клієнтського та серверного. В залежності від того, як між ними розподіляються наведені вище функції, розрізняють:

– модель тонкого клієнта, в рамках якої вся логіка ПЗ та управління даними зосереджена на сервері. Клієнтська програма забезпечує тільки функції рівня представлення;

– модель товстого клієнта, в якій сервер тільки керує даними, а обробка інформації та інтерфейс користувача зосереджені на стороні клієнта. Товстими клієнтами часто також називають пристрої з обмеженою потужністю: кишенькові комп'ютери, мобільні телефони та ін.

Типовим прикладом клієнт-серверної взаємодії є WWW. Існує величезна кількість веб-серверів, на яких розміщується та чи інша інформація. У

найпростішому випадку ця інформація являє собою набір веб-сторінок, які можуть зберігатися на сервері у вигляді файлів, розмічених за допомогою мови розмітки HTML. Але ситуація, як правило, є складнішою; значна частина веб-ресурсів на сучасному етапі є динамічними, тобто вони не існують в заздалегідь підготовленому вигляді, а створюються безпосередньо в процесі обробки запиту від користувача.

Для того, щоб людина, яка працює в Інтернеті, могла переглянути ту чи іншу сторінку, на її комп'ютері повинно бути встановлено відповідне програмне забезпечення. Програми для перегляду веб-сторінок називаються браузерами.

Але, крім браузерів, до серверів можуть звертатися і інші клієнти, а саме – автономні програми. Вони можуть передбачати взаємодію з людиною, а можуть працювати в цілком автоматичному режимі. Типовим класом таких програм є роботи, призначені для автоматичного перегляду веб-ресурсів. Зокрема, роботи є важливим елементом пошукових систем і використовуються ними для перегляду сторінок і збору інформації про них.

Для запиту до веб-сервера клієнтська програма повинна задати місцезнаходження комп'ютера, на якому розміщується серверна програма, назву потрібного документа і, можливо, інші дані, які специфікують запит. Мережа забезпечує знаходження сервера і передачу йому клієнтського запиту. Серверні програми обробляють цей запит, відповідь пересилається по мережі клієнтові.

Трирівнева клієнт-серверна архітектура, яка почала розвиватися з середини 90-х років, передбачає відділення прикладного рівня від управління даними. Відокремлюється окремий програмний рівень, на якому зосереджується прикладна логіка ПЗ. Програми проміжного рівня можуть функціонувати під управлінням спеціальних серверів ПЗ, але запуск таких програм може здійснюватися і під управлінням звичайного веб-сервера. Нарешті, управління даними здійснюється сервером даних.

Для роботи з системою користувач використовує стандартне програмне забезпечення – звичайний браузер. Це позбавляє його необхідності завантажувати

та інсталювати спеціальні програми (хоча інколи така необхідність все-таки виникає).

Але користувачеві слід надати в розпорядженні інтерфейс, який дозволяв би йому взаємодіяти з системою і формувати запити до неї. Форми, що визначають цей інтерфейс, розміщуються на веб-сторінках та завантажуються разом з ними.

Веб-оглядач формує запит та пересилає його до сервера, який здійснює обробку. При необхідності сервер викликає серверні програмні модулі, які забезпечують обробку запиту і в разі потреби звертаються до сервера даних. Сервер даних здійснює операції з даними, що зберігаються в системі та складають її інформаційну основу. Зокрема, він може здійснити вибірку з інформаційної бази відповідно до запиту та передати її модулю проміжного рівня для подальшої обробки. Дані, з якими працює сервер даних, найчастіше організовані як реляційна база даних.

Найчастіше веб-сервер і серверні модулі проміжного рівня розміщуються на одному комп'ютері, хоч і являють собою окремі і логічно незалежні програмні модулі.

На сучасному етапі для програмування модулів проміжного рівня використовується мова серверних сценаріїв PHP, а для управління даними – СУБД MySQL. Таким чином, зв'язку PHP-MySQL слід розглядати як стандартний інструмент для створення порівняно простих інтерактивних веб-сайтів та систем електронної комерції; близько 90% комерційних систем сьогодні створюється саме на цій основі. Водночас як засоби управління даними, так і middleware-засоби можуть бути найрізноманітнішими. Так, для створення серверних програм, крім PHP, широко застосовуються Java, Perl, Python, Delphi.

Взагалі, технології створення розподілених, зокрема веб-програм, стрімко розвиваються. Слід згадати про технології EJB (Enterprise Java Beans), CORBA, а також про .NET – порівняно нову ініціативу компанії Microsoft. Для зберігання

даних та їх передачі часто використовується так звана розширювана мова розмітки XML (Extensible Markup Language).

Хоча я реалізовував програму сам, було використано підходи Scrum для саморозвитку та пришвидшенню розробки, розглянемо цей метод. Scrum – підхід управління проектами для гнучкої розробки програмного забезпечення. Скрам чітко робить акцент на якісному контролі процесу розробки.

Підхід вперше описали Гіротака Такеучі та Ікуджіро Нонака в статті The New New Product Development Game (Гарвардський Діловий Огляд, січ–лют 1986). Вони відзначили, що проекти, над якими працюють невеликі, крос–функціональні команди, зазвичай систематично продукують кращі результати, і пояснили це, як «підхід регбі». У 1991 році ДеҐрейс та Шталь у книжці Злі проблеми, справедливі рішення послалися на цей підхід, як на Scrum (штовханина; сутичка навколо м'яча (у регбі)), спортивний термін, згаданий в статті Такеучі і Нонака. Кен Швабер на початку 1990–х використовував підхід який привів Scrum в його компанію.

Вперше метод Scrum було представлено на загальний огляд задокументованим, чітко сформульованим та описаним спільно Сазерлендом та Швабером на OOPSLA'96 в Остіні. Швабер та Сазерленд протягом наступних років працювали разом щоб обробити та описати весь їхній досвід та найкращі практичні зразки для індустрії в одне ціле, в ту методологію, що відома сьогодні як Scrum. Швабер об'єднав зусилля з Майком Бідлом в 2001, щоб детально описати метод в книжці Agile Software Development with SCRUM. Не зважаючи на те, що для Scrum нарікли долю управління проектами з розробки ПЗ, він може також використовуватися в роботі команд обслуговувань програмного забезпечення (software maintenance teams), або як підхід управління розробкою і супроводом програм: Scrum of Scrums.

Scrum – це кістяк процесу, який включає набір методів і попередньо визначених ролей. Головні дійові особи – ScrumMaster, той хто опікується процесами, веде їх і працює як керівник проекту, Власник Продукту, людина, що

представляє інтереси кінцевих користувачів та інших зацікавлених в продукті сторін, та Команду, яка включає розробників.

Протягом кожного спринту, 15–30 денного періоду (тривалість визначається командою), працівники створюють функціональний ріст програмного забезпечення.

Набір можливостей, які імплементуються кожного спринту, приходять з етапу, що має назву product backlog (документація запитів на виконання робіт), який має найвищу пріоритетність за рівнем вимог до роботи, що повинна бути виконана.

Запити на виконання робіт (backlog items), що визначені протягом наради з планування спринту (sprint planning meeting), переміщуються в етап спринту. Протягом цієї наради Власник Продукту інформує про завдання, які він хоче, аби були виконані.

Тоді Команда визначає, скільки з бажаного вони можуть зробити, щоб завершити необхідні частини протягом наступного спринту. Протягом спринту команда виконує визначений фіксований список завдань (т.з. backlog items). Впродовж цього періоду ніхто не має права змінювати перелік запитів на виконання робіт, що слід розуміти, як заморожування вимог (requirements) протягом спринту.

Product backlog – це документ, який має список вимог до функціональності, які упорядковані згідно зі ступенем важливості. Product backlog представляє список того, що повинно бути реалізовано. Елементи цього списку називається «історіями» (user story) або елементами backlog–у (backlog items). Product backlog відкритий для редагування усім учасникам Scrum–процесу.

Обов'язкові поля:

1. ID – унікальний ідентифікатор, порядковий номер, який використовується для ідентифікації історій у разі їх перейменування.

2. Назва (Name) – стислий опис історії. Він повинен бути однозначним, щоб і розробники і product owner могли зрозуміти, про що йдеться і відрізнити одну історію від іншої.

3. Важливість (Importance) – ступінь важливості даної історії на погляд product owner ’а. Зазвичай являє собою натуральне число, іноді для цієї цілі використовуються числа Фібоначчі. Чим більше значення, тим більше пріоритет.

4. Попередня оцінка (initial estimate) – початкова оцінка об'єму робіт, необхідного для реалізації історії порівняно з іншими історіями. Вимірюється у story point'ах. Приблизно відповідає числу «ідеальних людино–днів».

5. Як продемонструвати (how to demo) – стисле пояснення того, як завершена задача буде продемонстрована у кінці спринта. Дане поле може являти собою код автоматизованого приймального тесту.

Додаткові поля. Іноді, також, використовуються додаткові поля у product backlog, в основному для того, щоб допомогти product owner'у визначитися з його пріоритетами.

Категорія (track). Наприклад, «панель управління» чи «оптимізація». За допомогою цього поля product owner може легко вибрати усі пункти категорії «оптимізація» і задати їм низький пріоритет.

Компоненти (components) – указує, які компоненти (наприклад, база даних, сервер, клієнт) будуть зачеплені при реалізації історії. Дане поле складається з групи checkbox'ів, які відмічаються, якщо відповідні компоненти потребують змін.

Ініціатор запиту (requestor). Product owner може захотіти зберігати інформацію про усіх замовників, зацікавлених у даній задачі. Це потрібно для того, щоб тримати їх у курсі діла про хід виконання робіт.

ID у системі обліку помилок (bug tracking ID) – якщо ви використовуєте окрему систему обліку помилок, тоді у описі історії корисно зберігати посилання на всі дефекти, які до неї відносяться.

Sprint backlog – містить функціональність, обрану Product Owner із Product Backlog. Всі функції розбиті по задачах, кожна з яких оцінюється командою. Кожен день команда оцінює об'єм роботи, який необхідно провести для завершення задачі.

Burndown chart – показує, скільки вже виконано і скільки ще залишається зробити.

Планування спринта (Sprint Planning Meeting)

Проходить на початку нової ітерації Спринта:

– Із Product Backlog обираються задачі, зобов'язання по виконанню яких за спринт приймає на себе команда;

– На основі обраних задач створюється Sprint Backlog. Кожна задача оцінюється у ідеальних людино–годинах;

– Рішення задачі не повинно займати більше 12 годин або одного дня. При необхідності задача розбивається на підзадачі;

– Обговорюється та визначається, яким чином буде реалізовано цей об'єм робіт;

– Тривалість наради обмежена зверху 4–8 годинами в залежності від тривалості ітерації, досвіду команди тощо;

– (перша частина наради) Беруть участь Product Owner + Команда: обирають задачі із Product Backlog;

– (друга частина наради) Бере участь лише команда: обговорюють технічні деталі реалізації, наповнюють Sprint Backlog.

Щоденна нарада (Daily Scrum Meeting)

Відбувається кожен день протягом спринта. Є «пульсом» ходу спринта. Нараді властиві наступні обмеження:

– починається точно вчасно;

– всі можуть спостерігати, але говорять тільки обрані;

– триває не більш ніж 15 хвилин;

– проводиться в одному і тому ж місці протягом одного спринта.

Протягом наради кожен член команди відповідає на 3 запитання:

– Що зроблено з моменту попередньої щоденної наради?;

– Що буде зроблено з моменту поточної наради до наступної?;

– Які проблеми заважають досягненню цілей спринта? (Над рішенням цих проблем працює ScrumMaster. Зазвичай це рішення проходить за рамками щоденної наради і у складі осіб, що безпосередньо займаються даною перешкодою.)

Демонстрація (Sprint Review Meeting):

– Проходить у кінці ітерації (спринта).

– Команда демонструє внесок функціональності до продукту всім зацікавленим особам.

– Залучається максимальна кількість глядачів.

– Усі члени команди беруть участь у демонстрації (одна людина на демонстрацію або кожен показує, що зробив за спринт).

– Обмежена 4–ма годинами в залежності від тривалості ітерації і змін у продукті.

Ретроспектива (Sprint Retrospective):

– Члени команди висловлюють свою думку про минулий спринт.

– Відповідають на два основних запитання: Що було зроблено добре у минулому спринті?; Що потрібно покращити в наступному?.

– Виконують покращення процесу розробки (вирішують питання та фіксують вдалі рішення).

– Обмежена 1-3ма годинами.

Блок-схеми показують весь процес роботи системи з підсистемами та частково доказують правильність вибраних проектних рішень.

Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма

буде об’єктно-орієнтована що вимагає високого рівня декомпозиції задач на класи.

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

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підсистеми та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ.

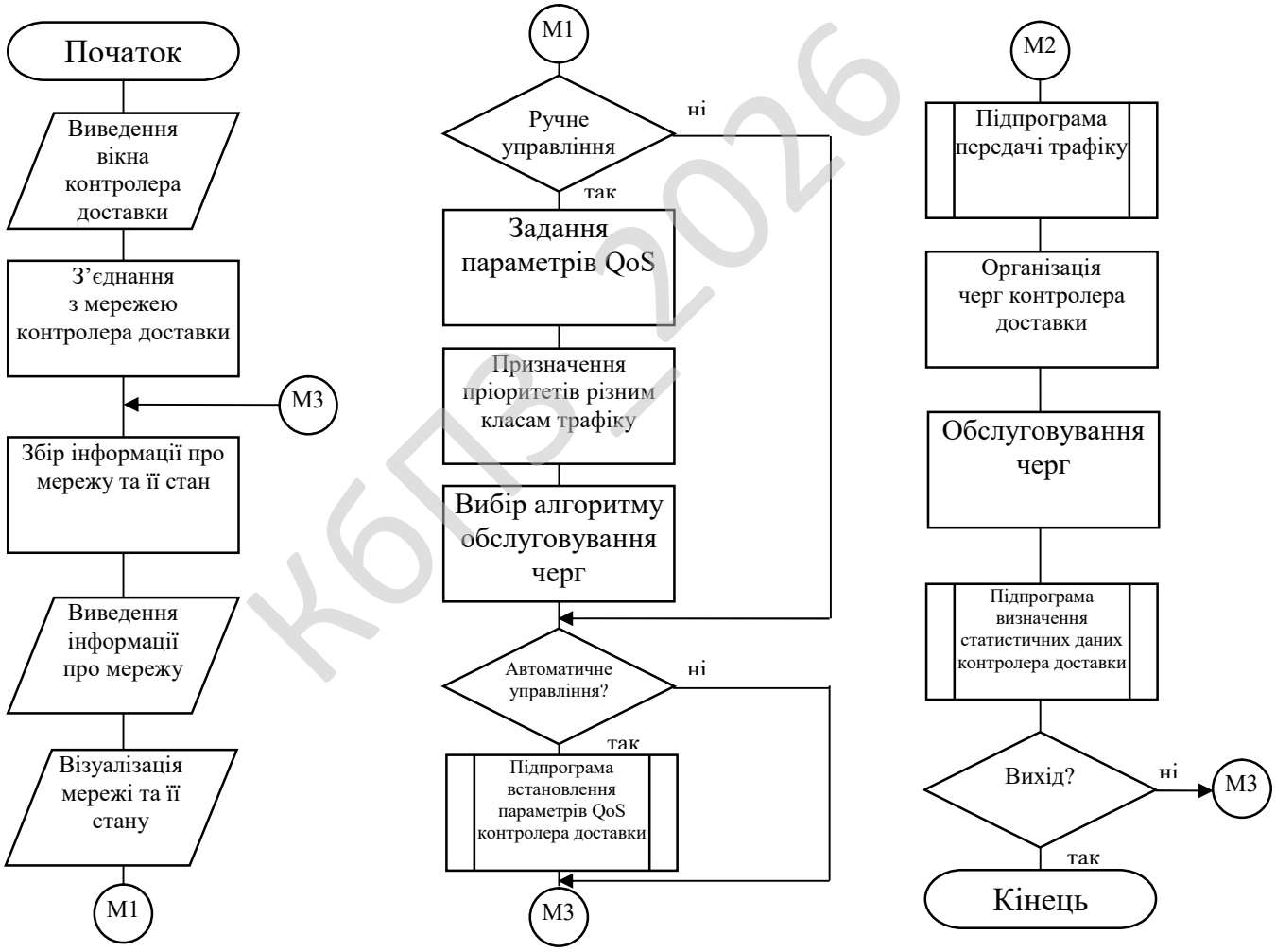

Рисунок 4.1 – Блок-схема основної програми

При роботі підпрограми виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

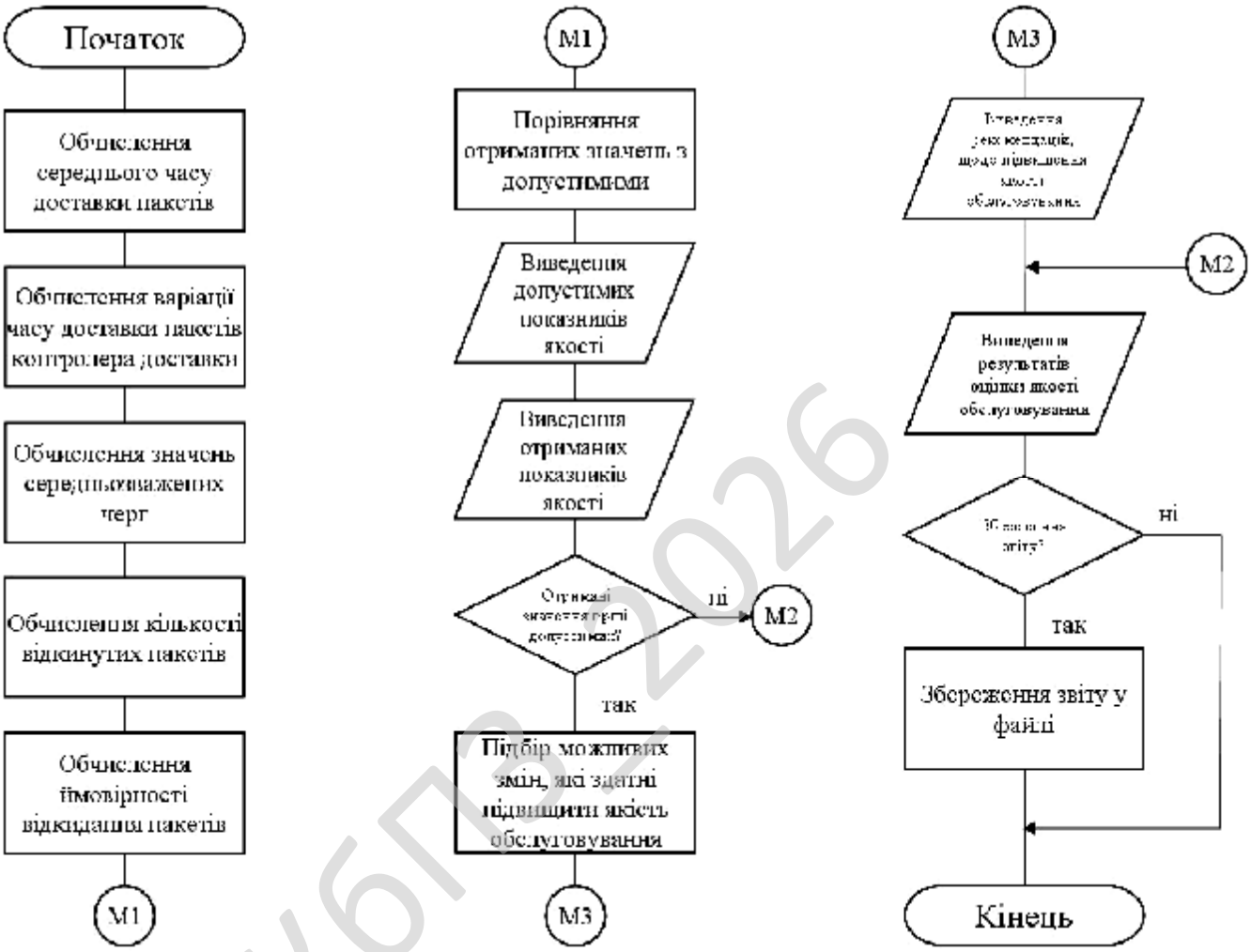

Рисунок 4.2 – Блок-схема роботи підпрограми

Було використано наступні підходи UML: Діаграма класів; Діаграма компонент.

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

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

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

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

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

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

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

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

якого є діапазон значень, або в явному вигляді. Вказуючи кратність на одному кінці асоціації, ви тим самим говорите, що на цьому кінці саме стільки об'єктів повинно відповідати кожному об'єкту на протилежному кінці.

Кратність можна задати рівною одиниці (1), можна вказати діапазон: "нуль або одиниця" (0..1), "багато" (0 .. *), "одиниця або більше" (1 .. *).

Дозволяється також вказувати певне число (наприклад, 3). За допомогою списку можна задати і більш складні кратності, наприклад 0. . 1, 3..4, 6 .. *, що означає "будь-яке число об'єктів, крім 2 і 5".

Агрегація це проста асоціація між двома класами відображає структурний відношення між рівноправними сутностями, коли обидва класу знаходяться на одному концептуальному рівні і ні один не є більш важливим, ніж інший.

Але іноді доводиться моделювати відношення типу «частина/ціле», в якому один з класів має більш високий ранг (ціле) і складається з декількох менших за рангом (частин).

Ставлення такого типу називають агрегацією; воно зараховане до відносин типу «має» (з урахуванням того, що об'єкт-ціле має кілька об'єктів-частин). Агрегація є окремим випадком асоціації і зображується у вигляді простої асоціації з незафарбованим ромбом з боку «цілого».

Графічно агрегація представляється порожнім ромбом на блоці класу, і лінією, яка від цього ромба до міститься класу.

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

Композиція має жорстку залежність часу існування екземплярів класу контейнера та примірників містяться класів.

Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно представляється як і агрегація, але з зафарбовані ромбиком.

Діаграма компонент в UML це діаграма, на якій відображаються компоненти, залежності та зв'язки між ними.

Діаграма компонент відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись.

Модуль програмного забезпечення може бути представлено в якості компоненти. Деякі компоненти існують під час компіляції, деякі – під час компонування, а деякі під час роботи програми.

Діаграма компонент відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Компоненти об'єднуються разом використовуючи структурні зв'язки (assembly connector) щоб об'єднати інтерфейси двох компонент. Це ілюструє зв'язок типу «клієнт-сервер».

Структурна взаємодія – «зв'язок двох компонент, який передбачає, що один з них надає послуги, потрібні іншому компоненту».

При використанні діаграми компонент щоб показати внутрішню структуру компонента, клієнтські та серверні інтерфейси можуть утворювати пряме з'єднання з внутрішніми. Таке з'єднання називається з'єднанням делегації.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова

функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних [2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено розроблене у бакалаврської дипломної роботі програмне забезпечення системи реалізації інтелектуального контролера доставки додатків ADC. З рисунку можна побачити що інтерфейс головного вікна розподілено на наступні функціональні розділи:

– Верхнього меню: Файл; Управління навантаженням; Моніторинг; Статистичні дані та рекомендації; Параметри; Довідка.

– Навігаційного меню яке викликається натисканням правої клавіші маніпулятора миші.

– Розділу виведення результату роботи системи.

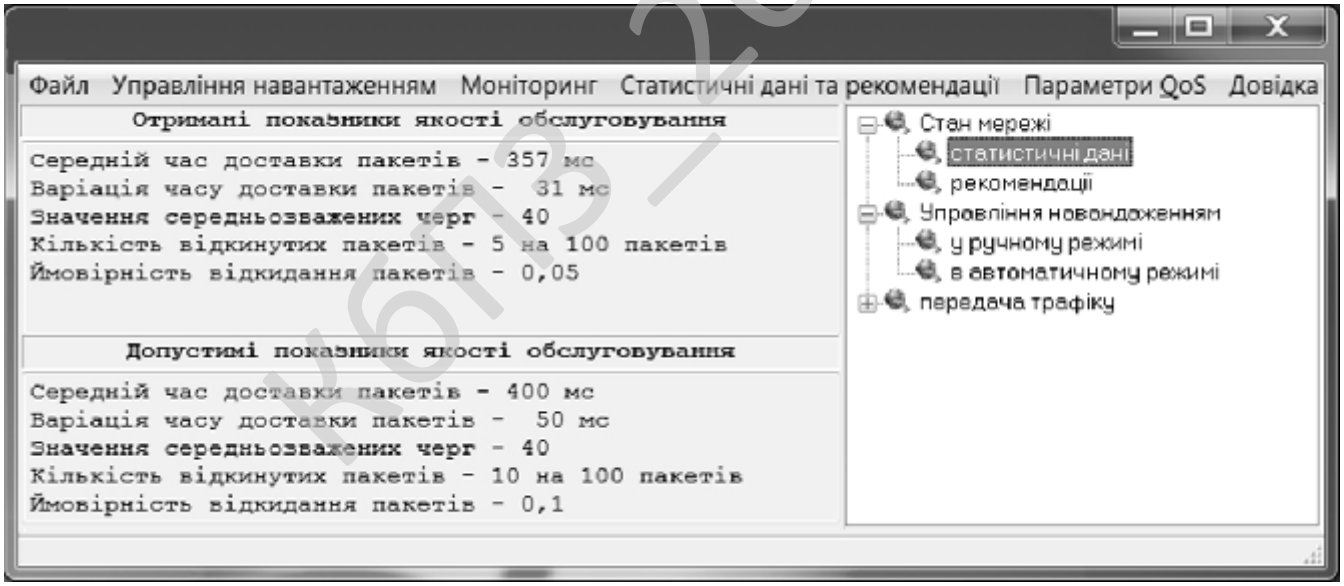

Рисунок 5.1 – Головне вікно ПЗ

Розроблена програма має дуже простий і зрозумілий інтерфейс з користувачем. Кожен, хто в достатньому обсязі володіє операційним середовищем Windows без особливих складностей освоїть і цю програму,

оскільки її інтерфейс інтуїтивно зрозумілий. Якщо програма не видала ніяких помилок, і працює, то можна використовувати, інакше слід слідувати інструкціям, які пропонує програма.

На рисунку 5.2 зображено авторські дані розробленого програмного забезпечення.

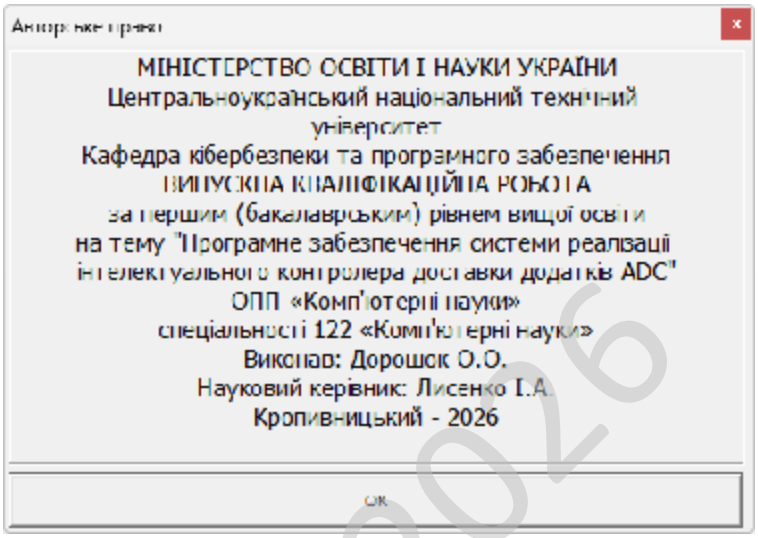

Рисунок 5.2 – Авторське право

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

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

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

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

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

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

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

Будь-який спосіб тестування «чорної скриньки» повинен:

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

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору OT.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

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

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

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – Shareware.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (не повнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання.

Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

Звичайно користувач платить тільки за час завантаження файлів через Інтернет або за носій (CD диск, флешку, ключ). Протягом певного терміну, що

становить зазвичай тридцять днів, він може користуватися програмою, тестувати її, освоювати її можливості.

Якщо після закінчення цього терміну користувач вирішить продовжити використання ПЗ, він зобов'язаний купити його (зареєструватися), заплативши авторові певну суму.

В іншому випадку користувач повинен припинити використання ПЗ та видалити його зі свого комп'ютера.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи реалізації інтелектуального контролера доставки додатків ADC.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем реалізації інтелектуального контролера доставки додатків ADC.

– Досліджена система реалізації інтелектуального контролера доставки додатків ADC.

– На основі отриманих результатів досліджень створена програмна реалізація системи реалізації інтелектуального контролера доставки додатків ADC.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання реалізації інтелектуального контролера доставки додатків ADC.

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

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

Програма реалізована на мові високого рівня Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи реалізації інтелектуального контролера доставки додатків ADC. 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 CRYPTON.

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

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

4. Вінтенко Б., Смірнов О., Миронець І., Смірнова Т., Смірнов С. «Імітаційна модель шляхів вхідних даних комп'ютерної інтелектуальної системи підтримки оператора енергоблоку АЕС». *Комбінаторні конфігурації та їхні застосування: Матеріали XXVII Міжнародного науково-практичного семінару, присвяченого 125-річчю Національного університету «Запорізька політехніка»* (Запоріжжя-Кропивницький-Київ, 4-6 червня 2025 р.). Запоріжжя: НУ «Запорізька політехніка», 2025. С.82-91.

5. Al-Azzeh, J., Ayyoub, B., Mesleh, A., Smirnova, T., Gnatyuk, S., Drieiev, O., Smirnov, O., Dorenskyi, O. «Cloud-Based Information System for Evaluating Caverns in the Process of Blasting Metal Surfaces of Details». *International Review on Modelling and Simulations* 18 (1), 2025. pp. 32-42.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

22. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

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

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

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

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

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

31. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», *2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019*/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

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

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

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

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

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

37. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у

комп'ютерних системах операторів стільникового зв'язку» *Вісник Черкаського державного технологічного університету. Технічні науки.* №4. С. 103-110. 2020.

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

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

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

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

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

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

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

45. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

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

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

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

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

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