НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО

Факультет інформатики та обчислювальної техніки
(назва факультету, інституту)

Кафедра автоматизованих систем обробки інформації і управління
(назва кафедри)

"На правах рукопису"
УДК *621.391*

«До захисту допущено»
В.о.завідувача кафедри

_____ О.А.Павлов
(підпис) (ініціали, прізвище)

“____” _____ 20*19* р.

МАГІСТЕРСЬКА ДИСЕРТАЦІЯ

на здобуття ступеня магістра

за спеціальністю *126 Інформаційні системи та технології*
(код та назва спеціальності)

ОПП *Інформаційні управляючі системи та технології*
(код та назва спеціалізації)

на тему: *Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі*

Виконав: студент *VI* курсу групи *ІС-82мп*
(шифр групи)

Труба Олександр Миколайович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник *доц., к.т.н., Коган А.В* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант *доц., к.т.н., доц., Жданова О.Г.* _____
(науковий ступінь, вчене звання, прізвище, ініціали) (підпис)

Рецензент *проф. каф. ОТ, д.т.н., проф. Кулаков Ю.О.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019

РЕФЕРАТ

Магістерська дисертація: 98с., 17 рис., 25 табл., 1 додаток, 37 джерел.

Актуальність. Останніми роками у всьому світі спостерігається надзвичайно стрімке зростання об’ємів даних, що передається та використовуються в мережі інтернет. Його спричиняють такі новітні тенденції в інформаційних та комунікаційних технологіях як постійне збільшення кількісті мобільних пристроїв та популярності соціальних мереж, накопичення великих об’ємів даних різного формату, що збираються з різних пристроїв, швидке накопичення об’ємів різноманітного мультимедійного контенту та стрімке використання хмарних технологій в повсякденному житті. Через це розмірність мереж та обсяги даних постійно збільшується і для того, щоб забезпечувати необхідну пропускну здатність та продуктивність для обробки, передачі і зберігання такої кількості даних, необхідно збільшувати інфраструктуру мереж. Але таке рішення призводить до значного ускладнення процесу налаштування та керування мережею, оскільки мережі, в більшості випадків, працюють з пристроями різного типу та різних виробників. Для того, щоб вони відповідали вимогам трафіку та змінам у мережі оператори мережі повинні налаштовувати їх вручну, витрачаючи дорогоцінний час. Тому традиційні підходи, які нині використовуються до архітектури мережі та керування нею швидке накопичення об’ємів різноманітного мультимедійного контенту та в найближчому майбутньому стануть неефективними. Виходячи з цього виникає необхідність пошуку та прийняття нових мережевих моделей, які дозоляться задовольнити потреби.

Програмно-конфігурована мережа (Software-Defined Networking, SDN) є одним з найбільш перспективних рішень для вирішення описаних проблем. Це підхід до архітектури мереж, який дозволяє значно спростити управління мережею. Він надає гнучкий та централізований контроль над нею, дозволяючи підвищити використання ресурсів мережі, зменшити операційні витрати та сприяє інноваціям. Це досягається завдяки тому, що рівень управління відокремлюється від мережевих пристроїв і

виноситься на окремий пристрій – контролер, який виступає в ролі доглядача над всією мережею.

Виходячи з того, що конструювання трафіку (Traffic Engineering, TE) є одним з головних методів для оптимізації роботи і підвищення надійності мережі, а існуючі технології в цій області не враховують унікальні особливості SDN і тому не є достатньо ефективними для них. Тому на сьогодні є актуальною задачею розробка нових технологій конструювання трафіку, які зможуть використовувати весь потенціал переваг програмно-конфігурованих-мереж.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась на кафедрі автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Ефективні методи розв’язання задач теорії розкладів» (№ ДР 0117U000919).

Мета дослідження – підвищення ефективності маршрутизації в програмно-конфігурованих мереж за рахунок вибору шляху, який відповідає заданим вимогам параметрів якості обслуговування в мережі.

Для досягнення мети необхідно виконати наступні **завдання**:

- провести огляд та аналіз існуючих способів маршрутизації та конструювання трафіку в програмно-конфігурованих мережах;
- розробити спосіб ТЕ з використанням багатошляхової маршрутизації для зменшення кількості втрат даних;
- розробити програмну реалізацію розробленого способу конструювання трафіку;
- дослідити ефективність розробленого способу та провести аналіз отриманих результатів.

Об’єкт дослідження – процес маршрутизації у програмно-конфігурованій мережі.

Предмет дослідження – методи вибору найкращого маршруту для передачі даних в програмно-конфігурованій мережі.

Методи дослідження – методи теорії графів, методи теорії оптимізації.

Наукова новизна полягає в наступному: запропоновано та обґрунтовано спосіб маршрутизації в мережі SDN, який відрізняється від існуючих меншою швидкістю на ремаршрутизацію трафіку та дозволяє зменшити відсоток втрати пакетів в програмно-конфігурованій мережі.

Публікації. Матеріали роботи опубліковані у тезах доповіді та подана до друку наукова стаття.

1. Труба О.М. Підвищення надійності конструювання трафіку в програмно-конфігурованій мережі / О.М. Труба, А.В. Коган // Матеріали ІІІ всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С. 39-42.
2. Ю.О. Кулаков. Конструювання трафіку в бездротових програмно-конфігурованих мережах / Ю.О. Кулаков, А.В. Коган, М.О. Сперкач, Труба О.М. // Східно-Європейський журнал передових технологій.

ПРОГРАМНО-КОНФІГУРОВАНА МЕРЕЖА, МАРШРУТИЗАЦІЯ, КОНСТРУЮВАННЯ ТРАФІКУ, ПЕРЕДАЧА ДАНИХ

ABSTRACT

Master dissertation: 98 pp., 17 fig., 25 tables, 1 app., 37 sources.

Actuality. In recent years, there has been an extremely rapid increase in the amount of data transmitted and used on the Internet worldwide. It is driven by the latest trends in information and communication technologies, such as the constant increase in the number of mobile devices and the popularity of social networks, the accumulation of large amounts of data of different formats collected from different devices, the rapid accumulation of various multimedia content and the rapid use of cloud technologies in everyday life. Due to this, the dimension of networks and data volumes is constantly increasing and in order to provide the necessary bandwidth and productivity for processing, transmitting and storing such amount of data, it is necessary to increase the network infrastructure. But this solution makes the network setup and management process much more complicated, since networks, in most cases, work with devices of different types and from different manufacturers. In order for them to meet traffic requirements and changes in the network, network operators must manually configure them by wasting valuable time. Therefore, the traditional approaches that are currently being used to manage and manage the network quickly accumulate a variety of multimedia content and will become ineffective in the near future. On this basis, there is a need to find and adopt new network models that will meet the needs.

Software-Defined Networking (SDN) is one of the most promising solutions to these problems. This is an approach to network architecture that greatly simplifies network management. It provides flexible and centralized control over it, increasing network utilization, reducing operating costs, and fostering innovation. This is achieved by the fact that the control layer is separated from the network devices and brought to a separate device - the controller, which acts as a caretaker over the entire network.

Considering that traffic engineering (TE) is one of the main methods for optimizing work and improving network reliability, existing technologies in this area do not take into account the unique features of SDN and therefore are not effective enough for them. Therefore, today it is an urgent task to develop new traffic engineering technologies that can use the full potential of the benefits of software-configured networks.

Relationship of work with academic programs, plans, themes. The work performed in accordance with the plan of the department of automated data processing systems and management of the National Technical University of Ukraine "Igor Sikorsky Kyiv Politechnic Institute" within the research theme «Efective Methods for Solving the Problems of the Theory of Schedules» (No. DR 0117U000919).

The purpose of the study is to improve the reliability of traffic design in software-configured networks by modifying the way traffic is constructed on the network.

To achieve the goal must perform the following **tasks**:

- review and analyze existing methods of routing and traffic design in software-configured networks;
- develop a method of TE using multi-path routing to reduce the amount of data loss;
- develop software implementation of the developed method of traffic design;
- investigate the effectiveness of the developed method and analyze the results obtained.

The object of the research – process of marching traffic on a software-configured network.

The subject of the research – methods of choosing the best path for data transmission on a software-configured network.

The research methods – graph theory, optimization theory.

The scientific novelty is the following: A method of routing on the SDN network is proposed and justified, which differs from the existing lower speeds on traffic re-routing and allows to reduce the percentage of packet loss in a software-configured network.

Publications. The materials of the work are published in the abstract of the report and submitted for publishing scientific articles.

1. O.M. Truba. Improving the reliability of traffic construction in a software-configured network / OM Pipe, A.V. Kogan // Proceedings of the Third All-Ukrainian Scientific and Practical Conference of Young Scientists and Students "Information Systems

and Technologies of Management" (ISTU-2019) - Kyiv.: NTUU "KPI them. Igor Sikorsky, November 20-22, 2019 - P. 39-42.

2. Y.O. Kulakov. Designing traffic in wireless software-configured networks / Y.O. Kulakov, A.V. Kogan, M.O. Sperkach, O.M. Truba // Eastern European Journal of Advanced Technology.

SOFTWARE-CONFIGURATED NETWORK, ROUTE, TRAFFIC ENGINEERING, DATA TRANSFER.

ЗМІСТ

ВСТУП

У зв'язку з постійним розвитком інформаційних технологій і появою нових технологій (таких як хмарні обчислення та Big Data), вимоги до комп'ютерних мереж зростають, а реалізація стає все складнішою. Це призводить до того, що сучасні мережі потребують більшої швидкості передачі даних та покращення інструментів, які використовуються для моніторингу мережі та управління нею. Така ситуація призводить до появи нових функціональних і технологічних мереж, з ускладненою інфраструктурою. Старі методи моніторингу та управління не відповідають новим вимогам та потребам.

З розвитком технологій зростає популярність програмно-конфігурованих мережі SDN (Software-Defined Networks), яка є ефективним способом забезпечення заданих параметрів якості обслуговування та перспективним рішенням для організації ефективної маршрутизації в умовах невизначеності.

Це нова мережева парадигма, що надає можливість спростити процес управління мережею, значно підвищити використання ресурсів мережі та зменшити операційні витрати. Однією з основних переваг такої мережі є управління на верхніх рівнях еталонної моделі, що дозволяє спростити як процес управління мережею, так і процес управління трафіком в корпоративних мережах і мережах центрів обробки даних.

Ідея програмно-конфігурованих мереж існує вже більше 10 років, але із швидким розвитком інформаційних технологій та збільшення потреб компанії починають все активніше пропонувати нові реалізації, які відкривають більше можливостей та можуть задовольнити всі вимоги. Однією з найпопулярніших є організація мережі SDN зі спільним застосуванням протоколу OpenFlow. Ключова перевага представленої технології над іншими полягає в тому, що вона працює окремо від мережевих пристроїв і її контроль може здійснюватися операторами за допомогою стандартного сервера.

Проте, як і кожна нова технологія, яка тільки починає розвиватися та потроху завойовувати ринок, SDN досі має чимало проблем, вирішення яких є актуальною

проблемою. Зараз однією з найбільш використовуваних технік конструювання трафіку у програмно-конфігурованих мережах є схема рівномірного розподілу, яка порівну ділить трафік між кількома доступними маршрутами – Equal Cost Multi-Path (ECMP). Але ні ECMP, ні інші популярні рішення зазвичай не враховують інформацію про стан мережі, наприклад завантаженість маршрутів. Іншим важливим фактором, що впливає на якість конструювання трафіку є тип трафіку, який проходить через мережу різного типу. Тому такі підходи не завжди виявляються придатними та ефективними в реальних інформаційних системах.

У зв’язку з цим актуальною задачею є розробка нових технологій конструювання трафіку та методів, що дозволять використовувати весь потенціал переваг програмно-конфігурованих мереж, для того щоб передавати дані ефективно, надійно і швидко, наскільки це можливо.

1 КОНЦЕПЦІЯ ТА ОСОБЛИВОСТІ ПОБУДОВИ SDN

1.1 Загальна концепція програмно-конфігурованих мереж і причини виникнення

Головна мета SDN є відокремлення рівня додатків від рівня управління і рівня управління від рівня передачі даних. Таким чином, можна значно спростити складність фізичних пристроїв, оскільки виконування логічних функції повністю переноситься на вищий рівень. Це не тільки здешевлює фізичні пристрої, а й покращує надійність і спрощує управління мережі в цілому. Тепер, замість маршрутизаторів можна використовувати звичайні комутатори. Для адміністратора мережі буде значно легше контролювати мережу в цілому. Також, це дозволяє абстрагуватися від реалізації кожного конкретного пристрою, оскільки рівень управління зв'язується з рівнем даних через стандартний інтерфейс. А це, в свою чергу, значно спрощує взаємодію між пристроями різних виробників, і зменшує час налаштування і підготовки або ремонту всієї мережі [1].

Окрім того, така побудова мережі значно прискорює створення нових мережевих додатків. Оскільки для взаємодії рівнів додатків і управління також використовується стандартний інтерфейс, програмісту більше не потрібно думати про те, яким саме чином можна передавати команди і запити до мережі. Головне реалізувати основний функціонал додатку, а додаток буде взаємодіяти з рівнем контролю через попередньо виділені інтерфейси [1]. Архітектура SDN схематично зображена на рисунку 1.1. Для більш чітко розуміння основних відмінностей традиційної мережі та SDN в таблиці 1.1 наведено порівняльний аналіз.

На рисунку 1.2 представлена конструктивна різниця в побудові та роботі традиційної архітектури і архітектури SDN.

У мережі SDN додаток отримує представлення про мережу, на відміну від традиційних мереж, де мережа отримує представлення про додаток. Традиційні (тобто не SDN) додатки лише неявно та опосередковано описують свої вимоги до мережі, зазвичай включають декілька кроків, що обробляються людьми, наприклад,

перевірка наявності ресурсів та керування політикою конфіденційності для підтримки додатків [1].

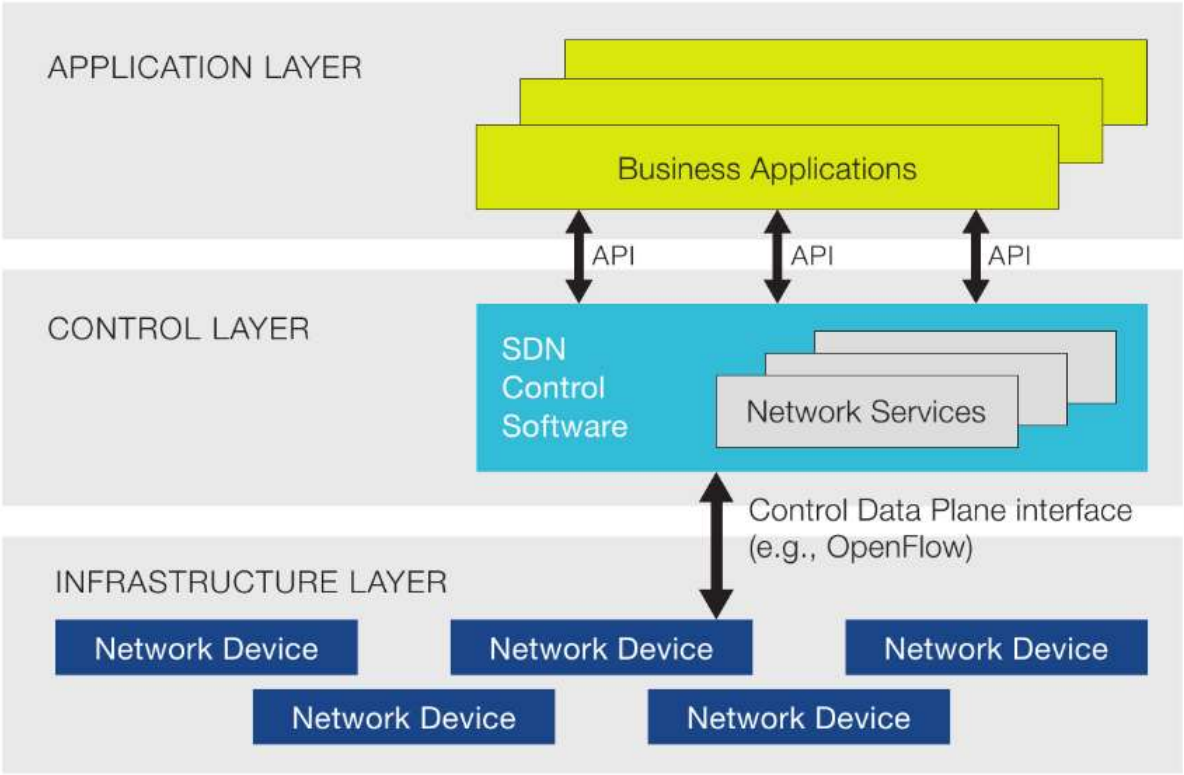


Рисунок 1.1 – Схематичне зображення архітектури SDN

Таблиця 1.1 – Порівняння підходу SDN з традиційними мережами

	Традиційні мережі	SDN
Мережевий вигляд	За рахунок апаратного забезпечення	За рахунок програмного забезпечення
Контроль за конфігурацією	Постачальник апаратного забезпечення	Користувач
Відкритість мережі	Закрита структура	Відкрита структура
Сумісність	Незалежні протоколи	Стандартизовані протоколи
Управління мережею	Управління на низькому рівні	Управління на логічному рівні
Адоптація нових технологій	Відповідно до потреб постачальника	Відповідно до потреб користувача

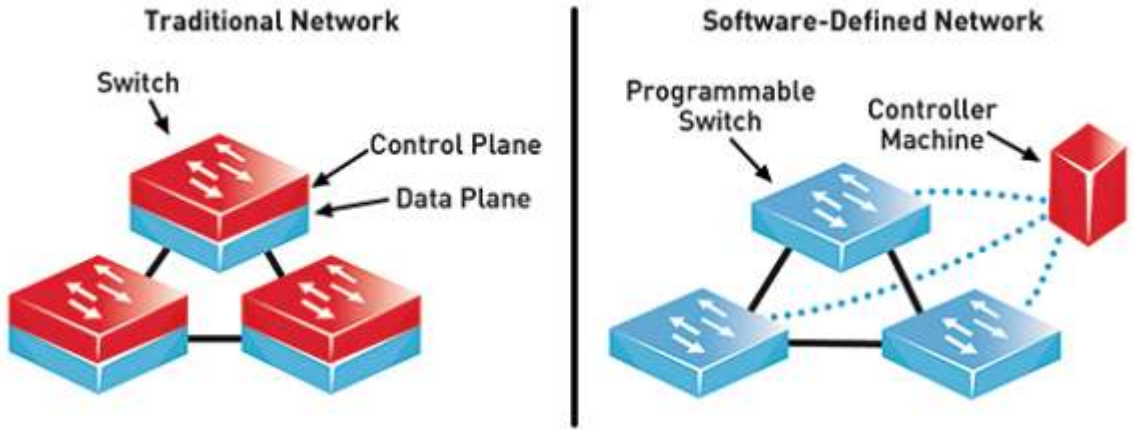


Рисунок 1.2 – Традиційна архітектура зліва і архітектура SDN справа

Традиційні мережі (наприклад, сучасний Інтернет та його послуги, такі як веб-переглядач, потокове передавання мультимедіа) не пропонують динамічний спосіб виразити весь спектр вимог користувачів, наприклад пропускну спроможність, затримку, варіацію затримки або доступність. Пакетні заголовки можуть зашифрувати запити пріоритету, але постачальники мережі зазвичай не довіряють маркуванням користувацького трафіку. Тому деякі мережі намагаються самостійно вивести вимоги користувачів (наприклад, через аналіз трафіку), що може спричинити додаткові витрати, а іноді й призвести до неправильної класифікації. SDN надає користувачеві можливість повністю вказувати свої потреби в контексті довірених відносин. традиційні (тобто не SDN) мережі не надають інформацію та стан мережі додаткам, що їх використовують. За допомогою підходу SDN додаток може відслідковувати стан мережі та адаптувати його відповідно. Рівень управління логічно централізований і відокремлений від рівня даних. Контролер SDN підсумовує інформацію про стан мережі для додатків і переводить вимоги додатків до низького рівня. Це не означає, що контролер фізично є централізованим. З точки зору продуктивності, масштабованості та/або надійності логічно централізований контролер SDN може бути розподілений таким чином, щоб декілька екземплярів фізичного контролера співпрацювали для керування мережею та обслуговування додатків. Контрольні рішення приймаються на глобальному перегляді стан мережі, а не розподіляються по ізольованій поведінці в кожній підмережі. За допомогою SDN

рівень керування виступає в ролі єдиної логічно централізованої мережевої операційної системи з точки зору як планування, так і вирішення конфліктів ресурсів, а також абстрагування від деталей пристроїв низького рівня, наприклад, від електричної або оптичної передачі [1].

Контролер SDN має повний контроль над всією мережею та рівнем даних. Виходячи з цього тому та обмежень, що накладаються на його можливості, він не повинен конкурувати з іншими елементами рівня контроля, що сприятливо вливає на планування та розподіл ресурсів в мережі. Це дозволяє мережам працювати з більш складними та більш точними політиками з використанням мережевих ресурсів та гарантією якості послуг. Це процес відбувається через загальну інформаційну модель (наприклад, як визначено в протоколі управління процесом обробки даних OpenFlow).

SDN дозволяє програмувати поведінку мережі в централізованому режимі через програмні додатки, що використовують відкриті API. Впровадження загального рівня управління SDN дозволяє керувати всією мережею та її пристроями послідовно, незалежно від складності базової мережевої технології. SDN дозволяє послідовно керувати мережею, яка складається з технологічно складних частин [1].

Відкриті стандарти: вільно та загальнодоступні специфікації для апаратного або програмного забезпечення, розроблені та підтримуються за допомогою спільних зусиль користувачів і постачальників обладнення.

Програмне забезпечення з відкритим кодом: вихідний код доступний для будь-кого для зміни або покращення, як OpenStack та OpenDaylight. API та SDK: API слугують інструментом для створення програмних додатків, часто визначаючи, яким чином компоненти програм повинні взаємодіяти один з одним. SDK - це пакети попередньо написаного коду, що мінімізує кількість повторної розробки коду. Деякі SDK публікуються API, доступні для всіх для написання у власних програмах [1].

Відкрите обладнання: проекти з відкритим посиланням на обчислювальні та мережеві продукти, такі як Open Compute Project Основоположними технологіями є

SDN контролери, vSwitches або API-програми. Вони дотримуються більш високих стандартів, ніж підтримуючі
технології, які дозволяють іншим технологіям взаємодіяти [1].

Основні переваги SDN над звичайними мережами полягають в наступному.

Можливість програмувати мережу. SDN дозволяє керувати поведінкою мережі програмним забезпеченням, яке знаходиться поза мережевими пристроями, що забезпечують фізичний зв'язок. Як наслідок, з'являється можливість налаштувати поведінку своїх мереж для підтримки нових послуг і навіть окремих клієнтів. Від'єднання апаратного забезпечення та програмного дозволяє запровадити інноваційні, диференційовані нові служби швидко, без обмежень закритих платформ [1].

Логічно централізований контроль. SDN побудований на мережевих топологіях, які дозволяють централізовано керувати мережевими ресурсами. Розподіляються традиційні методи керування мережею. Пристрої функціонують автономно. Завдяки такому централізованому управлінню забезпечується управління пропускною спроможністю, відновлення та безпека. Адміністратор в свою чергу отримує цілісний погляд на мережу [1].

Абстракція мережі. Сервіси та додатки, що працюють на технології SDN, абстрагуються від низькорівневих технологій і апаратного забезпечення. Програми взаємодіють з мережею через API замість інтерфейсів керування, які тісно пов'язані з апаратним забезпеченням [1].

Відкритість. Архітектура SDN відкриває нову еру відкритості, що забезпечує можливість взаємодії багатьох виробників, а також сприяє створенню нейтральної інфраструктури для взаємодії постачальників послуг.

Відкриті API підтримують широкий спектр додатків, включаючи хмарні послуги, OSS / BSS, SaaS і важливі мережеві додатки. Крім того, програмне забезпечення може контролювати обладнання від декількох постачальників з відкритими програмними інтерфейсами, такими як OpenFlow. Окрім того, сервіси мережі та програми можуть працювати в межах спільного програмного середовища [1].

Ключовою перевагою технології SDN є здатність операторів мережістворювати програми, які використовують API SDN, і надає додаткам можливість контролювати поведінку мережі. SDN дозволяє користувачам розробляти мережеві програми, спостерігати за вимогами мережі та автоматично адаптувати конфігурацію мережі, якщо необхідно. Програмне забезпечення, визначене SDN, є новим підходом до хмарних обчислень, що полегшує управління мережею та забезпечує програмно ефективну конфігурацію мережі для поліпшення продуктивності мережі та моніторингу. Мережі SDN призначені для вирішення того, що статична архітектура традиційних мереж децентралізована та складна, тоді як існуючі мережі вимагають більшої гнучкості та більше легкого вирішення проблем. SDN вміщає рівень управління мережею в одному компоненті, розділивши процес перенаправлення мережевих пакетів і процесу маршрутизації. Рівень управління складається з одного або більше контролерів, які розглядаються як мозок мережі SDN, де інтегрована вся логіка. Проте централізація має свої недоліки щодо безпеки, масштабованості та еластичності, і це є головною проблемою SDN [2].

Мережі SDN зазвичай пов'язують з протоколом OpenFlow (для віддаленого зв'язку з елементами мережевого рівня з метою визначення шляху мережевих пакетів через мережеві комутатори) з моменту появи останньої в 2011 році. З 2012 року багато компаній відійшли від OpenFlow і прийняли різні методи. До них належать Cisco Systems Open Network Environment та мережа віртуалізації мережі Nicira. Схема використання протоколу OpenFlow зображена на рисунку 1.3 [2].

SDN - це динамічна, керована, економічно ефективна та адаптована архітектура, придатна для високошвидкісної, динамічної природи сучасних програм. Архітектура SDN відокремлює функції керування та направлення трафіку в мережі, що дозволяє керувати базовою інфраструктурою, яка повинна абстрагуватися від додатків та сервісів мережі [2].

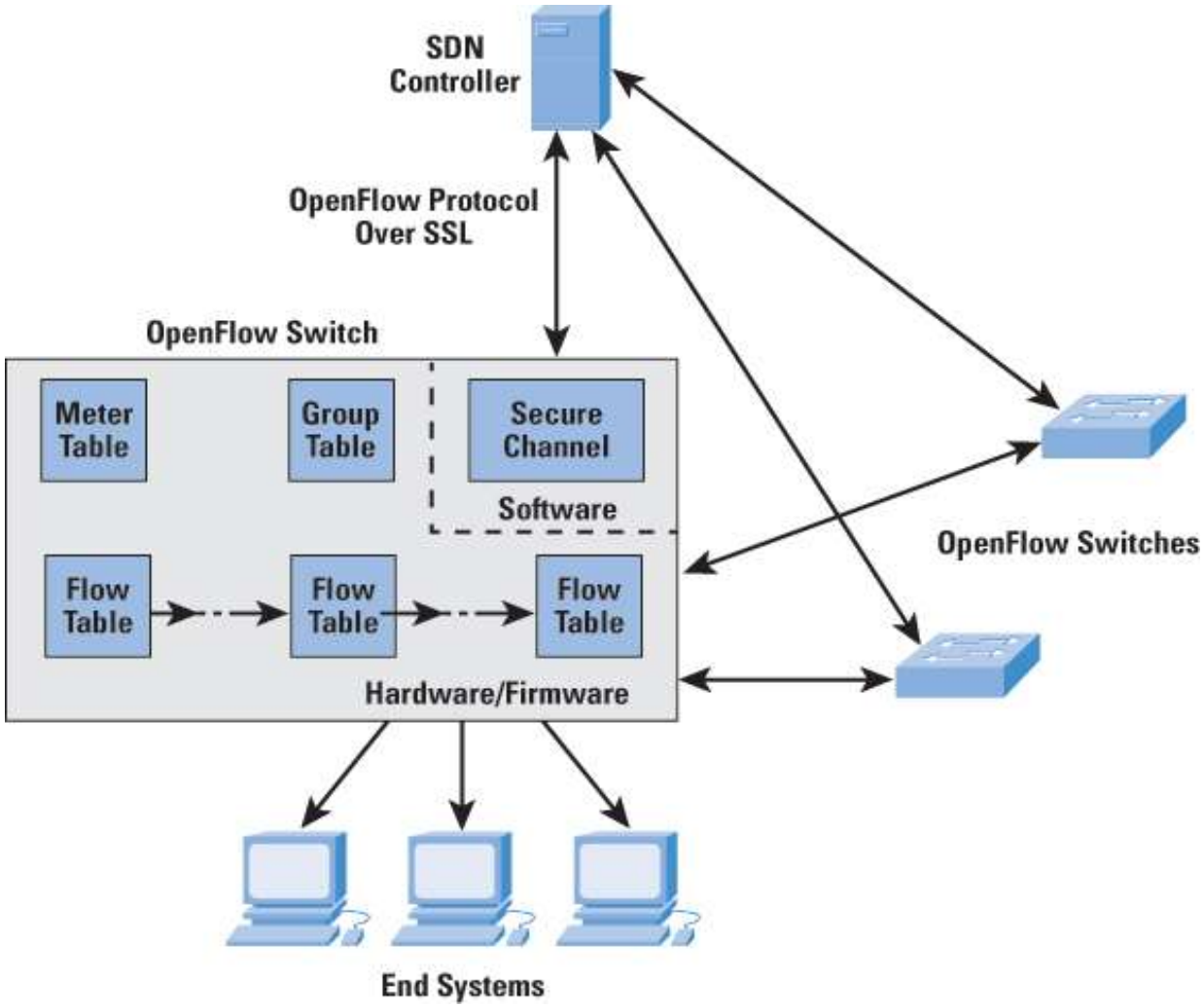


Рисунок 1.3 – Використання протоколу OpenFlow в мережі SDN

Інфраструктурний рівень складається з мережевих елементів, які виявляють свої можливості відносно рівня контроля. Додатки SDN існують на прикладному рівні та передають свої вимоги до мережі до рівня управління через північний інтерфейс, що називають NBI. Рівень контролю SDN передає вимоги додатків на низький рівень і контролює мережеві елементи. SDN може поєднувати конкуруючі вимоги додатків для обмежених мережевих ресурсів відповідно до політики конфіденційності [2].

Основною проблемою з базовою мережею зв'язку є динамічний характер мережевих додатків та їх середовища. Це означає, що вимоги до ефективності переданих потоків даних, такі як якість обслуговування (Quality of service (QoS)), можуть змінюватися з часом. Програми працюють у широкому діапазоні середовищ, тобто у дротових і бездротових мережах з безліччю мережевих пристроїв. Для того,

щоб програми працювали ефективно, основна мережа повинна бути достатньо гнучкою, щоб динамічно змінюватися у відповідь на будь-які зміни вимог додатків та їхнього середовища. Поточні докази або засновані на статичних або перевантажених мережах накладання, або вимагають, щоб програми змінювалися відповідно до продуктивності мережі [3].

Програмне забезпечення, що визначається мережею, виконує функції, що пов'язані з передачею даних та обробкою трафіку, таких як QoS, фільтрація, моніторинг або натискання. Трафік може віходити або виходити з рівня даних SDN через фізичні або логічні порти, і може бути спрямований до або з функцій пересилання та обробки. Обробка трафіку може бути пояснена на прикладі ядра OAM, функції шифрування або функції віртуалізованої мережі. Контроль за переадресацією або обробкою трафіку може виконуватися контролером SDN або окремими механізмами, які можуть бути організовані разом з даним контролером SDN. Схематичне порівняння рівнів управління і даних можна побачити на рисунку 1.9 [3].

Рівень даних реалізує рішення по пересиланню трафіка, прийняті на рівні управління. Як правило, рівень даних не робить автономних рішень щодо організації потоків трафіку у мережі. Однак рівень управління може налаштувати рівень даних таким чином, щоб він автономно реагував на такі події, як мережеві збої або підтримку функцій доставки, наприклад, через LLDP, STP, BFD або ICMP [4].

Інтерфейс між рівнями даних та управління (D-CPI) включає такі функції: програмний контроль усіх функцій, що виявляються RDB; інформація про потенціальні можливості мережі; повідомлення про події всередині мережі.

Агент рівня даних - це сутність, яка виконує інструкції контролера SDN на рівні даних. Координатор площини даних - це сутність, яка розподіляє ресурси даних для різних агентів клієнта та встановлює політики для керування ними. Агенти та координатори створені для досягнення однієї мети на кожному рівні архітектури [4].

На найнижчому рівні рекурсії ресурси рівня даних є фізичними (включаючи, наприклад, комутатори). Проте, на більш високих рівнях абстракції ресурси ресурсу

даних не повинні бути фізичними. Як і на інших рівнях, архітектура SDN працює за абстрактною моделлю рівня даних. І якщо функції цієї моделі, правильно виконані, архітектура не має жодних відмінностей з цією моделлю [4].

Адміністратор вибирає, якими ресурсами повинен управляти даний контролер SDN. Ці ресурси представлені у вигляді набору віртуальних серверів (VNEs), пов'язаних між собою для формування підмереж. VNE може бути послідовною абстракцією як підмережа [4].

Архітектура не накладає жодних обмежень на технології, що використовує рівень даних. Контролер SDN може бути використаний для програмування рівня даних, що реалізований як у вже існуючих технологіях, таких як DWDM, OTN, Ethernet, IP тощо, так і в більш нових технологіях рівня даних, які ще розвиваються [4].

OpenFlow - це програмований мережевий протокол для середовища SDN, який використовується для зв'язку між комутаторами OpenFlow і контролерами (рис. 1.4). OpenFlow відокремлює програмування мережевих пристроїв від базового обладнання та пропонує стандартизований спосіб доставки централізованої, програмованої мережі, яка може швидко адаптуватися до мінливих вимог до мережі [5].

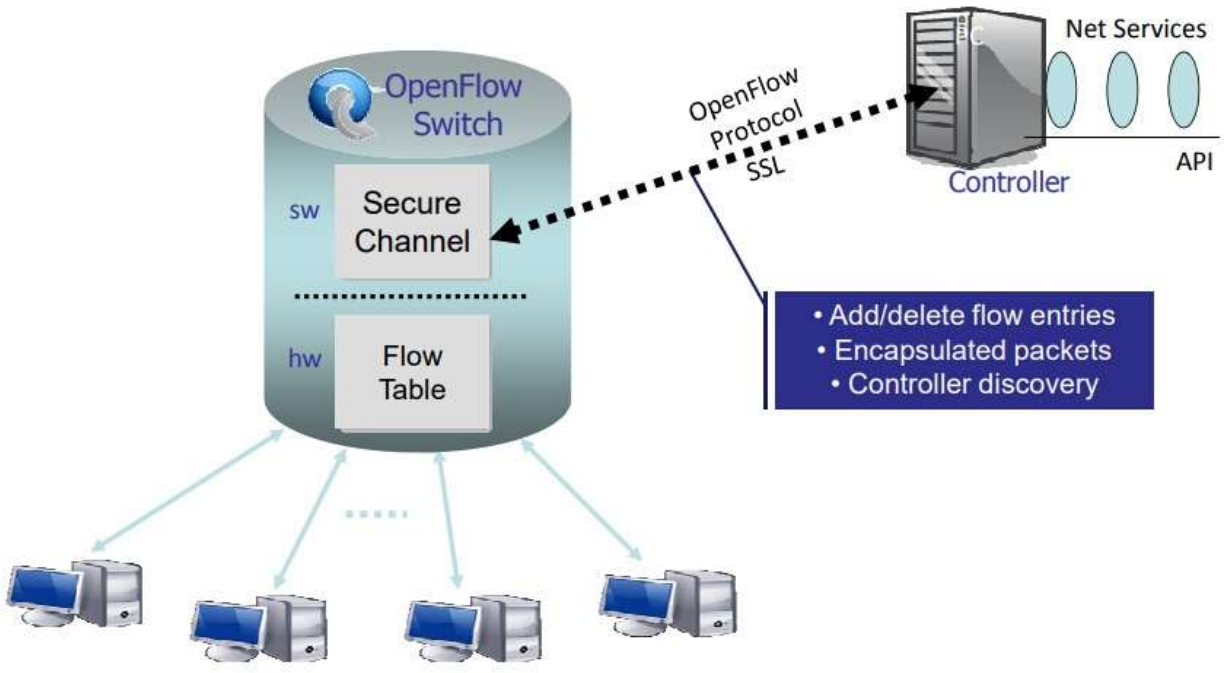


Рисунок 1.4 – OpenFlow перемикач в мережі SDN [5]

Перемикач OpenFlow - це перемикач даних з підтримкою OpenFlow, який

передається через канал OpenFlow до зовнішнього контролера. Він виконує пошук і пересилання пакетів відповідно до однієї або більше таблиць потоків і групової таблиці. Перемикач OpenFlow здійснює зв'язок з контролером, а контролер керує комутатором за допомогою протоколу комутатора OpenFlow. Вони або базуються на протоколі OpenFlow, або сумісні з ним [5].

OpenFlow (OF) вважається одним із перших стандартві SDN. Впочатку він визначив протокол зв'язку в середовищах SDN, який дає змогу рівню контроля SDN взаємодіяти з рівнем передачі даних, тобто з мережевими пристроями, такими як комутатори та маршрутизатори, як фізичними, так і віртуальними, щоб вони могли краще адаптуватися до змін вимог [5].

Контролер SDN в SDN - це "мозок" мережі SDN, що служить для передачі інформації між пристроями низьких рівнів (комутаторами/маршрутизаторами) та додатків та логічної складової, тобто високим рівнем. Останнім часом, оскільки організації розгортають більше мереж SDN, для контролерів SDN було поставлено завдання об'єднати між доменами контролера SDN, використовуючи загальні інтерфейси додатків, такі як OpenFlow та відкрити базу даних віртуальних комутаторів (OVSDB) [5].

Для роботи в середовищі OF будь-який пристрій, має підтримувати протокол OpenFlow, щоб встановити зв'язок з контролером SDN. Завдяки цьому інтерфейсу контролер SDN записує всі зміни до внутрішньої таблиці потоку комутаторів/маршрутизаторів, що в свою чергу дозволяє мережевим адміністраторам розподіляти трафік, керувати потоками для оптимальної продуктивності та починати тестування нових конфігурацій. Такий підхід є дуже зручним, оскільки в реальному часі можна слідкувати за змінами та швидко реагувати на незвичні ситуації в мережі.

OpenFlow дозволяє мережевим контролерам визначати шлях пакетів через мережу комутаторів [5]. Таке відокремлення управління від передачі даних дозволяє більш складне управління трафіком, ніж це можливо, використовуючи списки контролю доступу (ACL) та протоколи маршрутизації. Крім того, OpenFlow дозволяє використовувати комутатори від різних постачальників - часто з власними

інтерфейсами та мовами програмування - для керування віддалено за допомогою єдиного, відкритого протоколу. Винахідники протоколу вважають OpenFlow головним протоколом розвитку SDN [5].

OpenFlow дозволяє віддалене адміністрування таблиць передачі пакетів комутаторів третього рівня, додавання, редагування та видалення правил та дій, що пов'язані з передачею пакетів. Таким чином, рішення маршрутизації можуть бути здійснені контролером з використанням заздалегідь встановлених правил або прийнятих динамічно. Пакети, які не відповідають комутатору, можна передати контролеру. Контролер може потім вирішити змінити існуючі правила таблиці потоків для одного або декількох комутаторів або розгорнути нові правила, щоб запобігти структурному потоку трафіку між комутатором та контролером [5].

Протокол OpenFlow розташований на вищих рівнях порівняно з TCP і передбачає використання TLS. Контролери повинні слухати TCP-порт 6653 для комутаторів, які хочуть встановити з'єднання. Ранні версії протоколу OpenFlow неофіційно використовували порт 6633 [5].

Переваги OpenFlow:

- можливість керувати мережею програмно, дозволяє прискорено впроваджувати нові функції та послуги;
- абстракція; відокремлення обладнання та програмного забезпечення, рівня управління та пересилання даних, а також фізична і логічна конфігурація.
- централізований інтелект, спрощує підготовку мережі, оптимізує продуктивність, деталізує політику управління.

Одним з головних підходів для оптимізації роботи і підвищення надійності мережі – являється конструювання трафіку (Traffic Engineering (TE)). Це процес аналізу стану мережі, прогнозування і балансування переданих даних по мережних ресурсах. Це техніка, яка використовується для адаптації маршрутизації трафіку до змін у стані мережі. Мета інженерії трафіку полягає в підвищенні продуктивності мережі, QoS та користувальницького досвіду за рахунок ефективного використання ресурсів, що також може зменшити експлуатаційні витрати. Методи QoS

призначають наявні ресурси для трафіку з пріоритетом, щоб уникнути перевантаження для цього трафіку. Однак ці методи не надають додаткових ресурсів трафіку, який вимагає QoS. Традиційні методи маршрутизації не забезпечують жодного механізму для оптимального розподілу мережевих ресурсів [4].

Для вирішення цієї проблеми дослідницьке співтовариство почало працювати у сфері інженерії дорожнього руху та запропонувало нові шляхи підвищення надійності мережі у відповідь на зростання потреб у трафіку. Інженерний транспорт зменшує деградацію послуг через перевантаження та відмови. Відмовостійкість є важливою властивістю будь-якої мережі. Вона полягає в тому, щоб у випадку, якщо в мережі існує відмова, все одно запитані дані можуть бути доставлені до місця призначення. Комп'ютерні мережі складаються з численних мережевих пристроїв, таких як комутатори, середні коробки (наприклад, брандмауери) і маршрутизатори [4].

Існуючі методи конструювання трафіку, хоча і широко використовуються в мережах з традиційною архітектурою, але внаслідок неефективного використання доступних ресурсів, можливостей відстають від програмно-конфігурованих мереж. Ці мережі мають унікальні особливості, що дозволяють їм бути значно ефективнішими ніж звичайні мережі.

У традиційній моделі ЦОД (центр обробки даних) трафік генерується в основному між сервером і клієнтом, при цьому низька частка трафіку схід-захід між центрами обробки даних [6]. Завдяки широкому використанню Інтернету та інтеграції мобільної мережі, Internet of Vehicle [7], хмарні обчислення, великі дані та інші нові покоління мережевих технологій і розробок стали об'єктом масових даних і великих масштабів. масштабні розподілені центри обробки даних, що призводить до прискореного зростання трафіку Схід-Захід, що обмінюється між серверами, наприклад, файлової системи Google (GFS) [8], Hadoop Distributed File System (HDFS) [9] і каркаса Google MapReduce [10]. Однак, коли використовуються традиційно короткі протоколи маршрутизації в поточних центрах обробки даних, перевантаження часто відбувається в найкоротшій лінії зв'язку, і це може додатково знижувати якість мережевих послуг через тривалу затримку і низьку пропускну

здатність. Планування руху та контроль перевантаження є важливими технологіями для підтримки ємності мережі та підвищення ефективності мережі. Традиційні мережі мають деякі вроджені дефекти. Основні недоліки можна підсумувати наступним чином: по-перше, немає глобальної координаційної оптимізації. Кожен вузол незалежно реалізує стратегію управління трафіком, яка може досягти тільки локального оптимуму. Більше того, немає динамічної і само адаптивної корекції. Наперед визначені стратегії в маршрутизаторах не можуть відповідати часто мінливим вимогам бізнес-потоків. Крім того, традиційним мережам важко досягти ефективного і точного управління кожним мережевим пристроєм. Конфігурації мережевих пристроїв численні і різноманітні, коли команди настільки складні, що дуже важко знайти мережеві помилки, викликані конфігураціями. Отже, дуже важливо з'ясувати, як ефективно керувати мережевим трафіком і домінувати над ним, що спонукало мережевих архітекторів скористатися SDN для вирішення цих проблем у центрах обробки даних.[11]

У парадигмі SDN можна досягти більш високого рівня видимості і тонкого керування всією мережею. Контролер SDN може запрограмувати інфраструктурні пристрої пересилання в площині даних для моніторингу та управління мережевими пакетами, що проходять через ці пристрої дрібним способом. Тому ми можемо використовувати контролер SDN для реалізації періодичного збору цих статистичних даних. Крім того, ми також можемо отримати централізоване уявлення про стан мережі для SDN-додатків через відкриті API і повідомити про зміни в мережі реального часу . Центри обробки даних зазвичай складаються з тисяч високошвидкісних каналів, таких як 10 G Ethernet. У звичайних механізмах захоплення пакетів, таких як аналізатор портів комутатора і портів дзеркального відображення, нездійсненність повністю відштовхуються від вартості та можливості масштабування мережі, оскільки величезна кількість фізичних портів може бути використано. Прийняття підходів SDN на базі мереж центрів обробки даних стало центром сучасних досліджень для підвищення ефективності передачі трафіку в мережі [11].

Висновки до розділу

У результаті виконання першого розділу було вивчене та описане предметне середовище, розглянуті два основні типи мереж: традиційні та програмно-конфігуровані. Проведено порівняльний аналіз між ними для визначення переваг однієї на іншою. Також наведено детальну схему роботи програмно-конфігурованої мережі.

2 СПОСІБ КОНСТРУЮВАННЯ ТРАФІКУ У SDN МЕРЕЖІ НА ОСНОВІ БАГАТОШЛЯХОВОЇ МАРШРУТИЗАЦІЇ

2.1 Змістовна постановка задачі

Сучасні комп'ютерні мережі відрізняються великою розмірністю та різноманітним складом обладнання. У зв'язку з цим ускладнюється процес управління комп'ютерними мережами, зокрема, конструювання трафіку. Зі збільшенням розмірності комп'ютерних мереж і збільшенні обсягу мережевого трафіку актуальним є завдання конструювання трафіку з урахуванням його типу та параметрів якості обслуговування (QoS) і зниження енерговитрат.

Однією з основних переваг SDN є управління на рівні потоків, що дозволяє спростити як процес управління мережею, так і процес управління трафіком в корпоративних мережах і мережах центрів обробки даних.

Для скорочення часу TE і підвищення QoS необхідно організувати централізоване формування множини шляхів на основі багатошляхової маршрутизації в SDN.

Наявність множини маршрутів дозволяє практично виключити затримку або втрату пакетів в процесі ремаршрутизації трафіку. При цьому, чим більше шляхів буде сформовано в SDN контролері, тим імовірність затримки або втрати пакетів буде менше. Додатково, для більш точної маршрутизації трафіку, варто також враховувати завантаженість шляхів та тип трафіку, що передається в мережі.

Основною проблемою організації конструювання трафіку в програмно конфігурованій мережі, як і в традиційних мережах, являється не правильно визначена стратегія TE (вибору правильної альтернативи) на вузлі відправнику у відповідності з параметрами QoS. В процесі маршрутизації не уникнути ситуації перевантаження шляхів, виходу з ладу або переміщення вузла в мобільних мережах, для вирішення яких не завжди підходять традиційні способи.

Головною ціллю мережі зв’язку є передача даних (різного роду інформації) від одного вузла мережі до іншого по встановленим маршрутам, задовольняючи при

цьому необхідний рівень якості обслуговування користувачів. Для досягнення цієї цілі виконуються дві задачі:

- маршрутизація – визначення/побудова маршрутів, по яким будуть передаватись дані;
- конструювання трафіку – передача даних по існуючим маршрутам задовольняючи при цьому необхідний рівень якості обслуговування користувачів та уникаючи перевантаження маршрутів і утворення затор в мережі.

Методи конструювання трафіку намагаються здійснити передачу даних по мережі якомога ефективно, надійно і швидко. У програмно-конфігурованих мережах виконання цих методів виконує контролер – окремий пристрій у мережі, якій відповідає за рівень управління, налаштування пристроїв, які входять до мережі, та керування трафіком.

Отже, задачу конструювання трафіку можна визначити як: визначення способу розподілити трафік між кількома маршрутами, щоб забезпечити ефективну, швидку і надійну передачу даних; визначення стратегії поведінки передачі даних при виникненні перешкоджань: конфліктів або заторів, виходу з ладу вузла у маршруті, тощо.

2.2 Математична постановка задачі

Розглянемо модель реалістичної структури мережі SDN, у вигляді неорієнтованого графу, яка показує, що SDN-контролер відповідає за обробку запитів від багатьох комутаторів. Після ініціалізації мережі контролер SDN отримує найбільш оригінальні дані (топологія, пропускна здатність, затримка часу і т.д.). Удосконалений алгоритм маршрутизації працює на контролерах SDN. Коли надходять нові запити, контролери обчислюють оптимальний шлях для нових запитів і оновлюють ресурси смуги пропускання мережі (рис. 2.1).

Контролер на вхід отримує граф топології мережі представлений у вигляді (2.1).

$$G = (V, E, B, T), \qquad (2.1)$$

де $V = (v_1, v_2, v_3 \dots v_n)$, не порожня кінцева множина вершин;

$E = (e_1, e_2, e_3 \dots e_n)$, множина ребер;

$B = (b_1, b_2, b_3 \dots b_n)$, множина пропускної здатності ребер;

$T = (t_1, t_2, t_3 \dots t_n)$, множина затримок ребер.

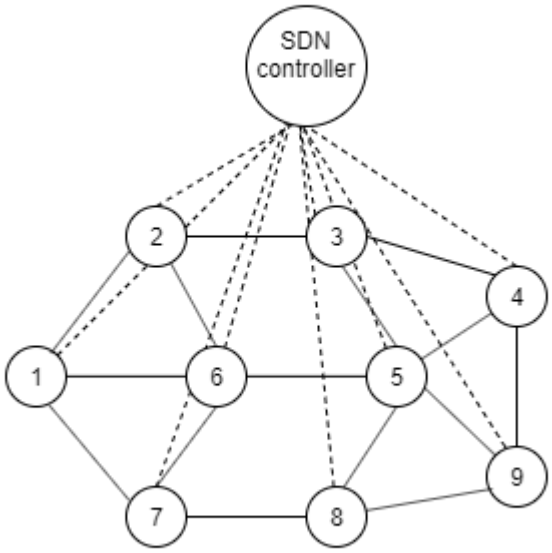


Рисунок 2.1 – Представлення мережі у вигляді неорієнтованого графу

Вважається, що для кожної пари вузлів S,T ∈ V маршрути у мережі вже знайдені і представлені у вигляді шляхів, що не перетинаються, наприклад: $Path = \{V_s, V_2, V_3 \dots V_t\}$.

Також відомі додаткові шляхи для обходу кожного з проміжних вузлів, надійність шляху (0<p<1), метрика шляху, вид трафіку, що передається.

В високонавантаженій мережі постійно відбуваються зміни стану з'єднань. Такі зміни важко передбачити, так як існує цілий ряд причин, що можуть послужити неочікуваному завантаженню лінії зв'язку, яка до цього була відносно вільною і допускала пересилання даних із збереженням високого рівня якості обслуговування.

Серед таки причин можна виділити:

– перебої в роботі маршрутизаторів;

– завантаження ключових каналів передачі іншими користувачами;
– збільшення кількості абонентів мережі.

У першому випадку, за певного збою у роботі одного з маршрутизаторів необхідна переадресація пакетів на інші канали. В такому разі, певні з'єднання можуть бути суттєво завантажені. В другому варіанті, можливі затримки можуть бути через часті одночасні з'єднання до одного з ресурсів великою кількістю користувачів. Такі перебої можна очікувати за низки соціальних факторів, як зростання популярності окремих вузлів, підвищена активність користувачів у певний час доби або різного роду хакерські атаки. Для мобільних мереж може зустрічатися варіант, коли після підключення додаткових абонентів з'являються проблеми з певними каналами передачі, але зазвичай в даних мережах такі можливі перебої передбачаються.

Приклади пріоритетних рішень:

– зв'язок чутливий до часу: підвищений пріоритет для сервісів IP-телефонії і передачі відео;
– не чутливий до часу зв'язок: знижений пріоритет для отримання веб-сторінки або відправлення листа по електронній пошті;
– висока важливість для організації: підвищений пріоритет для отримання інформації, що відноситься для управління виробництвом або торгівельних операцій;
– небажаний обмін даними: зниження пріоритету або блокування несанкціонованої активності, наприклад, обмін файлами між одноранговими.

Необхідно:

– навантажити рівномірно всі шляхи (розділити і направити трафік по всім доступним маршрутам);
– використовуючи онлайн метод провести маршрутизацію трафіку в мережі для забезпечення максимальної надійності мережі.

2.3 Огляд наявних рішень

Зараз найбільш популярним рішенням щодо багатошляхової маршрутизації в SDN є схема Equal-Cost-Multi-Path (ECMP) [12].

Багатоканальна маршрутизація з рівними витратами (ECMP) є стратегією маршрутизації, в якій пересилка пакетів наступного стрибка до одного пункту призначення може відбуватися через декілька "кращих шляхів", які пов'язують найвищі місця у розрахунках метрики маршрутизації. Багатоканальну маршрутизацію можна використовувати в поєднанні з більшістю протоколів маршрутизації, оскільки це рішення для кожного хопу, обмежене одним маршрутизатором. Він може істотно збільшити пропускну здатність трафіку з балансування навантаження по декількох контурах; однак, на практиці можуть виникнути серйозні проблеми.

Даний підхід має суттєвий недолік, а саме, не враховує завантаженість мережевих ресурсів та тип трафіку, що передається.

У роботі [13] запропонований підхід конструювання трафіку між декількома шляхами для SDN (MSDN-TE). Цей підхід базується на використанні декількох шляхів для передачі даних з урахуванням фактичного навантаження в процесі переадресації.

Основною метою алгоритму MSDN-TE є уникнення перевантажених мережі шляхом перерозподілу даних між маршрутами.

Однак у великих мережах кількість шляхів може залежати від топології мережі, а постійно перенаправляти дані між шляхами не є вирішенням задачі перевантаження мережі, так як може утворитися «вузьке місце», де пропускна здатність шляху буде критичною. Саме утворення таких «вузьких місць», як в MSDN-T, призводить до втрати більшості пакетів, що передаються.

Протокол Open Shortest Path First (OSPF) [14] приймає алгоритм відкритого найкоротшого шляху. Кожен маршрутизатор в мережі обчислює найкоротший шлях через алгоритм Дейкстри і записує результати в таблицю маршрутизації. Оскільки класичний алгоритм маршрутизації розглядає тільки вузли затримки, а ємність зв'язку

не розглядається. У даній роботі часові затримки та пропускна здатність, отримані контролерами SDN, використовуються для класичного алгоритму Дейкстри [15]. Топологія мережі дозволяє легко алгоритму дізнатися, чи з'єднані вузли чи ні. Після обчислення оптимального шляху ресурси каналу оновлюються синхронно [16]. Запропонований метод поєднує в реальному часі стан мережі і топологію мережі, що робить алгоритм більш адресним відповідно до реальної ситуації в мережі.

В роботі [17] запропонована технологія конструювання трафіку **Hedera**, точніше це система управління потоками у мережі, що спрямована на підвищення ефективності використання мережевих ресурсів (зокрема пропускної здатності). Зараз найбільш популярним рішенням щодо багатошляхової маршрутизації в SDN є схема Equal-Cost-Multi-Path (ECMP), вона є досить обмеженою і не враховує завантаженість мережевих ресурсів та розміри потоків. Ідея Hedera полягає в тому, що великі потоки (elephant flows) направляються на контролер, а інші потоки оброблюються згідно ECMP, це дозволяє уникати перенавантаження та колізій. Процес складається з трьох кроків, які виконує контролер: по-перше, виявлення великих потоків; по-друге, обрахування потреб виявлених потоків; по-третє, формування маршрутів, по яким треба перенаправити ці потоки. Всі кроки виконує контролер [17].

Виявлення великих потоків: будь-який потік спочатку вважається звичайним, він направляється по одному із своїх маршрутів згідно ECMP, якщо його швидкість передачі переходить порогову (зазвичай 100Mbps) то потік помічається як великий. Для виявлення великих потоків контролер (планувальник) збирає дані з кожного комутатора про кожен з його потоків кожні 5 секунд і потік вважається великим, якщо для нього виконується умова [17]:

$$V_f > 100 \text{ Mbps}, \qquad (2.2)$$

де V_f – швидкість потоку.

Щоб сформувати маршрути для перенаправлення великих потоків необхідно знати потреби у пропускній здатності цих потоків – це другий етап. По швидкості потоку не можна сказати про його потреби, тому автори роботи представили

алгоритми для обрахування потреб потоків, який ітеративно знаходить їх – «demand estimator» [18].

Вхідні дані для функції:

- F – множина пар джерело-призначення для всіх активних великих потоків;
- N – кількість хостів;
- M – матриця розмірністю NxN, Mij – містить три значення: кількість потоків з хоста i до хоста j, знайдені потреби потоків, мітка чи знайдено для потоків необхідна кількість ресурсів [17].

Вихідними даними функції є матриця зі знайденими потребами для кожного потоку. Наприклад у матриці M для чотирьох хостів (рис. 2.2) елемент M13 означає, що з 1 хоста на 3 хост направляється один потік, 1/3 означає, що всього з 3 хоста до інших хостів направляється 3 потоки, круглі дужки вказують що для цього потоку ще не знайдено його потреби [17].

$$\begin{bmatrix} & [\frac{1}{3}]_1 & (\frac{1}{3})_1 & [\frac{1}{3}]_1 \\ [\frac{1}{3}]_2 & & (\frac{1}{3})_1 & 0_0 \\ [\frac{1}{3}]_1 & 0_0 & & [\frac{2}{3}]_1 \\ 0_0 & [\frac{1}{3}]_2 & 0_0 & \end{bmatrix}$$

Рисунок 2.2 – Вихідна матриця M для функції «demand estimator» [17]

Для формування маршрутів, по яким треба перенаправити великі потоки пропонується два алгоритми Global First Fit та Simulated Annealing. Перший є жадібним, він лінійно перебирає всі можливі шляхи та обирає перший який може вмістити потік, другий базується на імовірнісному пошуку для ефективного обчислення шляхів для потоків і використовує результати функції «demand estimator» (рис. 2.3) [18].

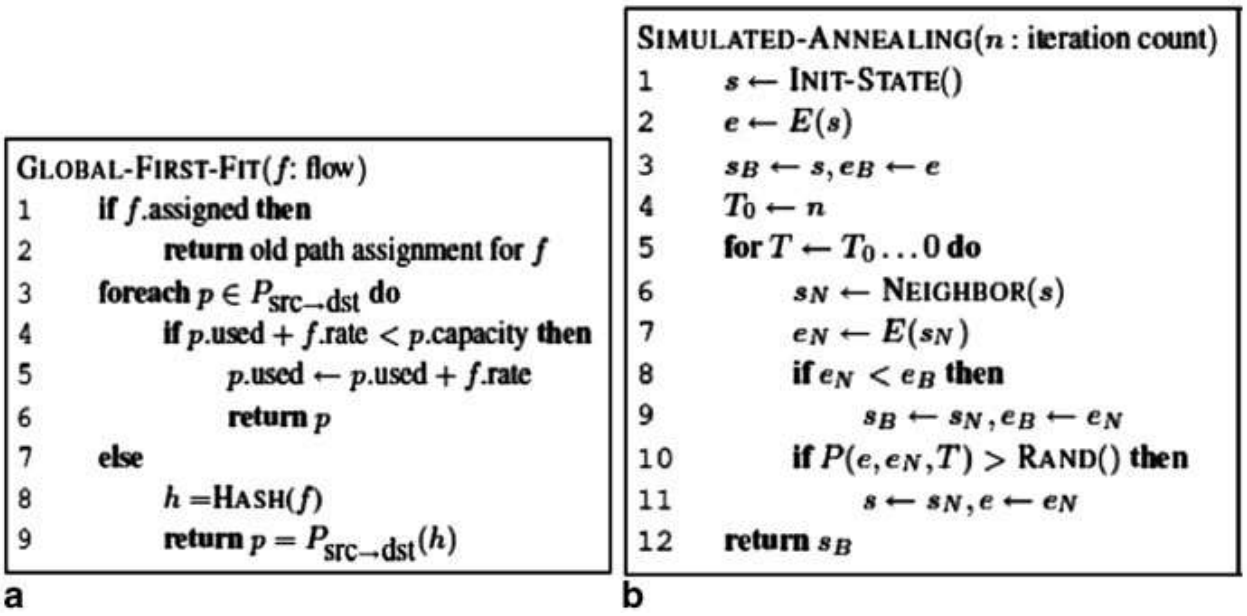


Рисунок 2.3 – Псевдокод алгоритмів Global First Fit та Simulated Annealing [17]

Технологія Hedera може значно поліпшити використання пропускної здатності мережі, але через те, що вона для виявлення великих потоків виконує постійний (кожні 5 секунд) збір даних з комутаторів, це призводить до суттєвого споживання ресурсів комутаторів та накладні витрати на моніторинг [17].

Схожим на Hedera, але трохи ефективнішим є механізм конструювання трафіку запропонований в роботі [18] який називається Mahout. Як і попередня технологія, він спрямований на виявлення та «особливе» управління великими потоками, тому що вони складають всього 10% усіх потоків, що проходять через мережу, але через те, що вони відповідають за передач 90% даних з'являються перевантаження ділянок мережі, якщо обробляти такі потоки так само як всі інші. Аналогійна до попереднього алгоритму потоки, які не відмічені як великі передаються згідно ECMP і лише великі потоки направляються на контролер [18].

Ключова відмінність полягає в тому, що виявлення великих потоків відбувається не за допомогою постійного збору інформації з комутаторів, а на граничних хостах і інформування контролеру про наявність великих потоків здійснюється не напряму. На кожному хості в операційній системі є shim layer, який слідкує за потоками що надходять на хост. Відбувається моніторинг сокетів буфера, коли його значення переходить порогове (експериментально виведене порогове

значення 100 КВ (або з інтервалу 200-500КВ), так як 85% потоків менші за це значення), то потік вважається великим і про це необхідно сповістити контролер [18].

Сповіщення відбувається за допомогою маркування пакетів цього потоку, а точніше у полі DS заголовка пакета вказується специфічне значення, і коли такий пакет надходить на комутатор, він одразу розпізнає цей потік як великий та направляє його на контролер [18].

Коли на контролер надходить пакет з великого потоку, то контролер повинен знайти для цього потоку кращий маршрут ніж з маршрутів ECMP, під кращим розуміється найменш навантажений серед усіх маршрутів між парою джерело-призначення, він знаходиться перебором [18].

OpenFlow є зручною і простою концепцією, але він вимагає багато надмірних витрат. Звісно він дозволяє спростити управляння мережею та управління трафіком, тому що він надає контроль на рівні потоків та глобальне бачення потоків, але для цього надто часто доводиться залучати комутатори та контролер для налаштування записів у таблицях та збору статистичних даних, це призводить до суттєвого споживання ресурсів. Для вирішення цієї проблеми було запропоноване рішення – DevoFlow [19].

За своєю суттю DevoFlow є модифікацією моделі OpenFlow, у якій використовується менша кількість витрат ресурсів, за допомогою знайденого компромісу співвідношення між централізованістю та витратами. Те, що взаємодія між пристроями та контролером виникає дуже часто, тобто виникає багато контрольного трафіку, призводить до того, що виникають суттєві затримки при налаштовуванні таблиць маршрутизації та несвоєчасність забезпечення статистичною інформацією для ефективного балансування навантаження. Тому автори роботи вказують на те, що повний контроль над усіма потоками не головна мета, ефективне управління потоками може бути досягнуто і тоді, коли більшість потоків керуються комутаторами, і лише великі потоки обробляє контролер. Це дозволяє зменшити кількість взаємодій між пристроями та контролером, а отже і

зменшити надмірні витрати н використання ресурсів мережі. Тобто головними принципами даного рішення є:

- передавати потоки без залучання контролера наскільки це можливо;
- підтримувати достатній рівень у контролера глобального бачення мережі, але надавати агреговану статистику [19].

Проблема надмірної взаємодії комутаторів з контролером вирішується за допомогою того, що більшість потоків передаються без залучання контролера, а проблема збору статистики вирішується за допомогою додавання ефективного механізму збору статистичних даних для визначення та маркування великих потоків. Отже ефективне балансування навантажень у мережі може бути досягнуто без обробки контролером кожного потоку [19].

Механізм управління потоками, який виконують пристрої, полягає у тому, що для малих потоків клонуються записи у таблиці комутатора і це дозволяє обрати по якому маршруту направити цей потік, на відміну від стандартного підходу, коли всі малі потоки, які відповідають запису у таблиці потоків, направляються по одному і тому самому маршруту. А також якщо обраний маршрут виходить з ладу, то якщо комутатор має один чи декілька обхідних маршрутів прописаних у таблиці маршрутизації, то він може майже миттєво зробити перенаправлення цього потоку, замість того, щоб чекати на відповідь від контролера [19].

Для покращення ефективності збору статистики по мережі пропонується три стратегії: перша полягає в тому, щоб надсилати дані контролеру лише про кожний 1000-й пакет (це значення можна регулювати); друга полягає у встановленні порогового значення для лічильника (пакетів, байтів, тривалості потоку), і після досягнення порогового значення контролеру надсилається повідомлення; третя полягає у надсиланні контролеру інформації про топ-k найбільших малих потоків.

Обробка потоків виконується за такою логікою: вхідні потоки передаються використовуючи багатошляхову маршрутизацію за допомогою схеми з використанням клонування записів. Потік визнається великим при досягненні порогового значення байтів 1-10 MB (рис.2.4). Потім направляється контролеру

повідомлення і він знаходить найменш навантажений маршрут для цього потоку алгоритмом best-fit bin packing [19] і перенаправляє потік вставляючи у таблицю запис з новим обрахованим маршрутом [19].

Algorithm 1 — Flow rate computation.

Input: set of flows F and a set of ports $\mathcal{P}$
Output: a rate $r(f)$ of each flow $f \in F$

begin
Initialize: $F_a = \emptyset$; $\forall f, r(f) = 0$
Define: $P.\text{used}() = \sum_{f \in F_a \cap P} r(f)$
Define: $P.\text{unassigned\_flows}() = P - (P \cap F_a)$
while $\mathcal{P} \neq \emptyset$ **do**
    Sort $\mathcal{P}$ in ascending order, where the sort key
        for P is $(P.\text{rate} - P.\text{used}())/|P.\text{unassigned\_flows}()|$
    $P = \mathcal{P}.\text{pop\_front}()$
    for each $f \in P.\text{unassigned\_flows}()$ **do**
        $r(f) = (P.\text{rate} - P.\text{used}())/|P.\text{unassigned\_flows}()|$
        $F_a = F_a \cup \{f\}$
end

Рисунок 2.4 – Псевдокод для визначення великого потоку DevoFlow

Загалом DevoFlow виконує у 10-53 разів менше записів у таблиці потоків та використовує у 10-42 разів менше контрольних повідомлень. Запропонований методі передачі потоків до виявлення великих досягає більш гнучкого балансування ніж схема ECMP. Дане рішення добре працює для різних топологій мереж і дозволяє ефективніше використовувати ресурси комутаторів [19].

MicroTE – система, яка адаптується до змін трафіку і використовує короткочасне прогнозування та часткову передбачуваність матриці трафіку для передачі даних по кільком маршрутам. Для ефективного управління завантаженням мережі, механізм використовує декілька шляхів та координує планування трафіку за допомогою глобального перегляду трафіку на доступних шляхах мережі. Як і в Mahout використовується виявлення великих потоків на граничних хостах. Для обробки всіх інших потоків використовується ECMP [20].

Сучасні техніки забезпечують лише 80-85% ефективності оптимального механізму маршрутизації через те, що вони не використовують багатошляхову маршрутизацію (або використовують недостатньо ефективну схему розподілу), не враховують зміни у площині трафіку, не використовують глобальне бачення

інформації про весь трафік у мережі для прийняття рішень конструювання трафіку. Ідеальний механізм конструювання трафіку повинен налаштовувати маршрути динамічно беручи до уваги глобальне бачення майбутньої матриці трафіку і обраховувати маршрути для майбутньої матриці трафіку одразу [20].

Так як більша частина трафіку у мережі є передбачуваною на короткому часовому проміжку (1,5 – 5 секунд), то це пропонується використати для досягнення кращого конструювання трафіку. Визначені три принципи яких потрібно дотримуватись для зменшення втрат даних та зменшення максимального використання каналів зв’язків: здійснення багатошляхової маршрутизації, координоване керування з використання глобального бачення картини про трафік та використання короткочасного прогнозування для адаптації до змін [20].

MicroTE логічно розмежовує передбачуваний та непередбачуваний трафік і приймає найбільш підходящі рішення щодо їх передачі кожен односекундний інтервал. Спочатку обраховуються маршрути для передбачуваного трафіку з урахуванням глобальної мети (наприклад мінімізація максимального використання каналів зв’язку, потім використовуючи схему ECMP непередбачуваний трафік розподіляється між пропускною здатність, що залишилася у маршрутів. Метод розроблений таким чином, що коли велика частина трафіку є передбачуваною, то для неї виконується оптимальна передача, інакше без проблем вона передається схемою рівномірного розподілу [20].

Архітектура MicroTE складається з трьох компонент: блок моніторингу, блок маршрутизації, мережевий контролер (рис. 2.5).

Блок моніторингу слідкує за вимогами трафіку та статистикою потоків між rack-серверами. Також тут відбувається виявлення трафіку, який є передбачуваний, для цього у матрицях трафіку знаходиться середнє значення для кожної пари rack-серверів . Перед тим як інформація надсилається на контролер спочатку вона компресується, для того щоб зменшити витрати ресурсів, також замість того, щоб надсилати матриці трафіку контролеру кожен раз коли вони оновлюються, пропонується надсилати їх лише тобі коли змінюється набір пар [20].

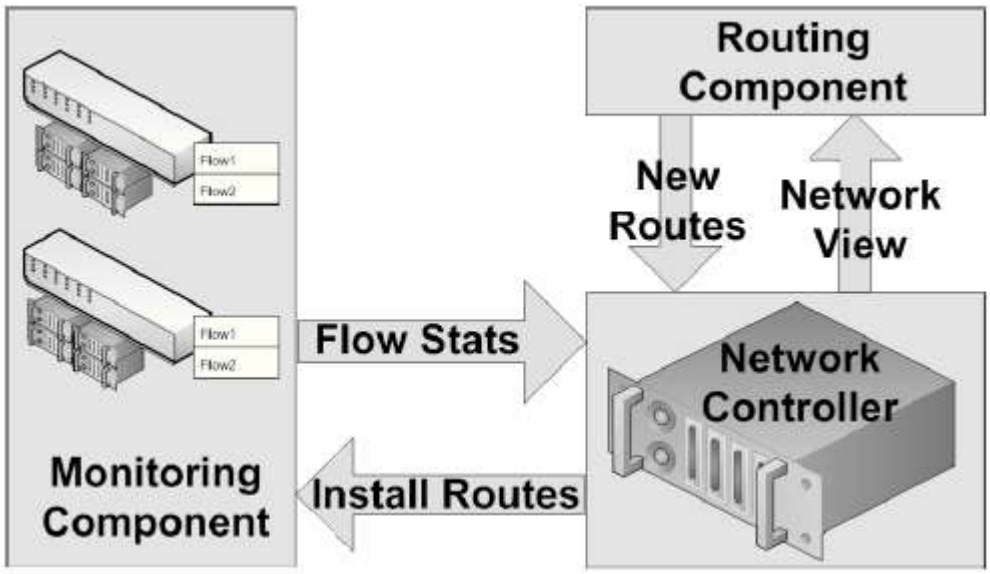


Рисунок 2.5 – Архітектура MicroTE

Контролер мережі відповідає за завдання агрегації інформації, яку він отримує з блоку моніторингу. Після цього він формує представлення мережі з обсягом трафіку між парами комутаторів та передбачуваністю трафіку, і передає це представлення блоку маршрутизації для формування маршрутів. Також контролер отримані маршрути від блоку маршрутизації встановлює у таблиці маршрутизації на комутаторах [20].

Блок маршрутизації обчислює маршрути базуючись на наданій контролером агрегованій інформації. Він знаходить маршрути спочатку для передбачуваного трафіку, потім для непередбачуваного по модифікованій схемі ECMP: вага на кожному шляху відображає кількість пропускної здатності, яка вже споживається передбачуваним трафіком, таким чином умикається ризик гіршої маршрутизації ніж у традиційній ECMP [20].

Численні моделювання та експерименти показали, що MicroTE надає близькі до оптимальних результатів по продуктивності коли трафік передбачуваний, і надає такі ж результати як і ECMP, коли трафік непередбачуваний. MicroTE працює краще за різними показниками ні багато інших сучасних запропонованих рішень. [20]

MiceTrap – підхід, що пропонується у роботі [21], робить акцент на малих потоках на відміну від описаних вище рішень, які концентрують увагу на управлінні

великими потоками, у той час як інші (малі) потоки оброблюються загальноприйнятим способом. Автори роботи вказують, що хоча малі потоки кожен окремо не становить загрози, але так як таких потоків 90% у мережі, то разом вони можуть значно впливати на роботу мережі, тому спрямовування технік конструювання трафіку лише на 10% потоків (великі потоки) може бути не ефективним. MiceTrap більш гнучко управляє потоками ніж попередні методи, тому що в ньому закладено три стратегії поведінки: великі покоти обробляються відповідною схемою пересилання великих потоків; малі потоки обробляються схемою MiceTrap, вони групуються по адресі вузла-призначення; а до тих пір поки потік не класифіковано він обробляється традиційною схемою ECMP [21].

Архітектура пропонованого рішення включає:

- модуль виявлення великих-потоків на граничних хостах;
- модуль для багатошляхової передачі агрегованих малих потоків;
- додаткові модулі для контролера, які виконують агрегацію та маршрутизацію (рис 2.6).

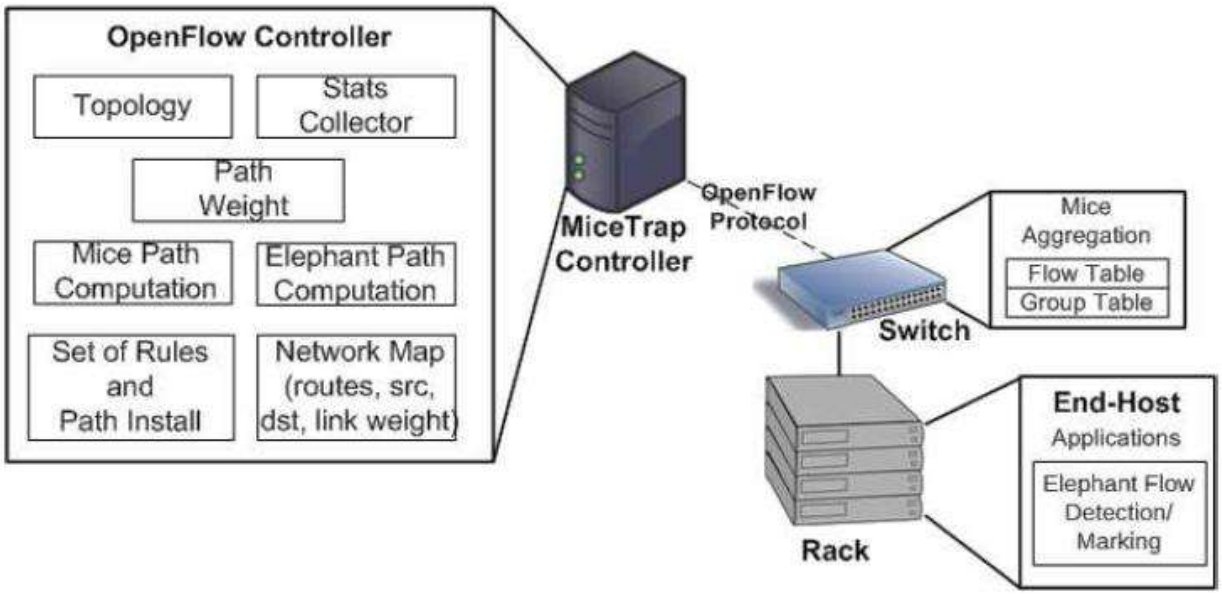


Рисунок 2.6 – Архітектура MiceTrap

Для класифікації потоків використовується механізм виявлення великих потоків на граничних хостах, якщо протягом вказаного періоду часу потік не позначений як великий, то він автоматично позначається як малий [21].

Модуль Mice Aggregation групує малі потоки по IP-адресі хоста призначення, групування значно зменшує кількість записів у таблицях та кількість повідомлень: замість надсилання повідомлення кожному потоку, одне повідомлення надсилається усій групі [21].

Щодо контролеру MiceTrap, то його основні частини це: блок формування шляхів, блок формування множини найкоротших шляхів для будь-якої пари вузлів (тут використовується алгоритм багатошляхової зваженої маршрутизації), блок формування шляхів для великих потоків [21].

Автори запропоновали алгоритм, що розраховує коефіцієнти (ваги маршрутів) для розподілу трафіку між маршрутами, ці коефіцієнти враховують поточне завантаження мережі, тому алгоритм досягає ефективного балансування навантажень. Чим більша вага маршруту, тим менш він завантажений. Отже, коли необхідно передати групу малих потоків до одного й того самого вузла-призначення по кільком маршрутам, то кожен маршрут ділиться на частини, які в свою чергу поділяються на сегменти – власне зв'язки, і для кожного сегменту знаходиться коефіцієнт використання зв'язку [21]:

$$L_{kji} = \frac{\gamma_{kji}}{c_{kji}}, \tag{2.3}$$

де γ_{kji} – завантаженість k-го зв'язку j-ої частини i-го маршруту;

c_{kji}– пропускна здатність k-го зв'язку j-ої частини i-го маршруту.

Після цього обраховуються коефіцієнти використання маршрутів шляхом сумування коефіцієнтів використання зв'язків та розділення на кількість підмаршрутів. І потім обраховується ваги для кожного маршруту за формулою [21]:

$$w_i = 1 - \frac{(\sum_j \sum_k L_{kji}) / M_i}{\sum_i (\sum_j \sum_k L_{kji}) / M_i}, \tag{2.4}$$

де L_{kji} – коефіцієнт використання k-го зв'язку j-ої частини i-го маршруту;

M_i – кількість частин i-го маршруту.

Даний алгоритм досягає кращого балансування навантажень ніж ECMP, та використовуючи управляння на рівні потоків дозволяє спростити процес управління трафіком у мережі.

Зробивши аналіз існуючих рішень по конструюванню трафіку в програмно-конфігурованих мережах було виявлено, що для багатошляхової маршрутизації зазвичай використовується досить обмежена схема рівномірного розподілу даних між маршрутами - ECMP. Також було виявлено, що не враховується пріоритетність трафіку (тип трафіку), але це критично необхідно для забезпечення необхідного рівня якості обслуговування (Quality of Service, QoS). В додаток до цього не розглядається також питання визначення стратегії поведінки при перевантаженні ділянки мережі чи виході з ладу вузла з маршруту по якому передаються дані.

У роботі [30] наведено аналіз різних методів оптимізації трафіку в мережах SDN, а саме переваги використання технології SDN для вирішення завдань балансування навантаження. Контроль перевантаженості мережі має за собою дуже важливе завдання. Перевантаження мережі відбувається в ситуації, коли багато пакетів переміщуються по одній і тій же частині підмережі, перевищуючи максимальную пропускну ємність каналу. Внаслідок чого відбувається затримка і втрата пакетів, що призводить до помітного зниження продуктивності мережі та низької якості обслуговування. Контроль перевантаження повинен зменшити затримку передачі і споживання смуги пропускання, викликаючи оптимізацію швидкості передачі і, отже, продуктивності мережі. Цей метод називається «Управління перевантаженнями з використанням Openflow в програмно - конфігурованих центрах обробки даних мережі».

У запропонованому методі розпізнавання перевантажень в зв’язках між вузлами виявляється шляхом постійної статистичної перевірки портів комутаторів. Якщо знаходиться потік з більш ніж 70% використання, то на основі поточної статистики мережі починається перерахунок нового шляху з більш вільними ресурсами.

У роботі [31] розглянуто централізоване керування трафіком за допомогою контролера SDN і дозволяє зменшити затримку і втрату пакетів в комп’ютерній мережі.

Була представлена новітня система якості обслуговування(QoS), яка заснована на централізованому плануванні трафіку для SDN. Конкретні питання, на яких зосереджено увагу в рамках, включають наступне. По-перше, в мережі, що має перекриваючі вузли або зв'язки, як вибрати відповідні шляхи для бізнес-потоків декількох додатків. По-друге, відповідно до вимог QoS сума смуг пропускання, призначених кожному шляху, не може перевищувати фізичну ємність каналу [31].

Для вирішення цих проблем платформа розробляє кілька ключових компонентів, а саме: модуль відображення топології мережі, модуль збору стану мережі, модуль вибору шляху і модуль динамічної конфігурації шляху. Перші два модулі відповідають за моніторинг рівня даних для збору оновлень топології, динамічного збору параметрів мережі і створення ваг кожного шляху на схемі мережі. Модуль вибору шляху відповідає за визначення найкращого шляху відповідно до вимог QoS бізнесу та значень ваг у мережевій діаграмі. Модуль конфігурації динамічного шляху відповідає за оновлення правил маршрутизації на рівніпересилання данихв потрібнийчас. В якостіосновногомодуля структури модуль вибору шляху приймає рішення про планування з використанням моделі багатокомпонентного потоку і обмеженого найкоротшого шляху (MCFCSP). Модель встановлює стратегію переадресації в мережі G = D (N, A) для набору бізнес-потоків K, а цільовою функцією моделі є довжина затримок зв'язку і розмір втрат пакетів. На основі обмежень змінних параметрів, що надаються моделлю, можна вирішитизавдання NP повне для отримання глобальноїстратегії планування. Автори підтвердили, що це рішення на основі платформи Mininet може ефективно задовольняти вимогам QoS бізнес-додатків реального часу, включаючи передачу відео і файлів.

Також на сьогоднішній день існує потоковий алгоритм багатошляхової маршрутизації [32], що дозволяє одночасно формувати множин шляхів між різними парами вузлів мережі. Даний алгоритм відрізняється меншою часовою складністюв порівнянні з відомимиалгоритмамибагатошляхової маршрутизації, але при цьому не враховується метрика шляхів. Це ускладнює процес конструювання трафіку. Даний

метод може бути використаний не тільки для конструювання трафікумережі SDN, а ймережі з мобільних пристроїв(MobileAd hoc Network - MANET).

Маршрутизація є важливим аспектом мережі Ad Hoc через її особливостей. У зв'язку з цим найбільш перспективним підходом є використання безпечної багатошляхової маршрутизації. Для забезпечення безпеки протоколів багатошляхової маршрутизації висуваються дві групи вимог. Вимоги визначають першу групу, що утворює множин найбезпечніших шляхів, як правило, непересічних. Друга група вимог пов'язана з необхідністю забезпечення низької обчислювальної складності, швидкої збіжності і мінімальних обсягів генерованого сервісного трафіку. Алгоритми багатошляхової маршрутизації діляться на комбінаторні та потокові алгоритми. Комбінаторні моделі базуються на основі математичного опису комп'ютерної мережі у вигляді орієнтованого або неорієнтованого графа з подальшим використанням комбінаторних алгоритмів пошуку множини найкоротших шляхів між заданими парами вузлів. Потокові алгоритми виключають операцію спрямованого перерахування і водночас формують множин допустимих шляхів [32].

У роботі [33] пропонується модифікований алгоритм формування максимальної кількості непересічних шляхів з урахуванням їх метрики. Оптимізація шляхів здійснюється за рахунок специфічного перенастроювання маршруту враховуючи суміжні. Перенастроювання здійснюється в межах підграфів, що включають тільки вершини основного і суміжного маршруту. Такий підхід дозволяє скоротити межі формування оптимального шляху і зменшує складність цієї процедури.

Розширення сфери використання сучасних мережевих технологій ставить нові, що вимагають більшої якості обслуговування (QoS) на рівні систем передачі інформації і побудови трафіку. При використанні комп'ютерних мереж динамічно змінюються такі характеристики, як пропускна здатність каналів передачі, завантаження вузлів комутації, реконфігурація мережі. Це призводить до

необхідності здійснення маршрутизації, що, в свою чергу, збільшує об'єм службового трафіка і впливає на ефективність функціонування комп'ютерної мережі [32]

Одним з найбільш ефективних рішень цього завдання є використання багатошляхової маршрутизації, яка дозволяє:

– підвищення ефективності доступу до розподілених ресурсів за рахунок оптимізації процедури побудови трафіку;
– забезпечення більш рівномірного завантаження мережі;
– підвищення безпеки і надійності передачі даних.

У більшості випадків при формуванні множини не пересічних шляхів необхідно враховувати метрику шляху. З метою формування множини не перетинаються шляхів з урахуванням метрики в даній роботі запропоновано модифікований потоковий алгоритм формування множини не перетинаються шляхів. На початковому етапі формується множин не пересічних шляхів за критерієм мінімального зовнішнього ступеня вершин. Для кожного шляху розраховується метрика. Потім проводиться послідовна корекція шляхів, починаючи зі шляху з максимальною метрикою. Сайти шляху замінюються сайтами суміжного тупикового шляху з мінімальною метрикою, починаючи з першої суміжної вершини. шляхи Li і Lj суміжні, якщо присутній один або кілька суміжних ребер. Ребро er, k між вершинами vr є Vi та Vk до VJ, що належать фактично шляхам Li і Lj, називається суміжним ребром. Ступінь суміжності шляхів Li і Lj визначається числом суміжних ребер між цими шляхами. Процес повторюється до того, як буде обраний остаточний шлях [33].

У роботі [34] представлено метод, який описує багатошляхову маршрутизацію в мережевих центрах даних, який за рахунок обліку топології мережі і централізованого управління дозволяє оптимізувати процедуру формування множини непересічних шляхів.

Сучасні комп'ютерні мережі характеризуються великою розмірністю і різноманітністю обладнання, включаючи мобільний зв'язок і мобільні точки доступу. У зв'язкуз цим ускладнюється процес управління такого родумережами і побудови даних по їх мережевих центрах, розподілу кластерів і використання хмарних

технологій. Для вирішення цих завдань в даний час широко використовується програмно-конфігурована мережева технологія (SDN). Ця технологія дозволяє підвищити ефективність мережевого обладнання, знизити експлуатаційні витрати, підвищити керованість і безпека мережі на основі багатошляхової маршрутизації. Технологія дозволяє керувати мережею на програмному рівні, тим самим розширюючи функціональні можливості управління передачею інформації по мережі [34].

В даний час у зв'язку з ростом попиту на обчислювальну потужність і обсяг інформації актуальними стають мережі центрів обробки даних (ЦОД). DCN - це високошвидкісна обробка і зберігання великих обсягів даних. DCNs повинен забезпечувати високу пропускну здатність і надійність даних. Це, в свою чергу, пред'являє високі вимоги до конструювання трафіку (TE) в таких системах. У зв'язку з цим в даний час важливо розробити способи організації роботи DCNs і TE з урахуванням особливостей і переваг SDN. При побудові сучасних DCNs в основному використовується топологія Fat Tree, яка є однією з найбільш поширених топологій для побудови розподілених систем, спрямованих на вирішення високопродуктивних завдань. Топологія Fat Tree - це дерево, листя якого є обчислювальними пристроями, а вузли - комутаторами. В цьому випадку для комутаторів більш високого рівня ширина смуги каналів більше, тобто з'єднання з іншими вершинами "товщі". Тому ця топологія називалася Fat Tree. При використанні топології Fat Tree DCN утворює чотирирівневу структуру. Верхній рівень складається з основних елементів, агрегатні перемикачі розташовані на один рівень нижче, наступний рівень має крайові перемикачі, до яких підключені шланги. У свою чергу, для DCN, що складається з сотень і тисяч великомасштабних серверів, топологія Fat Tree, заснована на принципах самоподібності, трансформується в так звану мережу Dragonfly. Враховуючи фрактальний характер топології мережі Dragonfly, можна спростити процес проектування трафіку, а також процес маршрутизації [34].

У роботі [35] запропоновано архітектуру оверлейної мережі на основі SDN для потокової передачі відео через інтернет. Потім пропонується динамічний підхід

багатопроменевої підготовки, який буде використовуватися в даних мережах. На відміну від багатьох існуючих робіт, що зосереджені на замірах затримки і швидкості базової мережі, пропонується розглядати нові дві метрики доступної смуги пропускання і перевірка доступності під час вибору шляху.

Причина в тому, що такі показники, як затримка або швидкість втрат, не є індикатором завантаження прямого трафіку і вказують, перевантажена шлях. Однак доступна пропускна здатність вказує обсяг трафіку, який все ще може бути перенаправлений через шлях, перш ніж він перевантажений. Тому, спираючись на таку метрику, ми можемо забезпечити механізм управління перевантаженням в мережі. З іншого боку, наявність компонента вказує на ймовірність того, що компонент знаходиться у функціональному стані в будь-який довільний час. Це значна метрика QoS для опису надійності, і ми враховуємо це для підвищення надійності мережі. Спираючись на результат активних зондувальних вимірювань на доступну смугу пропускання і інформацію про доступність посилань, запропонований нами алгоритм пропонує неявний механізм балансування навантаження, підвищує надійність мережі, одночасно намагаючись зменшити кількість перерахунків шляхи для ефективного розміщення мінливого у часі трафіку в інтернеті. У цій роботі, спираючись на накладку архітектури, ми пропонуємо механізм, що дозволяє адаптивно знайти каналів вводу-виводу, забезпечуючи балансування навантаження і високу доступність для потокового мовлення через Інтернет. Для зниження вартості і накладних витрат на розрахунок шляху ми пропонуємо мережу накладення на основі SDN. Наскільки нам відомо, така адаптивна багатошляхова підготовка в мережах накладення на основі SDN досі не досліджена [35].

Спираючись на програмне забезпечення Software Defined Networking, первинна мета якого - знизити вартість і накладні витрати на (пере)розрахунок шляху при досягненні високої відмовостійкості, балансування навантаження і гарантованого QoS передачі медіа файлів через Інтернет. Крім того, мета цього алгоритму - досягти більшої гнучкості в порівнянні з розгортанням приватних посилань або (G)тунелів

MPLS через інтернет. Нижче ми пояснюємо вибір, зроблений для досягнення цих цілей. Ми вирішили використовувати інтернет замість приватних посилань або тунелів MPLS для мінімізації витрат. Оверлейні мережі дозволяють нам розгорнути пропонований підхід багатопроменевої підготовки без внесення будь-яких змін в базовий інтернет. Використовуючи архітектуру накладення на основі SDN, постачальники послуг (SP) можуть самі керувати трафіком і реагувати на різні події, такі як збої. Оверлейна мережа складається з трьох типів вузлів: i) точки входу, ii) проміжні точки і iii) граничні точки. Ці вузли контролюються центральним елементом програмного забезпечення (контролер), який знаходиться у віданні СП. Точка входу працює як джерело потоків A / V і перенаправляє кожен потік на один / кілька проміжних вузлів. Проміжні вузли відіграють важливу роль в архітектурі і можуть направляти трафік в різних напрямках для відправки трафіку вузлів, близьким до межовим точкам поряд з кінцевими користувачами. Вони ведуть себе дуже схоже на комутатори SDN з тією різницею, що вони не підключені безпосередньо, а через IP-тунелі (наприклад, тунелі GRE). Оскільки прикордонна точка може отримувати декілька примірників потоку (за багатопроменевої підготовки), вона повинна об'єднати пакети різних потоків, відкинути дублікати, відновити втрачені і х пакетів поза порядку, щоб зробити один примірник потоку для відправки кінцевому користувачеві. Оскільки запропонована архітектура спрямована на підвищення відмовостійкості передачі інформації через Інтернет, слід передбачити метод переправлення трафіку в разі збоїв [35].

Центральний контролер виконує роль управління трафіком від збою, застосовуючи альтернативні шляхи маршрутизації до прикордонних точок / точки входу(ів). Для цього контролер повинен мати повне уявлення про мережу накладення. Кожен вузол накладення регулярно відправляє зонди на інші вузли накладення для перевірки підключення і відправляє результати вимірювань контролеру. В залежності від результату цього активного зондування контролер може виявити відмову/погіршення і застосувати альтернативні шляхи. Крім збоїв, оскільки ми покладаємося на інтернет замість приватних посилань або тунелів MPLS, щоб

виконати вимоги QoS медіа-інтенсивних програм з точки зору пропускної здатності, затримки і тремтіння, контролер може змінити шляхи для кожної передачі мультимедіа кілька разів протягом його життя, щоб задовольнити цим вимогам QoS. Це накладає великі накладні витрати на контролер, щоб реконструювати записи потоку в вузлах накладення. Щоб зменшити ці накладні витрати ми розглянули варіант вихідної маршрутизації, в якому контролер повинен реконструювати нещодавно обчислені шляху тільки в точці входу/краю шляху [35].

2.4 Офлайн та онлайн методи розподілу ресурсів

Проблеми планування ресурсів загалом класифікуються як офлайн-планування та онлайн-планування. У офлайн-плануванні час випуску, час обробки, дата завершення та інші необхідні дані кожної з завдань відомі до визначення графіка робіт на рівномірних паралельних машинах. Алгоритми планування онлайн роблять рішення щодо планування в кожний момент часу на основі характеристик робочих місць, які прийшли до цих пір без знань про роботу, яка може прибути в майбутньому. Онлайн-планування поділяється на наступні типи. Планування завдань по одному: у цьому типі планування завдання, які доступні, впорядковуються в певному списку. Потім кожна з завдань буде призначена для певної рівномірної паралельної машини або до деякого часового інтервалу один за одним зі списку перед тим, як будуть побачені наступні завдання. Невідомий час виконання: у певній ситуації планування час роботи кожного завдання може бути невідомим до завершення завдання. Онлайн-алгоритм знає лише, чи виконується робота чи ні. У будь-який час всі наявні робочі місця знаходяться в розпорядженні алгоритму. Вакансії надходять з плином часу: у цьому типі планування час роботи кожного завдання відомий на момент прибуття цього завдання, але час прибуття кожного завдання не відомо заздалегідь. Планування інтервалу: у цьому типі планування кожне завдання повинно виконуватися в заздалегідь визначений інтервал часу. Якщо неможливо виконати завдання за цей інтервал часу, завдання буде відхилено. Тут метою є максимізація кількості прийнятих робочих місць [22].

Оскільки трафік в програмно-конфігурованій мережі постійно змінюється, тобто його об’єм та тип, та сама конфігурація мережі також можеш змінюватися, наприклад, вихід із ладу одного із комунікаторів або зменшення пропускної здатності. Тому офлайн тип планування не підходить для вирішення проблеми конструювання трафіку в програмно-конфігурованій мережі, оскільки потрібно знати всі параметри цієї мережі на початку планування. Також варто зазначити, що для такого типу планування потрібно, щоб трафік, який постійно курсує в мережі, був детермінований, але дані умови практично ніколи не виконуються в реальних мережах. Якщо провести аналіз завантаженості мережі протягом доби, то можна помітити, що існують так звані піки завантаженості. Тому далі буде розглядатися онлайн метод планування, який дозволяє в реальному часі відстежувати зміни та швидко реагувати для ефективного розподілу трафіку з урахуванням стану мережі.

2.5 Розроблення методу конструювання трафіку

Для того, щоб дані були передані у відповідності з належним значенням якості обслуговування, при конструюванні трафіку необхідно враховувати поточний стан завантаження маршрутів, тип трафіку і параметри маршрутів (наприклад такі як довжина та надійність), а також слідкувати за передачею і якщо необхідно швидко знаходити альтернативний варіант передачі. Тому алгоритм конструювання трафіку повинен враховувати всі ці складові для забезпечення максимальної надійності.

Такий алгоритм повинен на вузлі «відправника» направляти трафік по наявним маршрутам у відповідності до типу трафіку і параметрів маршрутів. Дані високої важливості передавати по безпечним маршрутам, щоб уникнути втрати даних. Дані чутливі до часу – по коротким маршрутам, це найвищі пріоритети. На останок навантажувати маршрути даними нечутливими до часу у відповідності до ресурсів, що залишилися.

Запропоноване рішення буде досягати кращого балансування навантажень ніж традиційні підходи, оскільки попередні не враховують стан мережі та тип трафіку.

Трафік мережі умовно може бути поділений на дві великі категорії **еластичний** та **нееластичний трафік**. Під еластичним розуміється такий тип трафіку який може пристосовуватися до змін затримки і пропускної здатності в широкому діапазоні значень, продовжуючи задовольняти потреби застосувань (обмін файлами, електронна пошта). Для такого трафіку критично метрикою є надійність доставки, тобто щоб вся інформація повинна бути доставлена. Під нееластичним розуміється такий тип трафіку який погано пристосовується або взагалі не здатний пристосовуватися до змін затримки і пропускної здатності мережі (наприклад трафік реального часу, аудіо або відео конференція). Тому для такого трафіку важливою метрикою є час затримки.

Маршрути можуть бути охарактеризовані з використанням таких параметрів як:

- пропускна здатність – швидкість передачі даних;
- час затримки – час за який трафік буде доставлено у незавантаженій мережі;
- завантаженість – визначає яка частина пропускної здатності зайнята на даний момент;
- надійність – визначається кількістю помилок при передачі.

При плануванні трафіку потрібно враховувати метрику для кожного підходящого шляху, для того щоб розподіл даних між маршрутами, на вузлі-відправнику, відбувався з урахуванням QoS для кожного типу трафіку [27].

$$w_i = 1 - \frac{(\sum_j \sum_k L_{kji}) / M_i}{\sum_i (\sum_j \sum_k L_{kji}) / M_i}, \qquad (2.5)$$

де $\boldsymbol{B_i}$ – ефективна пропускна здатність, що визначається як добуток пропускної здатності на завантаженість;

$\boldsymbol{D_C}$ – час затримки;

$\boldsymbol{k_{1i}}$ і $\boldsymbol{k_{2i}}$ – вагові коефіцієнти пропускної здатності та затримки;

$\boldsymbol{r}$ – надійність, відсоток інформації успішно переданої до наступного вузла [27].

Дана метрика враховує всі необхідні показники маршрутів, а вагові коефіцієнти дозволяють балансувати важливість цих параметрів для кожного з категорій трафіку. Тобто для еластичного трафіку більш важливою є надійність, тому у формулі ваговий

коефіцієнт при надійності буде більшим ніж інші, а для нееластичного важливий час затримки, тому у формулі ваговий коефіцієнт біля затримки буде більшим ніж інші [27].

Якщо між двома вершинами існує декілька маршрутів, то потрібно вибрати найкращий враховуючи метрику W. Виходячи з цього, метрика кожного маршруту визначається як мінімальне значення з метрик усіх вершин, що входять до маршруту. Найкращий маршрут визначається з максимального значення [27].

$$P_{best} = \max_{i\in(0,n)} \min_{j\in(0.m)} W_{ij}, \qquad (2.6)$$

де $\boldsymbol{n}$ – кількість маршрутів, які можуть опрацювати між двома заданими вершинами;

$\boldsymbol{m}$ – кількість вершин, які входять до маршруту.

Після того як метрики усіх шлях обраховані і пакет відправлено по кращому маршруту, відповідно до параметрів трафіку, процес повторюється для наступного пакету. Такий розподіл пакетів по маршрутам дозволяє досягати менших втрат пакетів, а особливо пакетів еластичного трафіку [27].

Блок-схема алгоритму наведена на рисунку 2.6. Робота пропонованого алгоритму конструювання трафіку може бути описана такою послідовністю дій:

- визначення категорії трафіку
- обрахування значень узагальненої метрики для кожного доступного маршруту з вершини-відправника до вершини-отримувача.
- за наявності більше одного маршруту та при невеликій завантаженості, маршрут з низькою надійністю ігнорується
- вибір найкращого маршруту на основі підрахованої метрики з урахуванням рівномірного розподілу та відправка пакету по вибраному маршруту

Для розподілу трафіку між множиною шляхів в SDN буде використовуватися запропонований спосіб TE, блок-схема якого зображена на рис. 2.6. Основна ідея способу полягає в тому, що при наявності достатньої кількості альтернативних незавантажених маршрутів, маршрут з показником надійності менше 50 % буде ігноруватися, що дозволяє рівномірно розподілити між найкращими маршрутами

інформацію. ТЕ з урахуванням завантаженості шляхів дозволяє мінімізувати затори та економніше використовувати ресурси мережі [28].

Блок-схема алгоритму розподілу трафіку наведена у додатку А, плакат 1.

Висновки до розділу

У результаті виконання другого розділу була описана змістовна постановка задачі, розписана математична постановка задачі конструювання трафіку на основі багатошляхової маршрутизації.

Проведено ґрунтовний огляд способів конструювання трафіку, сучасних технік, які використовуються на практиці. Визначено переваги та недоліки представлених способів ТЕ на основі багато шляхової маршрутизації. Основним недоліком являється той факт, що при ТЕ за допомогою відомих способів в традиційній мережі не враховують тип трафіку при його відправці. Тому для розподілу трафіку було вирішено використовувати узагальнену метрику для маршрутів та врахувати параметри трафіку.

Проаналізувавши всі недоліки та переваги існуючих методів, був запропонований спосіб маршрутизації трафіку. Основна ідея способу полягає у виборі найкращого маршрут, враховуючи метрику кожного маршруту.

3 РЕЗУЛЬТАТИ ЕСПЕРИТМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

Розглянемо програмно-конфігуровану мережу у вигляді неорієнтованого графу. Мережа складається з 12 вузлів та контролера, який контролює загальний стан мережі. Структура мережі у вигляді графу наведена у додатку А, плакат 2.

Для конструювання трафіку в бездротовій програмно-конфігурованій мережі та ефективної маршрутизації необхідно сформувати множину шляхів [29], що не перетинаються. Це дозволить оптимально організувати розподіл інформації між шляхами, враховуючи метрики кожного шляху.

Припустимо, що для кожного вузла задана відповідна надійність, яка може набувати значень з інтервалу від 0 до 100 %. Надійність вузла визначається ймовірністю виходу з ладу відповідного вузла (табл. 3.1).

Таблиця 3.1 – Надійність вузлів

Вузол	Надійність, %	Вузол	Надійність, %	Вузол	Надійність, %
1	90	5	98	9	85
2	80	6	98	10	98
3	75	7	80	11	98
4	80	8	80	12	85

Для кращого моделювання на запропонованому графі кожній вершині було задано надійність з врахуванням її розташування в мережі та кількості зв’язків з іншими вершинами.

Приклад структури мережі з множиною шляхів, що не перетинаються між вершиною 1 та 9 знайдених за допомогою алгоритму «зворотної хвилі», наведена у додатку А, плакат 3.

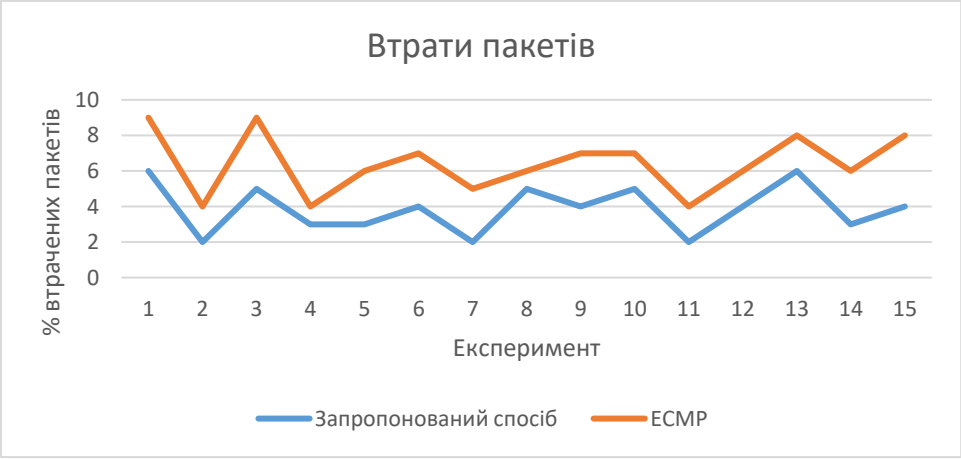
Можливі маршрути між вузлами 1 та 9: шлях № 1=1→2→3→ 4→9 (довжина шляху № 1: 4); шлях № 2=1→10→11→9 (довжина шляху № 2: 3); шлях № 3=1→6→5→9 (довжина шляху № 3: 3); шлях № 4=1→7→8→9 (довжина шляху № 3: 3)

Використовуючи запропонований спосіб конструювання трафіку було виконано розподіл інформації між шляхами з урахуванням параметрів QoS для перевірки ефективності даного способу з найбільш розповсюдженим способом на сьогоднішній день. В результаті експерименту були отримані результати втрати пакетів та завантаженість кожного маршруту.

На рис. 3.1 представлені результати ймовірності втрати пакетів при нормальному законі їх розподілу.

ECMP method
Problems with pockets happens: 9 times
Path: [1->6->5->9, 26], pocket send: 26, percent: 0,26%
Path: [1->10->11->9, 23], pocket send: 23, percent: 0,23%
Path: [1->7->8->9, 28], pocket send: 28, percent: 0,28%
Path: [1->2->3->4->9, 21], pocket send: 21, percent: 0,21%
ECMP with modification method
Problems with pockets happens: 3 times
Path: [1->6->5->9, 27], pocket send: 27, percent: 0,27%
Path: [1->10->11->9, 27], pocket send: 27, percent: 0,27%
Path: [1->7->8->9, 31], pocket send: 31, percent: 0,31%
Path: [1->2->3->4->9, 12], pocket send: 12, percent: 0,12%

а



б

Рисунок 3.1 – Порівняння втрати пакетів: а – результати моделювання; б – графічне представлення

Розглянемо приклад розподілу трафіку при відправці 100 пакетів з вузла 1 до

вузла 9. Результати наведені у таблиці 3.2.

Таблиця 3.2 – Розподіл трафіку по шляхам

Шлях	Запропонований спосіб	ECMP
1→6→ 5→ 9	27 %	26 %
1→10→11→ 9	27 %	23 %
1→7→ 8→ 9	31 %	28 %
1→2→ 3→ 4→ 9	12 %	21 %

Запропонований алгоритм забезпечить більш надійну доставку пакетів. В даному випадку менш надійним являється шлях 1→2→ 3→ 4→ 9 і тому цей шлях найменш завантажений.

Розглянемо випадок виходу з ладу вузла 11 (рис. 3.2).

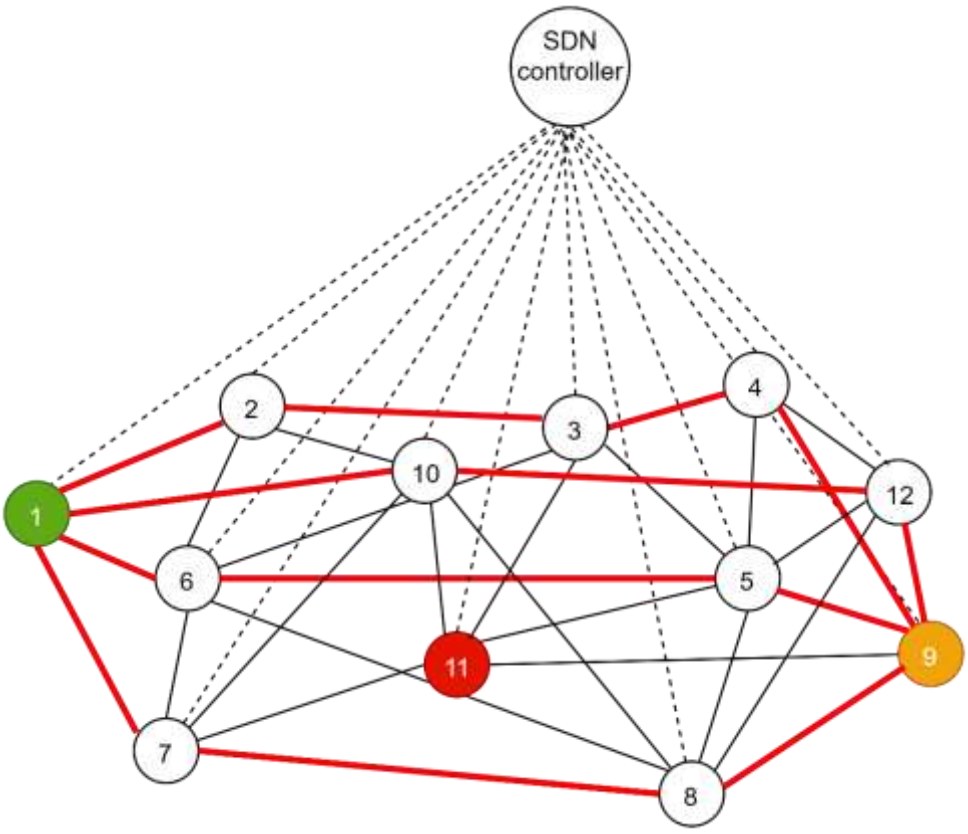


Рисунок 3.2 – Можливі маршрути між вузлами 1 та 9: шлях № 1=1→2→3→4→9 (довжина шляху № 1: 4); шлях № 2=1→10→12→9 (довжина шляху № 2: 3); шлях № 3=1→6→5→9 (довжина шляху № 3: 3); шлях № 4=1→7→8→9 (довжина шляху № 3: 3)

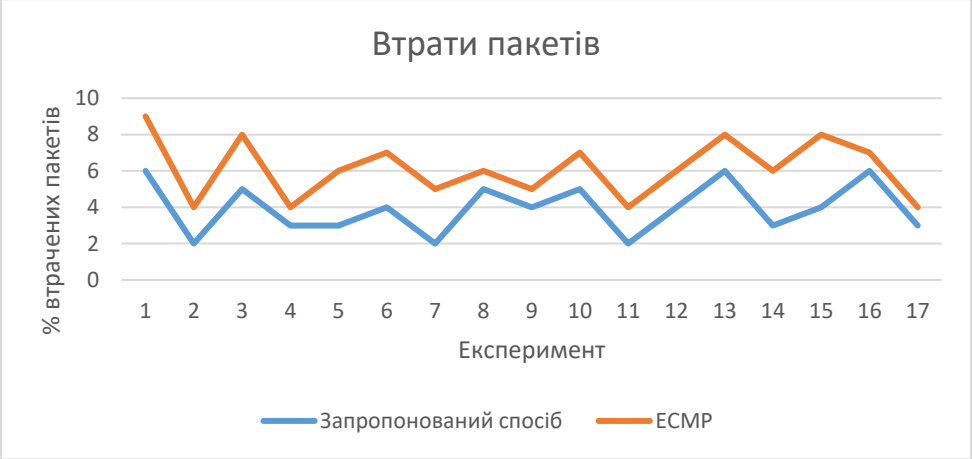
В результату моделювання цього випадку були отриманні наступні результати

порівняння запропонованого алгоритму з методом ECMP.

На рис. 3.3 представлені результати ймовірності втрати пакетів при виході з ладу вузла 11.

ECMP method
Problems with pockets happens: 9 times
Path: [1->6->5->9, 26], pocket send: 26, percent: 0,26%
Path: [1->10->11->9, 26], pocket send: 26, percent: 0,26%
Path: [1->7->8->9, 28], pocket send: 28, percent: 0,28%
Path: [1->2->3->4->9, 20], pocket send: 20, percent: 0,2%
ECMP with modification method
Problems with pockets happens: 6 times
Path: [1->6->5->9, 33], pocket send: 33, percent: 0,33%
Path: [1->10->11->9, 31], pocket send: 31, percent: 0,31%
Path: [1->7->8->9, 30], pocket send: 30, percent: 0,30%
Path: [1->2->3->4->9, 5], pocket send: 5, percent: 0,5%

а



б

Рисунок 3.3 – Порівняння втрати пакетів: а – результати моделювання; б – графічне представлення

Розглянемо приклад розподілу трафіку при відправці 100 пакетів з вузла 1 до вузла 9. Результати наведені у табл. 3.3.

Таблиця 3.3 – Розподіл трафіку по шляхам

Шлях	Запропонований спосіб	ECMP
1→6→5→9	33 %	26 %
1→10→12→9	31 %	26 %
1→7→8→9	30 %	28 %
1→2→3→4→9	5 %	20 %

Виходячи з вище наведених результатів моделювання, можна зробити висновок, що запропонований спосіб конструювання забезпечує надійнішу передачу трафіку, оскільки враховує надійність маршруту та базуючись на ній розподіляє трафік.

Детальний графік порівняння запропонованого способу та ECMP на відсоток страчених пакетів наведений у додатку А, плакат 4.

Висновки до розділу

Запропонований спосіб конструювання трафіку в бездротовій програмно-конфігурованій мережі, що дозволяє зменшити час на ремаршрутизацію та відсоток втрати пакетів за рахунок використання аналізу стану каналів при виборі оптимального шляху передачі інформації.

Отримані результати порівняльного аналізу у вигляді графіків підтверджують ефективність запропонованого способу ТЕ у порівнянні з найбільш розповсюдженим на сьогоднішній день алгоритмом ECMP. Спрощується процес вибору оптимального маршруту, який дозволяє зменшити ймовірність втрати пакетів. За допомогою запропонованого способу конструювання трафіку в бездротовій програмно-конфігурованій мережі вдалося зменшити час на реконфігурацію трафіку в середньому на 15 % .

4 ОПИС ПРОГРАМНОГО ТА ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Для створення програмного продукту було обрано наступні засоби:

Платформа розробки: .NET Framework.

Microsoft .NET — програмна технологія, запропонована фірмою Microsoft як платформа для створення як звичайних програм, так і веб-застосунків. Багато в чому є продовженням ідей та принципів, покладених в технологію Java. Одною з ідей .NET є сумісність служб, написаних різними мовами. Хоча ця можливість рекламується Microsoft як перевага .NET, платформа Java має таку саму можливість [25].

Кожна бібліотека (збірка) в .NET має свідчення про свою версію, що дозволяє усунути можливі конфлікти між різними версіями збірок.

.NET — крос-платформова технологія, в цей час існує реалізація для платформи Microsoft Windows, FreeBSD (від Microsoft) і варіант технології для ОС Linux в проекті Mono (в рамках угоди між Microsoft з Novell), DotGNU [1].

Захист авторських прав відноситься до створення середовищ виконання (CLR — Common Language Runtime) для програм .NET. Компілятори для .NET випускаються багатьма фірмами для різних мов вільно [25].

.NET поділяється на дві основні частини — середовище виконання (по суті віртуальна машина) та інструментарій розробки [25].

Середовища розробки .NET-програм: Visual Studio .NET (C++, C#, J#), SharpDevelop, Borland Developer Studio (Delphi, C#) тощо. Середовище Eclipse має додаток для розробки .NET-програм. Застосовні програми також можна розроблювати в текстовому редакторі та використовувати консольний компілятор [25].

Як і технологія Java, середовище розробки .NET створює байт-код, призначений для виконання віртуальною машиною. Вхідна мова цієї машини в .NET називається CIL (Common Intermediate Language), також відома як MSIL (Microsoft Intermediate Language), або просто IL. Застосування байт-коду дозволяє отримати крос-

платформність на рівні скомпільованого проекту (в термінах .NET: збірка), а не на рівні сирцевого тексту, як, наприклад, в C. Перед запуском збірки в середовищі виконання (CLR) байт-код перетворюється вбудованим в середовище JIT-компілятором (just in time, компіляція на льоту) в машинні коди цільового процесора [25].

Слід зазначити, що один з перших JIT-компіляторів для Java був також розроблений фірмою Microsoft (тепер в Java використовується досконаліша багаторівнева компіляція — Sun HotSpot). Сучасна технологія динамічної компіляції дозволяє досягнути аналогічного рівня швидкодії з традиційними «статичними» компіляторами (наприклад, C++) і питання швидкодії часто залежить від якості того чи іншого компілятора [25].

Мова програмування: C#.

C# — об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET. Розроблена Андерсом Гейлсбергом, Скотом Вілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft). Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію, підтримує поліморфізм, перевантаження операторів, вказівники на функції-члени класів, атрибути, події, властивості, винятки, коментарі у форматі XML. Перейнявши багато що від своїх попередників — мов C++, Delphi, Модула і Smalltalk — C#, спираючись на практику їхнього використання, виключає деякі моделі, що зарекомендували себе як проблематичні при розробці програмних систем, наприклад множинне спадкування класів (на відміну від C++).[26]

4.2 Вимоги до технічного забезпечення

Для коректної роботи даного продукту мають бути присутні такі технічні характеристики:

а) комп’ютер;

1) процесор з частотою - не менше 2.5 ГГц;

2) кількість ядер – не менше 2;

3) об’єм оперативної пам’яті - не менше 4 ГБ;

б) програмне забезпечення;

1) операційна система Windows 7+;

2) встановлено .NET Framework версії 4.5 та вище;

в) комп’ютерна периферія;

1) монітор;

2) мишка;

3) клавіатура.

4.3 Архітектура програмного забезпечення

У програмному продукті присутні такі основні класи: Node, Edge, Network, Path, Pocket, Controler. Опишемо кожен клас:

- Node: клас для вершини графу, його атрибути: номер, ip-адреса, суміжні ребра вершини, надійність;
- Edge: клас для ребра між двома вершинами, його атрибути: вершини що з’єднує, завантаженість, довжина, пропускна здатність;
- Network: клас що представляє мережу, його атрибути: вершини – набір елементів класу Node, ребра – набір елементів класу Edge, маршрути;
- Path: клас для маршруту, його атрибути: вершини через які проходить, надійність, довжина;
- Controller: клас, який представляє собою контролер мережі, саме він здійснює моделювання алгоритмів;
- Pocket: клас для пакету даних, його атрибути: тип трафіку у пакеті, розмір, початкова вершина, кінцева вершина.

Архітектура програмного забезпечення у вигляді діаграми класів наведена у додатку А, плакат 5.

На діаграмі діяльності, наведеній у додатку А, плакат 6, представлена послідовність дій для процесу моделювання передачі даних у програмі. Діаграма показує які дії має виконувати користувач та як система відповідає на його дії.

4.4 Інструкція користувача

Інструкція призначена для спрощення процесу користування програмою.

Перед запуском програми потрібно задати представлення мережі у виді графу. Для цього потрібно змінити зміст файлу *network.json*. Структура файлу наведена на рис 4.1. Це файл типу *json*, що складається з 2 ключових полів: nodes (вершини) та edges (зв'язки між вершинами). Для кожної вершина задається її номер та надійність, що розуміється як ймовірність виходу з ладу та неможливість опрацьовувати пакети. Зв’язок між двома вершинами характеризується початковою вершиною та кінцевою.

```
{
  "nodes": [
    {
      "number": 1,
      "reliability": 90
    },
    {
      "number": 2,
      "reliability": 80
    },
  ],
  "edges": [
    {
      "startId": 1,
      "endId": 2
    },
      "startId": 2,
      "endId": 6
    },
    {
      "startId": 2,
      "endId": 3
    },
  ]
}
```

Рисунок 4.1 – Представлення структури мережі через файл

Після того як запущено програму, користувачеві відкривається консольний додаток, де необхідно вказати вхідні данні для моделювання роботи запропонованого методу (рис 4.2).

```
Input start node:1
Input end node:9
```

Рисунок 4.2 – Представлення діаграми класів

На початку вказуються два параметри: початкова вершина та кінцева вершина, між якими буде проводитися моделювання. Після цього запускається процес моделювання (рис 4.3).

```
Start modeling for ECMP method!
Pocket №:0 send by path: [1->6->5->9] with current load:0
Pocket №:1 send by path: [1->10->11->9] with current load:0
Pocket №:2 send by path: [1->7->8->9] with current load:0
Pocket №:3 send by path: [1->7->8->9] with current load:0,03
Pocket №:4 send by path: [1->6->5->9] with current load:0,05
Pocket №:5 send by path: [1->7->8->9] with current load:0,09
Pocket №:6 send by path: [1->2->3->4->9] with current load:0
FAIL! Pocket with id: 6497458e-0388-44fa-a158-675fdeb8ab08 not sent from node #2!
Pocket №:7 send by path: [1->2->3->4->9] with current load:0
FAIL! Pocket with id: 87328d6c-e26e-4f02-93de-0b3ae197d004 not sent from node #2!
Pocket №:8 send by path: [1->2->3->4->9] with current load:0
Pocket №:9 send by path: [1->10->11->9] with current load:0,13
Pocket №:10 send by path: [1->6->5->9] with current load:0,14
Pocket №:11 send by path: [1->2->3->4->9] with current load:0,08
```

Рисунок 4.3 – Процес моделювання

Після моделювання отримуємо результати, що зображена на рис 4.4.

```
-----RESULTS-----
ECMP method
Problems with pockets happens: 7 times
Path: [1->6->5->9, 28], pocket send: 28, percent: 0,28%
Path: [1->10->11->9, 26], pocket send: 26, percent: 0,26%
Path: [1->7->8->9, 26], pocket send: 26, percent: 0,26%
Path: [1->2->3->4->9, 20], pocket send: 20, percent: 0,2%
ECMP with modification method
Problems with pockets happens: 3 times
Path: [1->6->5->9, 37], pocket send: 37, percent: 0,37%
Path: [1->10->11->9, 33], pocket send: 33, percent: 0,33%
Path: [1->7->8->9, 30], pocket send: 30, percent: 0,3%
```

Рисунок 4.4 – Результати моделювання

У результаті моделювання виводиться інформація про кожен маршрут справа наведена інформація, там вказано скільки пакетів пройшло і по якому маршруту.

Додатково вказується відносна завантаженість кожного маршруту, щоб можна було більш детальніше проаналізувати результати.

Копії екранних форм, на яких детальніше показано процес моделювання для запропонованого способу та способу ECMP, а також загальні результати моделювання, наведені у додатку А, плакат 7.

Висновки до розділу

У даному розділі описані основні засоби та технології, що використовувались для створення програмного продукту для моделювання та аналізу алгоритмів конструювання трафіку. Представлена архітектура програмного забезпечення у вигляді діаграми класів.

5 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Формування концепції товару чи послуги для визначеної клієнтської групи за наявних ринкових умов – це перший крок до перетворення ідеї у працюючу бізнес-модель. Даний розділ присвячено визначенню ринкових перспектив проекту, маркетинговому аналізу, тобто першому етапу розробки стартап-проекту.

5.1 Опис ідеї проекту

Опис ідеї проекту полягає у визначенні її змісту та основних напрямків застосування, а також треба виявити вигоди, які отримає користувач від використання продукту, ці дані наведено у таблиці 5.1.

Таблиця 5.1 – Опис ідеї проекту

Зміст ідеї	**Напрямки застосування**	**Вигоди для користувача**
1	**2**	**3**
Алгоритм конструювання трафіку у програмно-конфігурованій мережі, який підвищує надійність передачі даних. Для моделювання та оцінювання роботи алгоритму розроблена програма, в якій можливим є представлення мережі у вигляді графу, спостереження за ппроцесом моделювання роботи алгоритму, перегляд результатів роботи	Використання алгоритму в протоколі передачі даних на контролері програмно-конфігурованої мережі в: - центрах обробки даних; - транспортних мережах; - в мережах типу WAN	- зменшення втрачених даних під час їхньої передачі; - досягнення кращого балансування навантаження; - зменшення частоти появи перевантажень каналів зв’язку

Продовження таблиці 5.1

1	2	3
	Використання програми для прийняття рішення про доцільність застосування протоколу в існуючій мережі	Можливість швидкої та зручної перевірки ефективності роботи протоколу передачі даних в існуючої мережі
	Використання програми на етапі проектування мережі для визначення її оптимальної топології та спостереження за роботою протоколу передачі даних	Допомога в прийнятті рішення про топологію мережі на етапі її проектування та можливість протестувати роботу протоколу в майбутній мережі

Щоб визначити конкурентоспроможність ідеї мого проекту далі визначаються слабкі, сильні, та нейтральні техніко-економічні властивості та характеристики (табл. 5.2).

Таблиця 5.2 – Аналіз потенційних техніко-економічних переваг ідеї

№ п/п	Технікоекономічні характеристики ідеї	(потенційні)товари /концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Open Flow	Mahout	Mice Trap			
1	2	3	4	5	6	7	8	9
1	Легкість реалізації	Легко реалізувати	Вже реалізований	Не знайдено інформації	Дані не знайдено		+	
2	Врахування типу трафіку	Розділення трафіку на еластичний та нееластичний	Відсутнє	Розподілна великі та малі потоки	Розподілна великі та малі потоки			+
3	Відмовостійкість	Алгоритм врахову є та вирішує проблему виходу з ладу вузла	відсутня	відсутня	відсутня			+

Продовження таблиці 5.2

1	2	3	4	5	6	7	8	9
4	Легкість	Швидкота	Йде вже	Вимагає	Вимагає		+	
	Інсталювання на контролері	легко можна інсталювати на контролер	встановленим з багатьма комутаторами	спеціальної архітектури мережі, багато додаткових модулів	спеціальної архітектури мережі, багато додаткових модулів			
5	Зменшення втрат даних	Націлений на зменшення кількості втрати	Дані не знайдено	Дані не знайдено	Дані не знайдено			+
6	Запобігання перенавантаження та заторів	Алгоритм врахує стан мережі	Не врахує стан мережі	Не враховує завантаженість каналів	Враховує завантаженість каналів			+
8	Відповідність сучасним потребам	Висока	Висока	Висока	Середня		+	

Продовження таблиці 5.2

9	Простота використання	Алгоритм є досить зрозумілим і може бути легко реалізований на будь-якій мові програмування	Середня	Низька	Низька			+
10	Інтерфейс програми	Інтерфейс простий у програмі моделювання алгоритму, зручний та інтуїтивно зрозумілий	Поганий	Дані не знайдено	Дані не знайдено	+		
11	Легкість модифікації	Алгоритм можна легко модифікувати, змінювати ваги коефіцієнтів метрики, частоту обрахування метрики	Відсутня	Дані не знайдено	Дані не знайдено			+

Проаналізувавши ринок, було виявлено таких потенційних конкурентів:

- технологія OpenFlow;
- технологія Mahout;
- технологія MiceTrap.

5.2 Технологічний аудит ідеї проекту

Для визначення того, чи можлива технологічна реалізація проекту потрібно провести аудит по технологіям та визначити технологічний шлях, яким буде реалізовано ідею (табл. 5.3).

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Вираховування типу трафіку та параметрів маршрутів при розподілі трафіку	Алгоритм обрахування узагальненої метрики для маршрутів	Наявна	Доступна
2	Управління трафіком при перезавантаженні мережі	Методи аналізу стану мережі	Наявні	Доступні
		Методи маршрутизації основані на аналізу параметрів мережі		
4	Програма для моделювання пропонованого алгоритму	Будь-яка мова програмування	Наявна	Доступна
Технологічна реалізація ідеї можлива. Для реалізації проекту обрано: для розподілу трафіку – узагальнена метрика для маршрутів, для управління трафіку – метод, який базується на аналізі стану на параметрів мережі, для створення програми– c#				

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Важливим етапом є визначення напрямів розвитку проекту, необхідно враховувати ринкові загрози та можливості. Спочатку необхідно виконати аналіз попиту (табл. 5.4).

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	**Показники стану ринку (найменування)**	**Характеристика**
1	**2**	**3**
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж	
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Наявні такі обмеження:: 1. необхідність у дослідженні працездатності, ефективності запропонованого алгоритму у існуючих мережах; 2. необхідність визначення наскільки алгоритм є витривалим до збоїв та надмірних навантажень; 3. мінімізація вартості інтегрування запропонованого алгоритму у контролер програмно-конфігурованої мережі 4. необхідність у зіставлення алгоритму з найбільш популярними рішеннями
5	Специфічні вимоги до стандартизації та сертифікації	Для інтегрування алгоритму у протокол передачі даних необхідно дотримуватись стандартів документування мережевих протоколів

За попередніми оцінюваннями можна вважати, що ринок є привабливим для входження.

Далі необхідно сформувати перелік вимог до проекту для кожної групи клієнтів (табл. 5.5).

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	**Потреба, що формує ринок**	**Цільова аудиторія (цільові сегменти ринку)**	**Відмінності у поведінці різних потенційних цільових груп клієнтів**	**Вимоги споживачів до товару**
1	Потреба у протоколі передачі даних, який підвищить рівень надійності конструювання трафіку	Компанії, з програмно-конфігурованими мережами Виробники контролерів для програмно-конфігурованих мереж Власники корпоративних мереж та центрів обробки даних	Всіма цільовими групами клієнтів вимагається, щоб протокол на базі запропонованого алгоритму був надійним, відмовостійким, легким у налаштуванні, легким для модифікацій	1.Виконана з використанням існуючих стандартів до мережевих протоколів 2. Подальша підтримка проекту, технічна підтримка

Отже було визначено три основні групи потенційних клієнтів: компанії, які використовують програмно-конфігуровані мережі; виробники контролерів для програмно-конфігурованих мереж; власники корпоративних мереж та центрів обробки даних. Проаналізуємо фактори ринкового середовища (табл. 5.6 та 5.7).

Таблиця 5.6 – Фактори загроз

№ п/п	**Фактор**	**Зміст загрози**	**Можлива реакція компанії**
1	**2**	**3**	**4**
1	Головний конкурент OpenFlow вже декілька років займає головне місце як протокол для контролера програмно-конфігурованої	Споживачів можливо буде важко переконати переходити на використання інших технологій для конструювання трафіку	Необхідно знайти декілька testbed для перевірки пропонованого рішення, і розповсюдити інформацію про результати тестування серед споживачів, щоб зацікавити їх
2	Час, технології	Для проектів створення програмних рішень, характерне неточне визначення планових строків реалізації. Загрози, пов’язані з вибором оптимальної технології виконання проекту	Максимально прорахувати календарні графіки та продумати технології. Можливо залучити спеціалістів для цього завдання

Продовження таблиці 5.6

1	2	3	4
3	Інтеграція	Загрози, пов’язані з процесами інтеграції на всіх рівнях програмної архітектури	Приділити достатньо уваги та детально вивчити питання про сумісність та інтеграції запропонованого алгоритму або в новий протокол або в контролер програмно-конфігурованої мережі

Таблиця 5.7 – Фактори можливості

№ п/п	**Фактор**	**Зміст можливості**	**Можлива реакція компанії**
1	**2**	**3**	**4**
1	Орієнтація на широке коло клієнтів	Проект, що пропонується розрахований на багато цільових груп клієнтів, тому буде легко впроваджувати	Популяризація продукту якомога ширше
2	Модифікація та удосконалення методу, розширення його застосувань	Поступове поліпшення методу, збільшення його гнучкості	Розробка плану для розвитку проекту

Продовження таблиці 5.7

1	2	3	4
3	Покращення алгоритму	Коли продукт почнуть використовувати, можуть випливти питання які не повставали при його розробці. Вирішення цих питань наддасть можливість покращити продукт, тим самим зробивши його більш конкурентоспроможним	Реакція на відгуки споживачів та вирішення усіх питань якнайкраще для підвищення якості продукту

Необхідно зробити аналіз пропозиції, вивчити конкуренцію на ринку (табл. 5.8).

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	**В чому проявляється дана характеристика**	**Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)**
1	**2**	**3**
1. Тип конкуренції - монополія	Такий великий конкурент як OpenFlow з ECMP вже посів значне місце у цій ніші	Усунути головні недоліки, які є у конкурентів, запропонувати
2. За рівнем конкурентної боротьби - світова	Зараз програмно-конфігуровані мережі активно впроваджують по всьому світу	Спочатку потрібно отримати довіру на національному рівні для того щоб претендувати на світову конкуренцію

Продовження таблиці 5.8

1	2	3
3. За галузевою ознакою - міжгалузева	Міжгалузева конкуренція слідує з того що мережі, програмно-конфігуровані мережі впроваджуються для підтримки та організації роботи багатьох галузей	Спланувати з якої галузі почати, та до яких переходити потім, таким чином поступово отримати споживачів у кожній галузі
4. Конкуренція за видами товарів: товарно-видова	Товарно-видова конкуренція між іншими протоколами які враховують нехай не тип а розміри потоків Hedera, DevoFlow, Mahout	Потрібно вести конкуренцію як з технологіями - замінниками так і з тими чия назва зазнала визнання
5.За характером конкурентних переваг – неціновa	Здебільшого питання не в ціні а в простоті та ефективності інтеграції та використання	Створити умови для легкого інтегрування алгоритму у контролери програмно-конфігурованих мереж
6. За інтенсивністю – не марочна	Одним важливим критерієм є лише ефективність запропонованої технології, марка не має значення	Довести, що рішення яке пропонується є ефективним в певному розумінні цього слова

Щоб зробити висновок щодо можливості роботи на ринку з огляду на конкурентну ситуацію необхідно провести аналіз конкуренції в галузі (табл.5.9, 5.10).

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	**Прямі конкуренти в галузі**	**Потенційні конкуренти**	**Постачальники**	**Клієнти**	**Товари-замінники**
	OpenFlow ECMP Mahout	Питання є актуальним та відкритим, тому постійно публікуються та анонсуються нові пропозиції, а бар'єри входження на ринок досить високі	Для пропоно ваного проекту не є необхідн им постачал ьник	Фактори сили клієнтів це контроль якості та продуктова диференціація	Товари замінники можуть пропонувати більш легку реалізацію і таке інше, але поки зо замінників не було виявлено
Виснов ки:	Інтенсивні сть конкурент ної боротьби з боку прямих конкурент ів значна	Можливість виходу на ринок є, але поки що потенційних конкурентів не спостерігається	Постачал ьники не мають впливу на ситуацію ринку	Звичайно клієнти диктують умови роботи на ринку, їх потреби - головне	Не виявлено товарів-замінників

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	**Фактор конкурентоспроможності**	**Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)**
1	Зниження втрат даних при передачі	Алгоритм враховує тип трафіку та завантаженість маршрутів, що надає йому можливість зменшити кількість втрачених пакетів
2	Забезпечення рівномірної завантаженості	Алгоритм за рахунок того що враховує стан мережі та завантаженість маршрутів, до зволяє уникати заторів та перевантажень
3	Легкість реалізації та інтеграції з контролером	Алгоритм зручно та легко реалізовується будь-якою мовою програмування, у інтеграції алгоритму з контролером не повинно виникнути складнощів
4	Зменшення заторів у мережі	За рахунок того що Алгоритм враховує стан мережі та завантаженість маршрутів, до зволяє уникати заторів та перевантажень
5	Передача даних продовжується навіть при виході з ладу вузла у маршруті	Так як алгоритм використовує теорію прийняття рішень для вибору найкращого варіанту перенаправлення трафіку, то це дозволяє знизити втрату даних

Зазначимо сильні та слабкі сторони стартап-проекту (табл. 5.11, 5.12).

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	**Фактор конкурентоспроможності**	**Бали 1-20**	**Рейтинг товарів-конкурентів у порівнянні з системою**						
			–3	–2	–1	0	+1	+2	+3
1	Зниження втрат даних при передачі	18		+					
2	Забезпечення рівномірної завантаженості	15					+		
3	Легкість реалізації та інтеграції з контролером	18		+					
4	Зменшення заторів у мережі	29				+			

Таблиця 5.12 – SWOT- аналіз стартап-проекту

Сильні сторони	**Слабкі сторони**
Зниження втрат даних при передачі Легкість реалізації та інтеграції з контролером Зменшення заторів у мережі Алгоритм привітний до модифікацій	Забезпечення рівномірної завантаженості
Можливості	**Загрози**
Орієнтація на широке коло клієнтів Модифікація та удосконалення методу, розширення його застосувань Покращення алгоритму	Час, технології, Інтеграція Головний конкурент вже декілька років займає головне місце як протокол для контролера програмно-конфігурованої

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	**Альтернатива (орієнтовний комплекс заходів) ринкової поведінки**	**Ймовірність отримання ресурсів**	**Строки реалізації**
1	Виведення продукту на ринок, збір фідбеків від користувачі удосконалення продукту	60%	4 місяці
2	Тестування продукту закрито, відлагодження всіх нюансів та потім виведення на ринок	75%	5 місяці
3	Приєднання до головного лідера на ринку, щоб просуватися під його ім’ям	50%	2 місяці

Обирається друга альтернатива – тестування продукту закрито, відлагодження всіх нюансів та потім виведення на ринок.

5.4 Розробка ринкової стратегії проекту

Спочатку необхідно визначити стратегії охоплення ринку та базову стратегію розвитку (табл. 5.14, 5.15).

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	**Опис профілю цільової групи потенційних клієнтів**	**Готовність споживачів сприйняти продукт**	**Орієнтовний попит в межах цільової групи (сегменту)**	**Інтенсивність конкуренції в сегменті**	**Простота входу у сегмент**
1	**2**	**3**	**4**	**5**	**6**
1	Компанії, які використовують програмно-конфігуровані мережі	Висока	Невідомо	Наявна середньої інтенсивності	Середня

Продовження таблиці 5.14

1	2	3	4	5	6
2	Виробники контролерів для програмно-конфігурованих мереж	Висока готовність	Високий попит	Висока інтенсивність	Складно
3	Власники корпоративних мереж та центрів обробки даних	Середня готовність	Середній	Наявна середня конкуренція	Середня складність
Цільовими групами було обрано групи 2 тому що зараз найбільш популярно рішення продавати протоколи виробникам пристроїв. Отже стратегією охоплення ринку є стратегія концентрованого маркетингу					

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	**Стратегія охоплення ринку**	**Ключові конкурентоспроможні позиції відповідно до обраної альтернативи**	**Базова стратегія розвитку**
Тестування продукту закрито, відлагодження всіх нюансів та потім виведення на ринок	Стратегія концентрованого маркетингу	Відсутність недоліків та відлагоженість роботи продукту	Стратегія спеціалізації

Далі обирається стратегія конкурентної поведінки (табл. 5.16)

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	**Чи є проект «першопрохідцем» на ринку?**	**Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?**	**Чи буде компанія копіювати основні характеристики товару конкурента, і які?**	**Стратегія конкурентної поведінки***
1	Не зовсім	Планується, що спочатку у вільному доступі покористуються споживачі, це перша частина клієнтів, да методом хвилі вони розповсюдять товар	Ні, не буде	Стратегія виклику лідера

Далі визначимо стратегію позиціонування (табл. 5.17).

Таблиця 5.17 – Визначення стратегії позиціонування

№ п/п	**Вимоги до товару цільової аудиторії**	**Базова стратегія розвитку**	**Ключові конкурентоспроможні позиції власного стартап проекту**	**Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)**
1	Продукт має бути виконано по існуючих стандартів до мережевих протоколів	Стратегія спеціалізації	Відсутність недоліків та відлагоженість роботи продукту	Надійність передачі даних Врахування типу трафіку
2	Подальша підтримка проекту, технічна підтримка	Стратегія спеціалізації		

5.5 Розроблення маркетингової програми стартап-проекту

Щоб далі сформувати маркетингову концепцію товару необхідно підсумувати результати попереднього розділу (табл. 5.18).

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Потреба у протоколі передачі даних, який підвищить рівень надійності конструювання трафіку	зменшення втрачених даних під час їхньої передачі;	врахування типу трафіку
		досягнення кращого балансування навантаження;	перенаправлення трафіку з пошкодженої ділянки
		зменшення частоти появи перевантажень каналів зв’язку	врахування стану мережі, завантаження маршрутів

Далі розробимо трирівневу маркетингову модель товару (табл. 5.19).

Таблиця 5.19 – Трирівнева маркетингова модель товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Алгоритм конструювання трафіку, який забезпечує підвищення рівня надійності передачі даних, програма у якій моделюється даний алгоритм		
II. Товар у реальному виконанні	**Властивості/характеристики**	**/Нм**	**Вр/Тх /Тл/Е/Ор**
	Легкість реалізації, Врахування типу трафіку Відмовостікість Легкість інсталювання на контролер		

Продовження таблиці 5.19

Рівні товару	**Сутність та складові**		
II. Товар у реальному виконанні	**Властивості/характеристики**	**/Нм**	**Вр/ Тх /Тл/Е/Ор**
	Легкість реалізації, Врахування типу трафіку Відмовостікість Легкість інсталювання на контролер		
	Програма пройшла тестування, та задовольняє всім вимогам, які повинні бути присутні для виходу на ринок		
	Веб-застосування для перевірки роботи алгоритму конструювання трафіку		
	Марка: NetSdn		

Захист алгоритму буде виконаний за допомогою посвідчення про інтелектуальну власність. Далі необхідно розробити систему збуту та концепцію маркетингових комунікацій (табл. 5.20, 5.21).

Таблиця 5.20 – Формування системи збуту

№ п/п	**Специфіка закупівельної поведінки цільових клієнтів**	**Функції збуту, які має виконувати постачальник товару**	**Глибина каналу збуту**	**Оптимальна система збуту**
1	Покупка ліцензії на використання протоколу	Надіслати продукт після оплати ліцензії	Одноріневна система збуту	Виробник-споживач

Таблиця 5.21 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	**Канали комунікацій, якими користуються цільові клієнти**	**Ключові позиції, обрані для позиціонування**	**Завдання рекламного повідомлення**	**Концепція рекламного звернення**
Клієнт вимагає, щоб продукт був сумісним з іншими протоколами	Електронна пошта, телефон	Конструювання трафіку у програмно-конфігурованих мережах	Здійснення розподілу трафіку при якому втрата пакетів менша	Перегляд результатів розподілу трафіку по маршрутам

Висновки до розділу

У даному розділі був проведений перший етап розробки стартап-проекту, була визначена концепція проекту:

- описана ідея;
- проаналізовані ринкові можливості реалізації ідеї;
- визначені загальні напрямки використання;
- відмінність від конкурентів;
- визначені перспективи ринкової реалізації;
- стратегія ринкового впровадження маркетингової стратегії.

Визначено систему узгоджених рішень щодо ринкової поведінки стартап-проекту. Виявлено, що проект цілком може бути запропонований на ринку, та можна сподіватися, що для нього знайдуться споживачі.

ВИСНОВКИ

У результаті виконання магістерської дисертації був розроблений алгоритм конструювання трафіку для програмно-конфігурованих мереж, для моделювання і оцінки ефективності якого було розроблене програмне забезпечення.

На основі відомих технологій конструювання трафіку для програмно-конфігурованих мереж були визначені основні фактори, що впливають на безпечну передачу інформації, визначені основні переваги та недоліки сучасних рішень.

В результаті було розроблено та досліджено алгоритм конструювання трафіку в програмно-конфігурованих мережах, який базується на основі узагальненої метрики маршрутів та станом мережі. Запропонований спосіб відрізняється від існуючих меншою швидкістю на ремаршрутизацію трафіку та дозволяє зменшити відсоток втрати пакетів в програмно-конфігурованій мережі.

Розроблений метод здійснює конструювання трафіку з урахуванням завантаженості на маршрутах та інші параметри, що дозволяє підвищити надійність передачі даних та зменшити кількість втрат пакетів, і відповідно дозволяє підвищити рівень якості обслуговування в програмно-конфігурованих мережах.

Для перевірки його ефективності були проведені чисельні експерименти, у яких запропонований метод порівнювався з найбільш популярним на сьогодні рішенням щодо розподілу трафіку, результати експериментів показали, зо запропонований алгоритм досягає безпечнішої передачі даних, а також уникає перенавантажень.

Таким чином були виконані усі заявлені завдання та досягнуто мету дослідження, а саме підвищення рівня надійності конструювання трафіку в програмно-конфігурованих мережах за рахунок використання особливим способом організованого розподілу даних між маршрутами.

ПЕРЕЛІК ПОСИЛАНЬ

3. SDN basics: Understanding centralized control and programmability [Електронний ресурс] / / TechTarget. - Режим доступу: https://searchsdn.techtarget.com/tip/SDN-basics-Understanding-centralized-control-and-programmability.
4. Goransson P. Software Defined Networks: A Comprehensive Approach. / P. Goransson. — MA.: Elsevier Inc, 2014. — С. 215.
5. Ajay Guleria Traffic Engineering in Software Defined Networks: A Survey / T. Journal of Telecommunications and Information Technology, 2016. — С.3-14
6. Azodolmolky S. Software Defined Networking with OpenFlow / S. Azodolmolky – UK.: Packt Publishing, 2013. – 148 с.
7. OpenFlow Switch: What Is It and How Does it Work? [Електронний ресурс] Режим доступу: http://www.cables-solutions.com/whats-openflow-switch-how-it-works.html
8. Greenberg, A.; Hamilton, J.; Maltz, D.A.; Patel, P. The cost of a cloud: Research problems in data center networks. ACM SIGCOMM Comput. Commun. Rev. 2008, 39, 68–73.
9. Cheng, J.; Cheng, J.; Zhou, M.; Liu, F.; Gao, S.; Liu, C. Routing in Internet of Vehicles: A Review. IEEE Trans. Intell. Transp. Syst. 2015, 16, 2339–2352.
10. Ghemawat, S.; Gobioff, H.; Leung, S.T. The Google file system. In Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA, 19–22 October 2003; pp. 29–43.
11. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The Hadoop Distributed File System. In Proceedings of the IEEE 26th Symposium on MASS Storage Systems and Technologies, Incline Village, NV, USA, 3–7 May 2010; pp.1–10.
12. Dean, J.; Ghemawat, S. MapReduce: Simplified Data Processing on Large Clusters. In Proceedings of the6th Conference on Symposium on Opearting Systems Design & Implementation, San Francisco, CA, USA,6–8 December 2004; pp. 137–150.
13. Dong Sun, Kaixin Zhao, Yaming Fang and Jie Cui: Dynamic Traffic Scheduling and Congestion Control across Data Centers Based on SDN: July 2018.

14. Lei Cai, Dianjun Chen and Luyong Zhang A Strategy of Dynamic Routing Based on SDN, IEEE, 2017: 373-378.
15. Zhang, H., Guo, X., Yan, J., Liu, B., Shuai, Q. (2014). SDN-based ECMP algorithm for data center networks. 2014 IEEE Computers, Communications and IT Applications Conference. doi: https://doi.org/10.1109/comcomap.2014.7017162
16. Dinh, K. T., Kukliński, S., Kujawa, W., Ulaski, M. (2016). MSDN-TE: Multipath Based Traffic Engineering for SDN. Intelligent Information and Database Systems, 630–639. doi: https://doi.org/10.1007/978-3-662-49390-8_61
17. Rodríguez F.J., Fernandez S., Sanz I, et al. Distributed Approach for Smart Grids Reconfiguration Based on the OSPF Routing Protocol [J]. IEEE Transactions on Industrial Informatics, 2016, 12(2): 864-871.
18. Broumi S., Talea M., Bakali A., et al. Application of Dijkstra algorithm for solving interval valued neutrosophic shortest path problem, Computational Intelligence. IEEE, 2017: 1-6.
19. M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks”. 2010 7th USENIX Symp. on Netw. Syst. Design & Implemen. NSDI’10, San Jose, CA, USA. 28-30 April 2010. pp. 19–19.
20. A. R. Curtis, W. Kim, and P. Yalagandula. “Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection”, 30th IEEE Int. Conf. Comp. Commun. IEEE INFOCOM 2011, Shanghai, China, 10-15 April 2011, pp. 162–163.
21. A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, “DevoFlow: Scaling Flow Management for High-Performance Networks”. ACM SIGCOMM Comp. Commun. Rev., 2011. No41(4). PP.254-265.
22. T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE: Fine grained traffic engineering for data centers”, in Proc. 7th Conf. On Emerg. Networking Experim. & Technol. Co-NEXT’11, Tokyo, Japan, 2011, pp. 8.

23. R. Trestian, G.-M. Muntean, and K. Katrinis, “MiceTrap: Scalable traffic engineering of datacenter mice flows using OpenFlow”. in IFIP/IEEE Int. Symp. on Integr. Netw. Managem. Ghent, Belgium, 2013, pp. 904–907.
24. Panneerselvam Senthilkumar, Sockalingam Narayanan Literature Review of Single Machine Scheduling Problem with Uniform Parallel Machines, Intelligent Information Management, 2010, 2, 457-474
25. Y. Azar and L. Epstein, “On-Line Machine Covering,” Proceedings of 5th ESA conference, Graz, 1997, pp. 23-36.
26. Y. Azar and L. Epstein, “On-Line Machine Covering, Journal of Scheduling,” Vol. 1, No. 2, 1998(a), pp. 67-77.
27. C# [Електронний ресурс] URL: https://uk.wikipedia.org/wiki/C_Sharp
28. .NET Framework [Електронний ресурс]. URL: https://uk.wikipedia.org/wiki/.NET_Framework
29. Труба О.М. Підвищення надійності конструювання трафіку в програмно-конфігурованій мережі / О.М. Труба, А.В. Коган // Матеріали III всеукраїнської науково-практичної конференції молодих вчених та студентів «Інформаційні системи та технології управління» (ІСТУ-2019) – м. Київ.: НТУУ «КПІ ім. Ігоря Сікорського», 20-22 листопада 2019 р. – С. 39-42.
30. Труба О.М., Ю.О. Кулаков, А.В. Коган, М.О. Сперкач «Конструювання трафіку в бездротових програмно-конфігурованих мережах». Східно-Європейський журнал передових технологій. №6. – 2019 р.
31. Podzirey Ya. Method of forming multi-path disjoint channels in a program-configured network / Podzirey Ya., Kohan A. // Bulletin of NTUU "KPI". Informatics, management and computer engineering.– К. – 2017. – Vol. 66. – Pp. 137-141.
32. Yasir Ali Matnee, Chasib Hasan Abooddy, Zainab Qahtan Mohammed / Analyzing Methods and Opportunities in Software-Defined (SDN) Networks for Data Traffic Optimizations // International Journal on Recent and Innovation

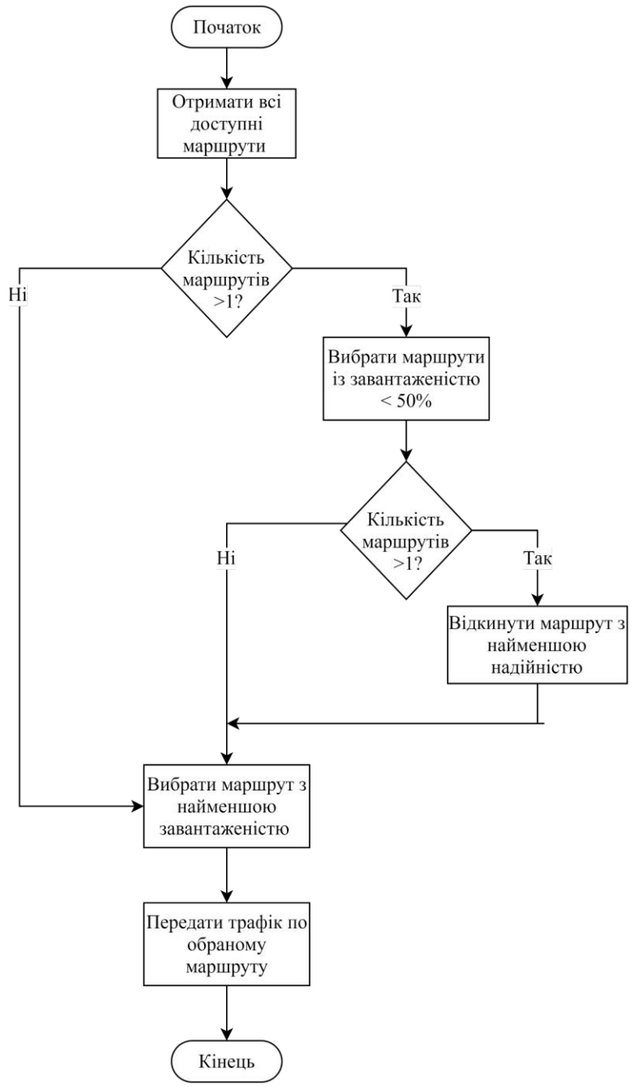
Trends in Computing and Communication. – 2018. – №6. – С.75 – 82. – Режим доступу до ресурсу: http://www.ijritcc.org

33. S. Agarwal, M. Kodialam, and T. V. Lakshman / Traffic engineering in software defined networks // IEEE INFOCOM. – 2013. – С. 2211-2219.
34. 4. As'ad Mahmoud As’ad Al-Naser / Streaming algorithm for multi-path secure routing in mobile networks // IJCSI International Journal of Computer. –2014. – №4. – С.112-114.
35. 5. As'ad Mahmoud As'ad Alnaser / A Method of Forming the Optimal Set of Disjoint Path in Computer Networks // Journal of Applied Computer Science & Mathematics. – 2017. – №23. С.9 -12.
36. Yurii Kulakov, Sergii Kopychko, Victoria Gromova / Organization of Network Data Centers Based on Software-Defined Networking // International Conference on Computer Science, Engineering and Education Applications ICCSEEA. – 2018. – С.447-455. – Режим доступу до ресурсу: https://link.springer.com/book/10.1007/978-3-319-91008-6
37. Sahel Sahha / Adaptive and Reliable Multipath Provisioning for Media Transfer in SDN-based Overlay Networks // – 2017. – С.11-12

ДОДАТОК А

Графічний матеріал

Блок-схема алгоритму розподілу трафіку



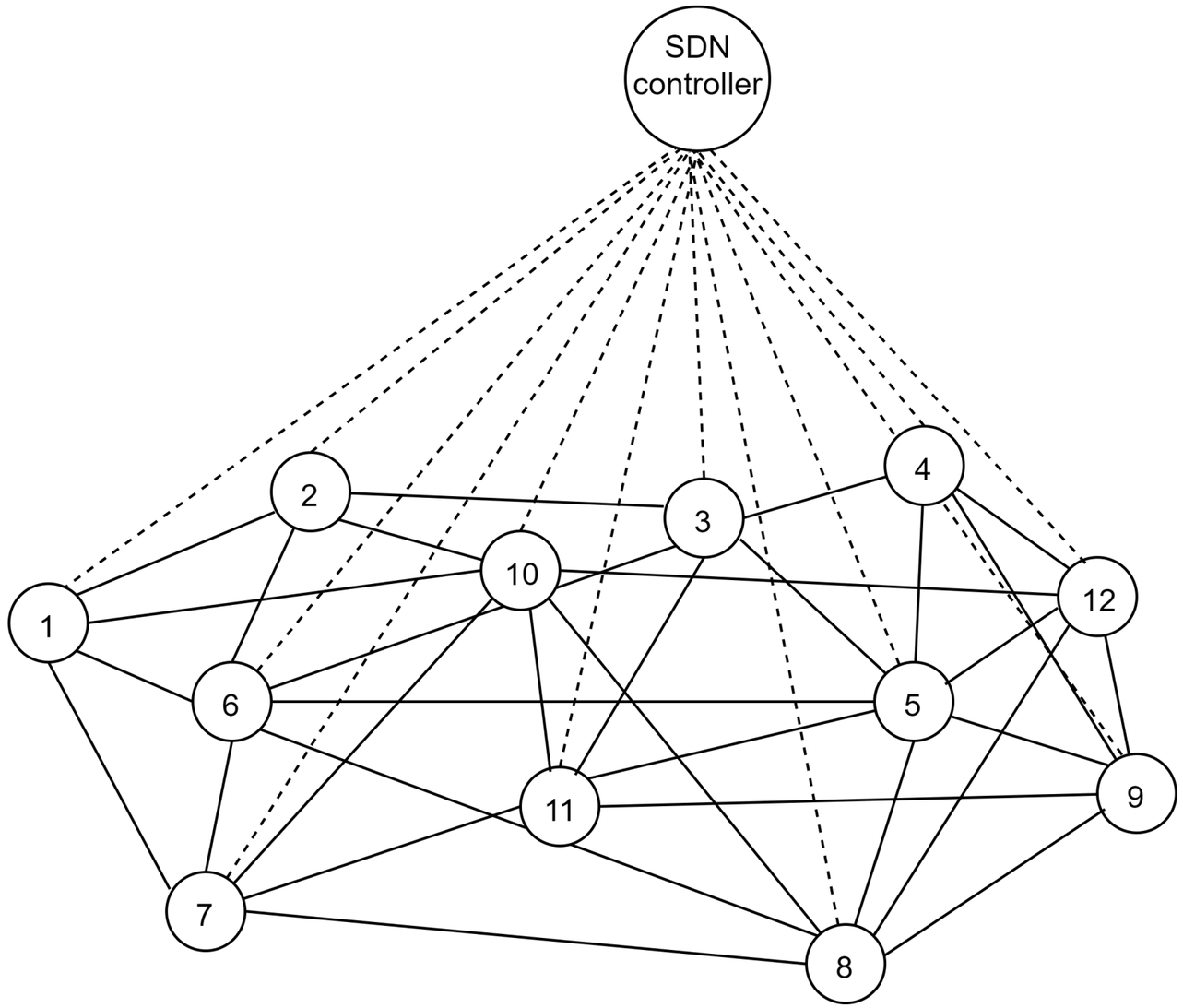
Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Представлення мережі у вигляді графу



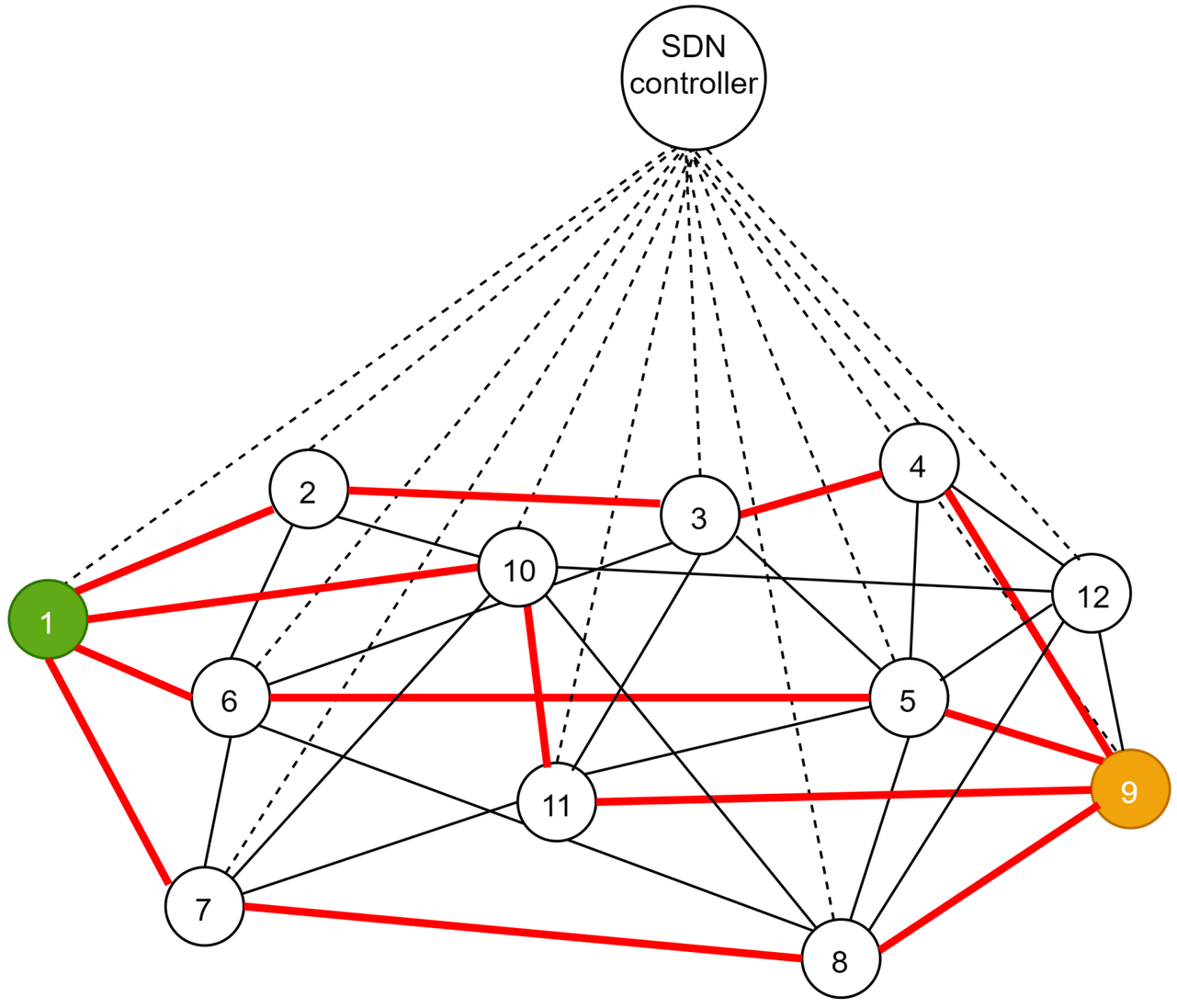
Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Структура мережі з множиною шляхів



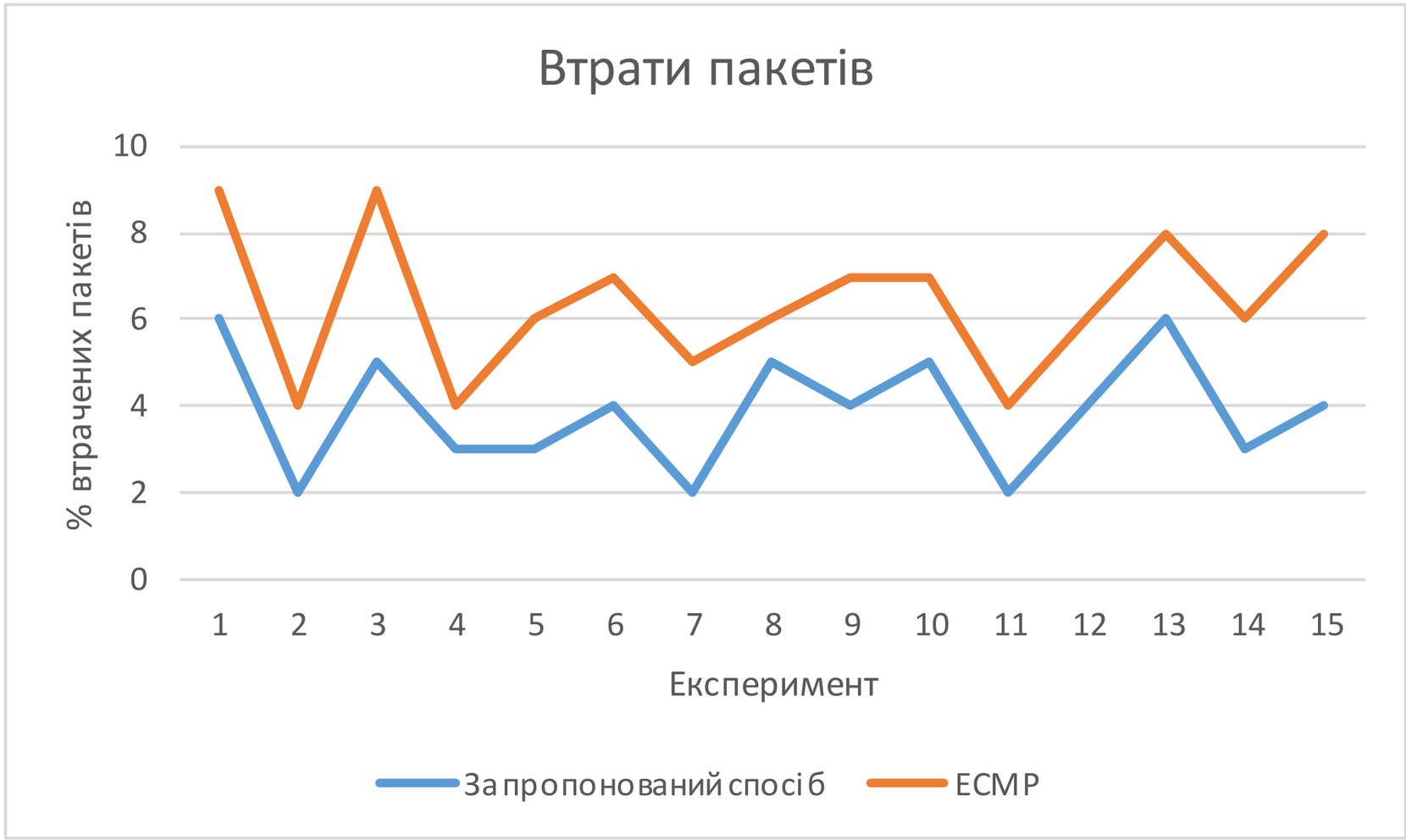
Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Результати експериментальних досліджень



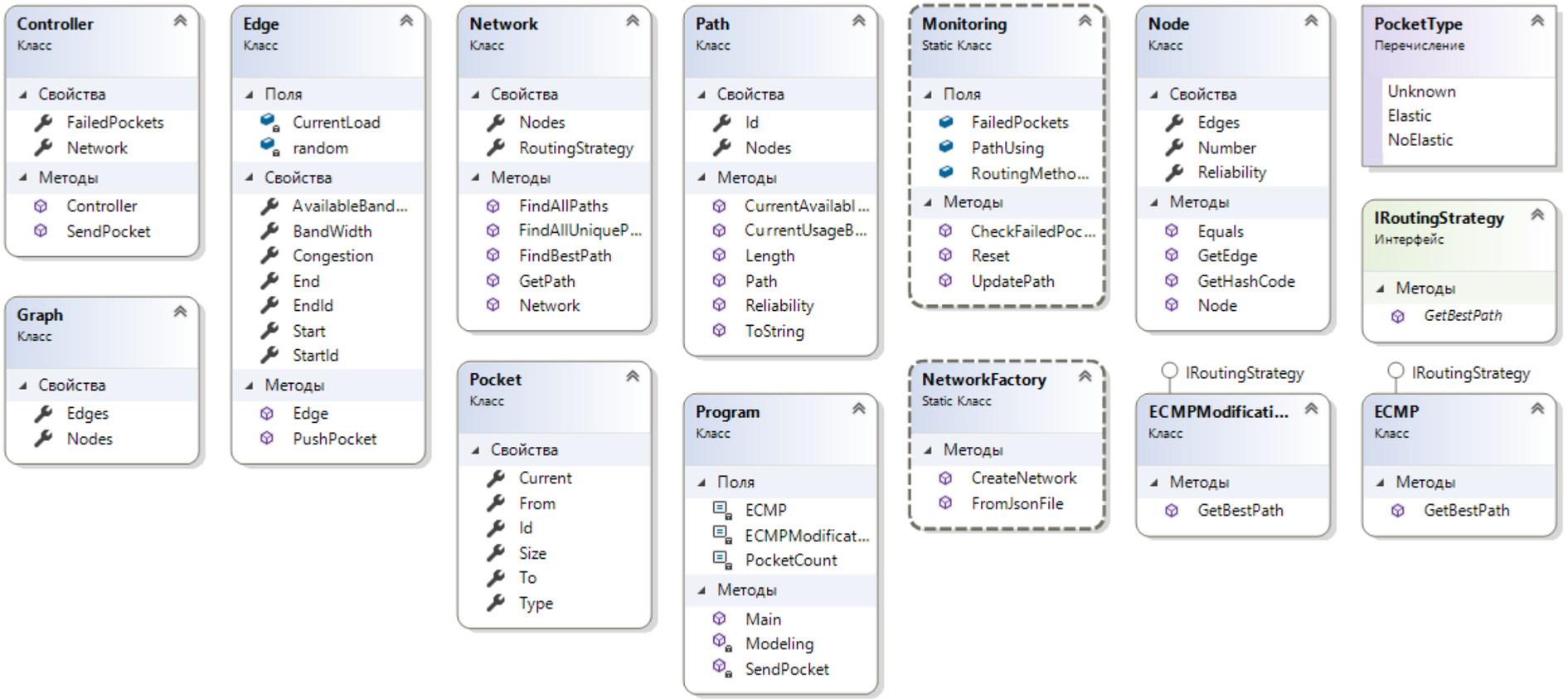
Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Структурна схема класів



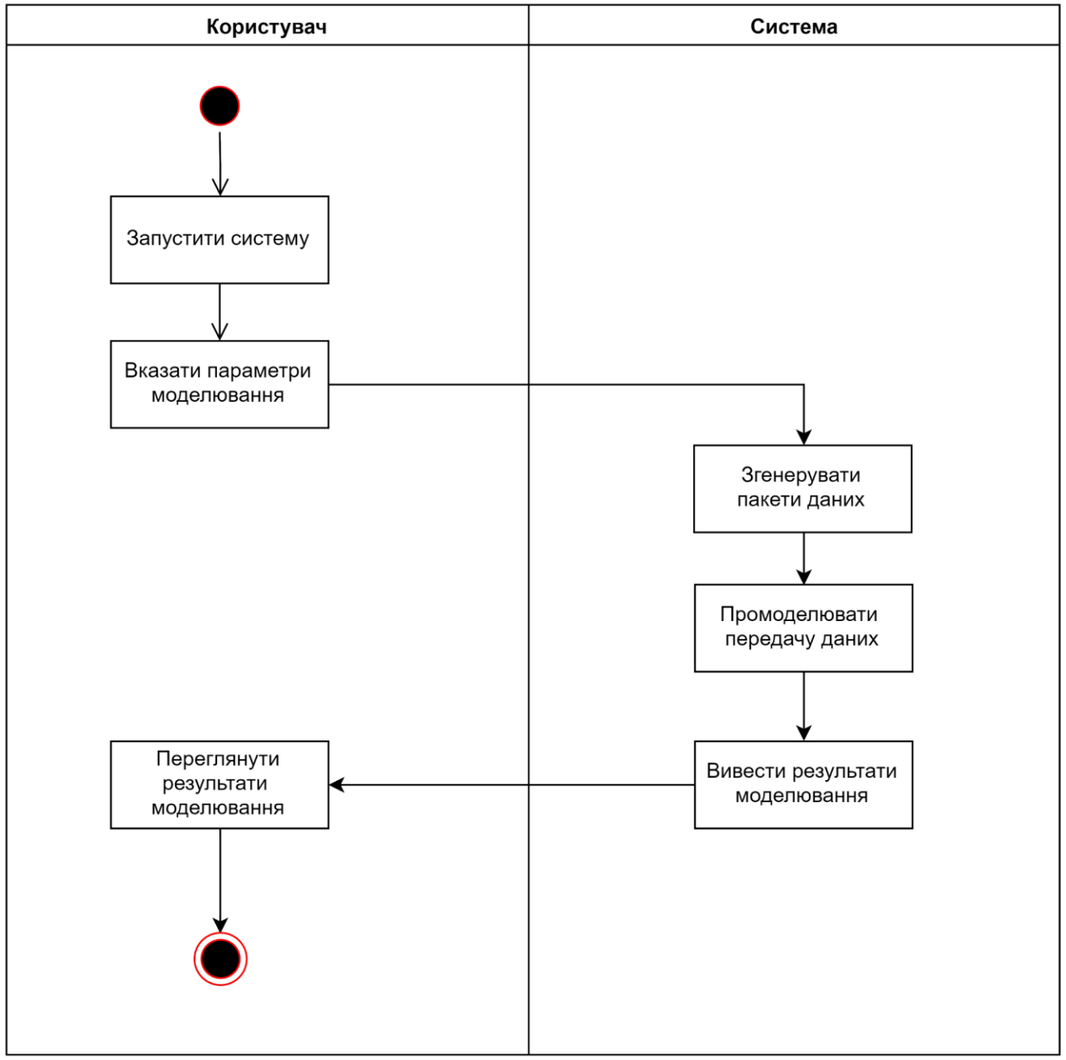
Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Структурна схема діяльності процесу моделювання



Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.

Копії екранних форм

```
Start modeling for ECMP method!
Pocket №:0 send by path: [1->6->5->9] with current load:0
Pocket №:1 send by path: [1->10->11->9] with current load:0
Pocket №:2 send by path: [1->7->8->9] with current load:0
FAIL! Pocket with id: 69b023f3-8184-42c3-9416-520e61c2d3fa not sent from node #7!
Pocket №:3 send by path: [1->7->8->9] with current load:0
Pocket №:4 send by path: [1->7->8->9] with current load:0,03
Pocket №:5 send by path: [1->6->5->9] with current load:0,07
Pocket №:6 send by path: [1->2->3->4->9] with current load:0
Pocket №:7 send by path: [1->2->3->4->9] with current load:0,02
Pocket №:8 send by path: [1->10->11->9] with current load:0,12
Pocket №:9 send by path: [1->7->8->9] with current load:0,15
Pocket №:10 send by path: [1->6->5->9] with current load:0,17
Pocket №:11 send by path: [1->2->3->4->9] with current load:0,12
Pocket №:12 send by path: [1->2->3->4->9] with current load:0,2
FAIL! Pocket with id: 177f0414-3cd1-461d-899d-d80a12578b8a not sent from node #2!
Pocket №:13 send by path: [1->6->5->9] with current load:0,16
Pocket №:14 send by path: [1->10->11->9] with current load:0,1
Pocket №:15 send by path: [1->10->11->9] with current load:0,15
Pocket №:16 send by path: [1->7->8->9] with current load:0,08
Pocket №:17 send by path: [1->6->5->9] with current load:0,22
Pocket №:18 send by path: [1->6->5->9] with current load:0,17
Pocket №:19 send by path: [1->7->8->9] with current load:0,19
Pocket №:20 send by path: [1->10->11->9] with current load:0,17
Pocket №:21 send by path: [1->7->8->9] with current load:0,24
Pocket №:22 send by path: [1->6->5->9] with current load:0,24
Pocket №:23 send by path: [1->2->3->4->9] with current load:0,18
Pocket №:24 send by path: [1->2->3->4->9] with current load:0,16
Pocket №:25 send by path: [1->2->3->4->9] with current load:0,21
Pocket №:26 send by path: [1->10->11->9] with current load:0,22
FAIL! Pocket with id: 4fcaaecb-6be4-47cf-8499-0bb3a3606d02 not sent from node #3!
Pocket №:27 send by path: [1->2->3->4->9] with current load:0,1
Pocket №:28 send by path: [1->10->11->9] with current load:0,13
Pocket №:29 send by path: [1->10->11->9] with current load:0,19
Pocket №:30 send by path: [1->7->8->9] with current load:0,15
Pocket №:31 send by path: [1->6->5->9] with current load:0,11
Pocket №:32 send by path: [1->6->5->9] with current load:0,17
Pocket №:33 send by path: [1->10->11->9] with current load:0,13
Pocket №:34 send by path: [1->7->8->9] with current load:0,12
Pocket №:35 send by path: [1->6->5->9] with current load:0,11
Pocket №:36 send by path: [1->6->5->9] with current load:0,2
Pocket №:37 send by path: [1->7->8->9] with current load:0,23
Pocket №:38 send by path: [1->10->11->9] with current load:0,22
Pocket №:39 send by path: [1->6->5->9] with current load:0,27
```

```
Start modeling for ECMP with modification method
Pocket №:0 send by path: [1->6->5->9] with current load:0
Pocket №:1 send by path: [1->10->11->9] with current load:0
Pocket №:2 send by path: [1->7->8->9] with current load:0
Pocket №:3 send by path: [1->7->8->9] with current load:0,05
Pocket №:4 send by path: [1->6->5->9] with current load:0,11
Pocket №:5 send by path: [1->10->11->9] with current load:0,12
Pocket №:6 send by path: [1->7->8->9] with current load:0,13
Pocket №:7 send by path: [1->10->11->9] with current load:0,21
Pocket №:8 send by path: [1->6->5->9] with current load:0,23
Pocket №:9 send by path: [1->7->8->9] with current load:0,22
Pocket №:10 send by path: [1->6->5->9] with current load:0,26
FAIL! Pocket with id: ff967ee9-16e8-4a9b-8359-782f35904cb4 not sent from node #8!
Pocket №:11 send by path: [1->7->8->9] with current load:0,21
Pocket №:12 send by path: [1->6->5->9] with current load:0,22
Pocket №:13 send by path: [1->10->11->9] with current load:0,16
Pocket №:14 send by path: [1->7->8->9] with current load:0,2
Pocket №:15 send by path: [1->7->8->9] with current load:0,16
Pocket №:16 send by path: [1->6->5->9] with current load:0,13
Pocket №:17 send by path: [1->6->5->9] with current load:0,24
Pocket №:18 send by path: [1->10->11->9] with current load:0,17
Pocket №:19 send by path: [1->10->11->9] with current load:0,24
Pocket №:20 send by path: [1->7->8->9] with current load:0,28
Pocket №:21 send by path: [1->6->5->9] with current load:0,3
Pocket №:22 send by path: [1->10->11->9] with current load:0,29
Pocket №:23 send by path: [1->7->8->9] with current load:0,26
Pocket №:24 send by path: [1->6->5->9] with current load:0,29
Pocket №:25 send by path: [1->7->8->9] with current load:0,29
Pocket №:26 send by path: [1->10->11->9] with current load:0,24
Pocket №:27 send by path: [1->7->8->9] with current load:0,17
Pocket №:28 send by path: [1->10->11->9] with current load:0,28
Pocket №:29 send by path: [1->6->5->9] with current load:0,16
Pocket №:30 send by path: [1->6->5->9] with current load:0,24
Pocket №:31 send by path: [1->10->11->9] with current load:0,25
Pocket №:32 send by path: [1->10->11->9] with current load:0,3
Pocket №:33 send by path: [1->6->5->9] with current load:0,28
Pocket №:34 send by path: [1->7->8->9] with current load:0,19
Pocket №:35 send by path: [1->7->8->9] with current load:0,26
Pocket №:36 send by path: [1->6->5->9] with current load:0,34
Pocket №:37 send by path: [1->6->5->9] with current load:0,33
Pocket №:38 send by path: [1->10->11->9] with current load:0,29
Pocket №:39 send by path: [1->10->11->9] with current load:0,32
```

```
-----RESULTS-----
ECMP method
Problems with pockets happens: 5 times
Path: [1->6->5->9, 27], pocket send: 27, percent: 0,27%
Path: [1->10->11->9, 25], pocket send: 25, percent: 0,25%
Path: [1->7->8->9, 28], pocket send: 28, percent: 0,28%
Path: [1->2->3->4->9, 20], pocket send: 20, percent: 0,2%
ECMP with modification method
Problems with pockets happens: 1 times
Path: [1->6->5->9, 34], pocket send: 34, percent: 0,34%
Path: [1->10->11->9, 34], pocket send: 34, percent: 0,34%
Path: [1->7->8->9, 32], pocket send: 32, percent: 0,32%
```

Демонстраційний плакат до магістерської дисертації

на тему «Підвищення ефективності маршрутизації трафіку в програмно-конфігурованій мережі»

Виконав студент гр. ІС-82мп Труба О.М.

Керівник Коган А.В.