Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод управління трафіком SDN-мережі

(тема)

Виконав:
студент II курсу, групи СПм-18-1
Солтан Д.Д.
(прізвище, ініціали)

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Системне програмування
(повна назва освітньої програми)

Керівник: доц. Філімончук Т.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп'ютерна інженерія
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Солтану Денису Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод управління трафіком SDN-мережі

затверджена наказом по університету від " 01 " листопада 2019 р. № 1615Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 2019 р.

3. Вхідні дані до роботи
Управління трафіком, SDN-мережа, алгоритм рою сальп, алгоритм летючої миші

4. Перелік питань, що потрібно опрацювати в роботі
1 Теоретичні аспекти архітектури SDN-мереж.
2 Аналіз існуючих методів і алгоритмів управління трафіком SDN-мереж.
3 Практичне застосування алгоритму рою сальп та алгоритму летючої миші для управління трафіком SDN-мережі.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Демонстраційні матеріали. Плакати – 12 арк. ф. А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд архітектури та методів управління трафіком SDN-мереж	04.11.19-14.11.19	
2	Вибір та обгрунтування методики дослідження	15.11.19-21.11.19	
3	Вибір інструментальних засобів	22.11.19-27.11.19	
4	Проведення експериментів	28.11.19-05.12.19	
5	Оформлення матеріалів атестаційної роботи	06.12.19-11.12.19	
6	Подання атестаційної роботи керівникові та її попередній захист	12.12.19-13.12.19	
7	Подання атестаційної роботи на рецензування	14.12.19-18.12.19	

Дата видачі завдання 4 листопада 2019 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Філімончук Т.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 64 с., 13 рис., 8 табл., 2 дод., 27 джерел.

SDN, АЛГОРИТМ ЛЕТЮЧОЇ МИШІ, ETHERNET, АЛГОРИТМ РОЮ САЛЬП, МУЛЬТИ КОНТРОЛЕР.

Метою атестаційної роботи є дослідження та систематизація теоретичних даних щодо управління трафіком SDN-мереж, існуючих проблем даної технології та моделювання можливих шляхів вирішення цих проблем.

Було досліджено принципи функціонування SDN-мереж, їх архітектуру. Було проведено порівняльний аналіз SDN-мереж та традиційних мереж. Було досліджено галузі застосування галузі застосування SDN-мереж, запропоновано можливі шляхи підвищення безпеки технології Ethernet за допомогою технології SDN.

Також наведено модель мульти контролера для мережі SDN. Дана модель може бути представлена у вигляді системи з чергою. Було наведено алгоритм рою сальп для мережі SDN. Для перевірки впливу динамічних змін запитів потоку на кількість активних контролерів розглядаються чотири випадки з різними діапазонами швидкості потоку кожного пристрою пересилання.

Враховуючи здатність мережі SDN збирати інформацію, а також вивчати ідею розподілу міток та резервування ресурсів, було запропоновано використання алгоритму летючої миші для SDN.

ABSTRACT

Master’s thesis: 64 pages, 13 figures, 8 tables, 2 appendices, 27 sources.

SDN, BAT ALGORITHM, ETHERNET, SALP SWARM ALGORITHM, MULTICONTROLLER.

The major goal of this thesis is the research and systematization of theoretical data on traffic management of SDN networks, existing problems of this technology and modeling of possible ways of solving these problems.

The principles of functioning of SDN networks and their architecture were investigated. A comparative analysis of SDN networks and traditional networks was conducted. The field of application of the field of application of SDN networks was investigated, and possible ways to improve the security of Ethernet technology using SDN technology were proposed.

A multi-controller model for the SDN network is also provided. This model can be represented as a queue system. An algorithm for swarms of salps for the SDN network was provided. To test the impact of dynamic changes in flow requests on the number of active controllers, four cases with different flow rate ranges of each forwarding device are considered.

Given the ability of the SDN network to collect information, as well as to explore the idea of label allocation and resource reservation, it was proposed to use the SDN bat algorithm.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

SDN – Software determined network (програмно-конфігурована мережа)

API – Application programming interface (інтерфейс програмного програмування)

IP – Internet protocol (ідентифікатор мережевого рівня)

WAN – Wide aria network (глобальна комп

ACL – Access control list (список управління доступом)

NFV – Network function virtualization (технологія віртуафлізації фізичних мережевих елементів)

ЦОД – центр обробки даних

IPS – Image packaging system (система управління пакетами)

VLAN – Virtual local area network (віртуальна локальна мережа)

DHCP – Dynamic host configuration protocol (протокол динамічної конфігурації вузла)

SSL – Secure sockets layer (рівень захищених сокетів)

CPU – Central processing unit (центральний процесор)

ВСТУП

На сучасному етапі розвитку мережевих технологій існує необхідність у відділенні рівня управління від рівня контролю. Така можливість реалізована у мережевій технології SDN. Актуальність дослідження даної теми обумовлена тим, що дана технологія є інноваційною, але не є широко розповсюдженою. Принципи, на яких базується SDN, були закладені та почали використовуватися у телефонних мережах задовго до того, як ця архітектура була переведена до мереж передачі даних. Головне питання полягає в тому, щоб централізувати весь «інтелект» мережі в єдиний контролер замість того, щоб розділити його між усіма комутаторами та/або маршрутизаторами.

Ця сегрегація між апаратним забезпеченням (площиною даних) та програмним забезпеченням (площиною управління) призначена для підвищення ефективності за рахунок скорочення часу роботи в проміжних вузлах з використанням різних протоколів потоку.

Даний тип мережі є новою технологією, призначеною замінити фізичний дизайн мережі мережевою інфраструктурою, керованою програмним забезпеченням. Зазвичай це виявляється практичним, порівняно економічно ефективним та динамічним рішенням.

SDN-мережа застосовується у різноманітних галузях: програми для спільної роботи, конвергентне сховище, мережевий обмін, організація послуг мобільної мережі, масштабовані мережі центрів обробки даних.

Метою роботи є дослідження та систематизація теоретичних даних щодо управління трафіком SDN-мереж, існуючих проблем даної технології та моделювання можливих шляхів вирішення цих проблем.

Для досягнення поставленої мети в роботі сформульовано та вирішено наступні задачі:

- дослідити архітектуру SDN та галузі застосування;

- порівняти технологію SDN та традиційні мережі;

- дослідити алгоритми та методи управління трафіком мережі SDN;

- систематизувати отримані дані;

- навести результати практичного використання досліджених алгоритмів та методів для управління трафіком мережі SDN.

Отримані результати дослідження можуть бути застосовані при оптимізації локальних мереж, застосуванні хмарних обчислень, технології Big Data тощо.

Результати досліджень опубліковано у тезах доповідей на форумі.

Магістерська робота складається з вступу, трьох розділів, висновків та переліку джерел посилання.

1 ТЕОРЕТИЧНІ АСПЕКТИ АРХІТЕКТУРИ SDN-МЕРЕЖІ

1.1 Принципи функціонування SDN

Software-defined network (SDN), або програмно-конфігурована мережа – це підхід до управління мережею, що передбачає розділення рівня контролю мережі та рівня передачі даних. Принципи, на яких базується SDN, були закладені та почали використовуватися у телефонних мережах задовго до того, як ця архітектура була переведена до мереж передачі даних [1].

Робоча група Internet Engineering почала розглядати різні засоби відключення функцій управління та переадресації у запропонованому стандарті інтерфейсу, опублікованому в 2004 році, «Розділення елементів пересилання та управління». Також було запропоновано супутню архітектуру SoftRouter. Додаткові ранні стандарти IETF, які здійснювали відділення контролю від даних, включають Linux Netlink як протокол IP-служб та архітектуру Path Computation Element (PCE).

Ці перші спроби не змогли здобути визнання у науковій спільноті того часу з двох причин. По-перше, дослідники з Інтернет-спільноти вважали відділення контролю від даних ризикованим, особливо через потенціальний вихід з ладу в площині управління. По-друге, постачальники були стурбовані тим, що створення стандартних інтерфейсів програмування прикладного забезпечення (API) між площинами управління та даними призведе до посилення конкуренції [15].

Використання програмного забезпечення з відкритим кодом у архітектурах з розділеними площинами управління та даних простежується до проекту Етан у відділі комп'ютерних наук Стенфорда. Простий дизайн комутатора Етана привів до створення протоколу OpenFlow. API для OpenFlow був вперше створений у 2008 році.

Робота над OpenFlow тривала в Стенфорді, в тому числі зі створенням

тестових панелей для оцінки використання протоколу в єдиній мережі кампусу, а також по всій WAN в якості основи для підключення декількох кампусів. В академічних умовах існувало декілька дослідницьких та виробничих мереж на основі перемикачів OpenFlow від NEC та Hewlett-Packard.

Поза межами академій, перші розгортання SDN були зроблені Nicira в 2010 році для контролю OVS від Onix, спільно розробленого з NTT та Google. Помітним розгортанням стало розгортання Google B4 у 2012 році. Пізніше Google одночасно визнав їх перше відкриття OpenFlow разом з Onix у своїх Центрах обробки даних. Ще одне відоме велике розгортання знаходиться в China Mobile [9].

Фонд «Відкриті мережі» був заснований у 2011 році для просування SDN та OpenFlow.

1.2 Теоретичні засади архітектури SDN

Програмно-визначені мережі, також відомі як SDN, є новою парадигмою в мережі управління. Головне питання полягає в тому, щоб централізувати весь «інтелект» мережі в єдиний контролер замість того, щоб розділити його між усіма комутаторами та/або маршрутизаторами [13].

Ця сегрегація між апаратним забезпеченням (площиною даних) та програмним забезпеченням (площиною управління) призначена для підвищення ефективності за рахунок скорочення часу роботи в проміжних вузлах з використанням різних протоколів потоку, зазвичай OpenFlow.

Створений у 2008 році в Стенфордському університеті, OpenFlow - це протокол зв'язку, який дозволяє адміністрування мережі від централізованого контролера. Це основний протокол, що використовується для зв'язку між площиною управління та площиною даних [17]. Це дозволяє визначити шляхи, якими будуть слідувати пакети всередині мережі, використовуючи прості правила. Ці правила – це дії, які необхідно здійснити над пакетом,

якщо виконуються певні умови. Типові умови можуть відповідати конкретному порту входу, джерелу або призначеній MAC-адресі, протоколу тощо [16].

Ці правила, що називаються потоками, контролер висуває в кожен перемикач. Отримуючи пакет, що не відповідає потоку, перемикач повертає його до контролеру.

Отже, OpenFlow – це протокол, який визначає функціонування SDN. OpenFlow є першим стандартизованим відкритим інтерфейсом, які відповідають за взаємодію між рівнем управління і рівнем передачі даних. OpenFlow забезпечує доступ, обмін інформацією та доставку керуючих команд елементів мережевої інфраструктури. 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 [12].

Архітектура SDN-мережі представлена у загальному вигляді на рисунку 1.1.

Рисунок 1.1 – Архітектура SDN-мережі у загальному вигляді

Логічна модель SDN-мережі складається з наступних компонентів:

- додатки SDN: додатки, які предоставляють кінцевим користувачам бажані сервиси. Додатки SDN містять ряд вимог до стану і поведінки мережевої інфраструктури;

- контролер SDN виступає єдиною централізованою точкою управління, яка взаємодіє з рівнем додатків за допомгогю відкритого інтерфейсу API, а також виконує моніторинг і управління фізичними приладами мережі за допомогою відкритого інтерфейсу – протоколу OpenFlow. Контролер складається з декількох модулів або рівнів, кожен модуль відповідає за ряд необхідних функціональних можливостей;

- OpenFlow комутатори забезпечують безпосередню взаємодію продуктів всієї мережевої інфраструктури з рівнем управління. Комутатор містить одну або кілька таблиць переадресації (flowtable), які містять всі дані про потоках інформації, що передається. Записи у таблицях переадресації містять набір полів з інформацією щодо кожного пакету (номер вхідного і вихідного порту, пріоритет переданих даних, лічильник і види дій, які необхідно виповнити після обробки пакета (перенаправлення, модифікація або скидання);

- FlowVisor є відповідальним за розподіл керуючої інформації між потоками даних. За своєю природою FlowVisor – це прозорий проксі-сервер між коммутаторами і контролером. При цьому FlowVisor визначає, які безлічі потоків відносяться до тієї чи іншої мережі (комутатора) і, отже, передають керуючу інформацію певного конролера. FlowVisor забезпечує віртуалізаціюмпотоків керуючих пакетів в окремі зрізи мережі (slices), кожен з таких потоків має свою логіку управління і передачі;

- компоненти управління і адміністрування – це набір статичних даних, які включають зовнішні завдання: координацію політик і правил, встановлених при проектуванні бізнес-моделі архітектури SDN, початкова конфігурація обладнання і правила розподіл мережевих ресурсів [7].

Компоненти архітектури SDN-мережі наведено на рисунку 1.2 [4].

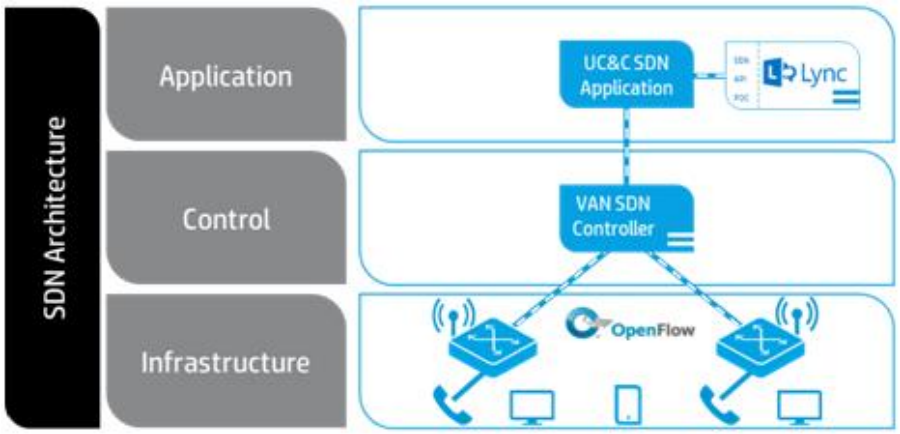


Рисунок 1.2 – Компоненти архітектури SDN-мережі

Отже, мережева технологія SDN відрізняється за свою сутністю від звичайного підходу до мережі. В таблиці 1.1 наведено основні відмінності SDN від традиційного підходу.

Таблиця 1.1 – Порівняння підходу SDN зі звичайними мережами

Параметр	Звичайні мережі	SDN
Вигляд мережі	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
Адаптація нових технологій	Відповідно до потреб постачальника	Відповідно до потреб користувача

SDN принципово відрізняється від звичайної мережі шляхом отримання представлення про мережу: у звичайних мережах сама мережа

отримує представлення про додатки, в той час як у програмно-конфігурованій мережі додатки визначають представлення мережі [22].

SDN-мережа прямо описує вимоги до мережі, на відміну від звичайних додатків, що можуть описати вимоги до мережі поступово, в декілька етапів, та потребують обробки спеціалістом. В якості прикладу можна навести перевірку наявності ресурсів та керування політикою конфіденційності для підтримки додатків [2].

Для звичайних мереж, які представлені веб-переглядачами, передачою мультимедіа шляхом потоку, характерні пасивні засоби вираження вираження ряду користувацьких потреб. До таких потреб можна віднести пропускну спроможність, затримку, варіацію затримки або доступність. Заголовки пакетів здатні до шифрування пріотритетних запитів [8]. В такому разі мережевий постачальник. Пакетні заголовки можуть зашифрувати запити пріоритету, але постачальники мережі не завжли довіряють маркуванням користувацького трафіку. Саме цьому певні мережі вдаються до самостійного виводу користувацьких вимог. Але використання аналізу трафіку здатний зашкодити, і в деяких випадкаих унеможливити правильну класифікацію. SDN робить для користувача можливим виразити свої потреби [21].

Також звичайні мережі не можуть надати інформацію та стан мережі додаткам, що їх використовують. SDN-підхід дозволяє додатам відстежувати стан, у якому занходиться мережа, та адекватно реагувати [5].

Виходячи з наведеного вище опису, SDN є комплексною системою взаємодії елементів як логічної, так і фізичної природи, що має складну архітектуру. Упродовж останніх кількох років дослідники та науковці багато уваги приділяли SDN. Даний тип мережі залишається новою технологією, призначеною замінити фізичний дизайн мережі мережевою інфраструктурою, керованою програмним забезпеченням. Зазвичай це виявляється практичним, порівняно економічно ефективним та динамічним рішенням.

1.3 Галузі застосування SDN-мережі

SDN-мережа застосовується у різноманітних галузях: програми для спільної роботи, конвергентне сховище, мережевий обмін, організація послуг мобільної мережі, масштабовані мережі центрів обробки даних.

Постачальник уніфікованих комунікацій Sonus Networks оголосили про новий варіант використання SDN у своїх додатках для відео та спільної роботи. За словами Аашу Вірмані, старшого директора корпоративної стратегії та розвитку бізнесу, це дозволяє краще контролювати мережу та нові сервіси, які потребують динамічних угод на рівні послуг, сказав він. Постачальники можуть забезпечити кращий контроль за якістю обслуговування, не потребуючи розширення потужностей, оскільки вони зможуть забезпечити і видалити мережевий простір виключно на потреби [6].

Edgenet, постачальник послуг передачі даних, використовує SDN для створення програмованої тканини за допомогою центру обробки даних та технологій зберігання даних. Компанія обрала SDN з метою віртуалізації мережі та зробити її досить спритною, щоб не відставати від віртуалізації сервера та зберігання. За допомогою комбінованої технології Edgenet пропонує програмне забезпечення як сервіс та послуги передачі даних. Компанія впровадила екосистему ProgrammableFlow SDN корпорації Америки NEC Corporation, яка включає контролер OpenFlow, а також високошвидкісну фізичну комутацію та віртуальну комутацію, що підтримує середовище Hyper-V та забезпечує можливість мережевої віртуалізації. Додатковою умовою прийняття архітектури SDN була здатність компанії будувати багатошарову мережу на сході та заході, що в кінцевому підсумку дозволить їй реалізувати повністю конвергентну мережу сховища та даних [10].

Мережевий обмін на базі SDN був створений групою організацій в Атланті і дозволяє операторам, науково-дослідним інститутам та підприємствам взаємозв'язувати фізичні мережі за допомогою SDN. Мета

проекту – в кінцевому рахунку надати великим установам і підприємствам можливість уникати використання публічного Інтернету, а натомість створити приватні мережі, в яких розміщуються чутливі програми та дані мультимедіа. Директори бірж не розкрили повний спектр апаратного та програмного забезпечення, яке вони використовують, однак вони підтвердили, що Brocade пропонує тканину крос-з'єднання. Група використовує контролери та комутатори OpenFlow, але це не єдиний маршрут SDN, який група вивчає. Біржа планує випробувати Cisco onePK та інші підходи.

Останнім часом як віртуалізація SDN, так і мережеві функції (NFV) набули центрального етапу в операторах мобільних операторів та інших мереж постачальників послуг. Технології можуть забезпечити організацію ресурсів та послуг з динамічним забезпеченням, яке може зайняти хвилини, а не місяці. Зокрема, дивлячись на такі функції керуючої площини, як сховище даних користувача (UDR), SDN та NFV, це може допомогти при забезпеченні часу надання. Крім того, інші проблеми, вирішені SDN та NFV, включають відділення логіки програми та примусового виконання від відповідних даних абонентів. Технології вирішують проблеми двома способами - за допомогою динамічної оркестрації ресурсів та інтелектуальної оркестрації послуг. Динамічна оркестрація ресурсів, яка використовує NFV, вимагає рівномірного стека віртуалізації в додатках, часто в контексті приватної хмари для операторів.

Комутатори SDN використовуються для тестування нової архітектури мереж даних в Університеті Іллінойсу в Океанському кластері Урбана-Шампейн для експериментальних архітектурних мереж. Нова архітектура дозволяє інженерам нарощувати пропускну здатність між серверами і робити це, не витрачаючи значних коштів на обладнання. Організація встановила 13 комутаторів Pica8, які мають загалом 670 портів.

Платформа Huawei Cloud Fabric для SDN ЦОД побудована на базі серії комутаторів Huawei CloudEngine. Однією з багатьох відмінних рис сімейства

комутаторів CloudEngine є можливість прямої інтеграції з середовищем VMware за допомогою контролера віртуалізації мережі (VMware NSX) за допомогою протоколу OVSDB (Open vSwitch Database). У загальному вигляді платформу зображено на рисунку 1.3.

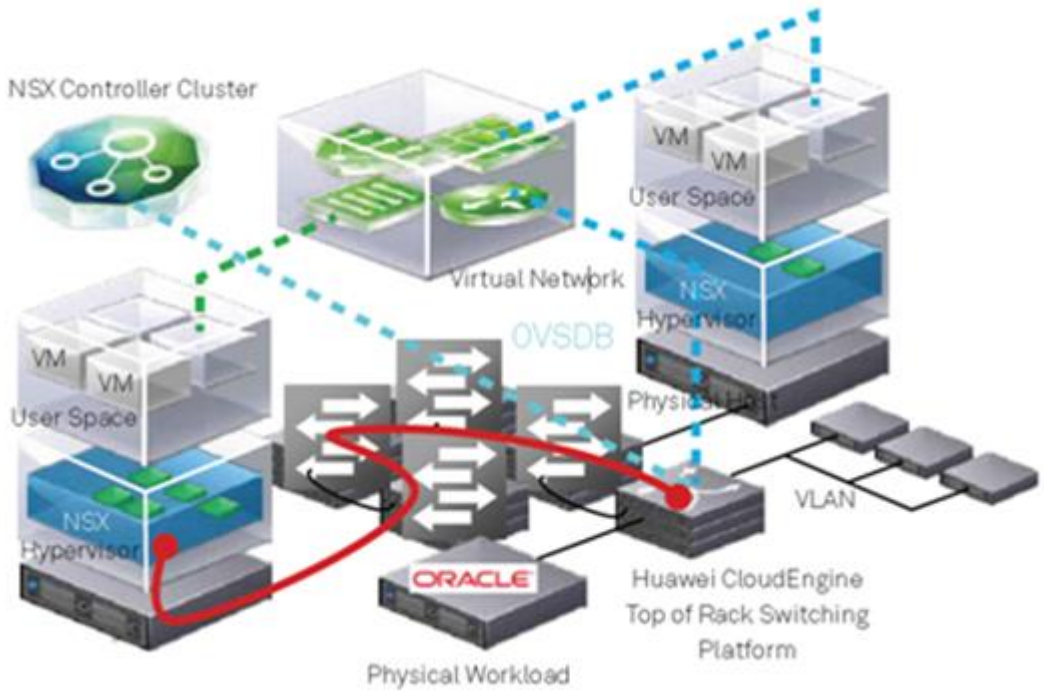


Рисунок 1.3 – Платформа Huawei Cloud Fabric для SDN

Завдяки цьому процеси управління, планування і конфігурації мережевої інфраструктури можуть здійснюватися власними коштами vSphere. Крім того, такий зв'язок дозволяє апаратних платформ мережевої інфраструктури працювати в тісній взаємодії з обчислювальної інфраструктурою, що забезпечує екстремально просте, пов'язане управління обчислювальними і мережевими ресурсами при створенні, переміщенні або видаленні віртуальних машин.

Такий рівень інтеграції і автоматизації кардинально скорочує час, ресурси, трудовитрати і експлуатаційні витрати, пов'язані з розгортанням нових послуг, оптимізацією ресурсів і аварійним відновленням в середовищі, де використовується платформа віртуалізації VMware.

Huawei Cloud Fabric забезпечує значне скорочення часу та спрощення

процедури виведення на ринок нових сервісів, аж до рівня моделі «самообслуговування». Завдяки цьому IT-ресурси ЦОД можуть перестати функціонувати як «центр витрат» і трансформувати свою роль в «бізнес-акселератор» цифрової ери [10].

Інтеграція з Cloud Fabric реалізується шляхом впровадження в операційну середу VMware vSphere через API мережевої віртуалізації vSphere. Інформація про стан, профілі, асоціації сегмента віртуальної мережі з конкретними віртуальними машинами і зміни в них в режимі реального часу передаються безпосередньо SDN-контролером vSphere на комутатори CloudEngine.

Інтеграція між сімейством комутаторів ЦОД CloudEngine і vSphere на рівні API дає IT-адміністраторам однакове, просте і забезпечене якісною підтримкою рішення для автоматизації роботи з хмарою VMware, яке не потребує внесення будь-яких змін в існуючу середу VMware.

Функціональні можливості, що виникають в результаті інтеграції Cloud Fabric з VMware vSphere, забезпечують суттєву перевагу в таких аспектах:

- спрощення процесу розгортання послуг і додатків;
- скорочення часу, необхідного для створення і оптимізації настройки мережевих ресурсів;
- зниження рівня операційних, адміністративних і логістичних накладних витрат;
- скорочення числа помилок при розгортанні і конфігурації.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ І АЛГОРИТМІВ УПРАВЛІННЯ ТРАФІКОМ SDN

2.1 Використання мережі SDN для підвищення безпеки Ethernet

На сучасному етапі розвитку мережевих технологій підвищується попит користувачів на інтенсивний та екстенсивний розвиток можливостей мереж. Така тенденція стимулює спеціалістів з мереж до винаходу нових методологій та розвитку перспектив мережевих архітектур. SDN є одним з перспективних напрямів розвитку мережевих технологій.

Домінуючою мережевою технологію у напрямку локальних мереж є технологія Ethernet.

Приклад застосування Ethernet та SDN для домашньої мережі наведено на рисунку 2.1.

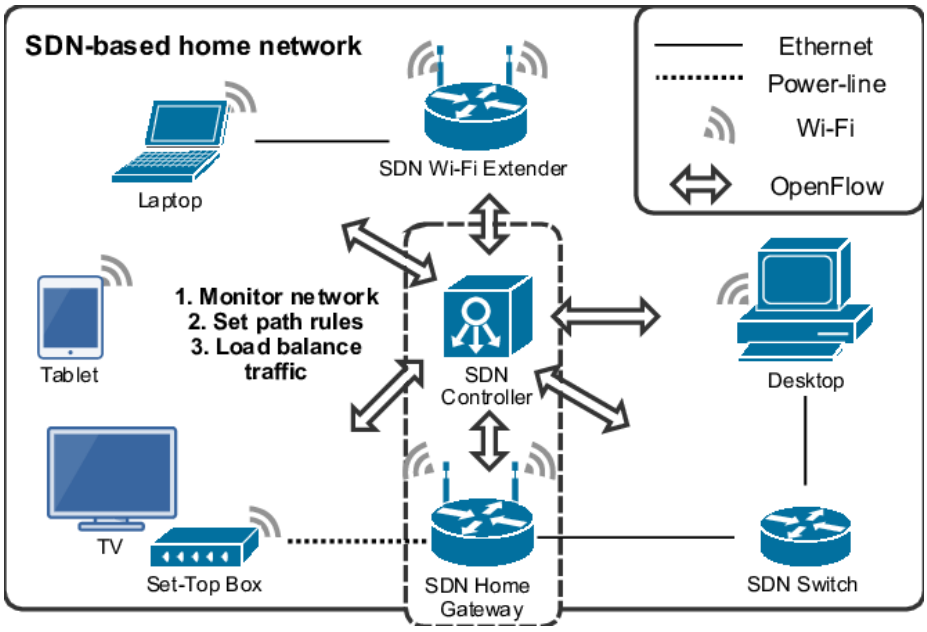


Рисунок 2.1 – Гібридна архітектура домашньої мережі на основі SDN

Більшість провідних локальних мереж, що використовуються сьогодні, використовують стандарт IEEE 802.3 Ethernet (LAN MAN Standards

Committee, 2012), який надає технічні характеристики для передачі даних по витій парі мідного та волоконно-оптичного кабелю. Стандартна мережа Ethernet складається з хостів та перемикачів. Перемикачі використовуються для з'єднання хостів між собою та створення мережі. Типова для мережі топологія – це зірка, що перетворюється на сітку під час з'єднання хостів між собою [3].

Більшість рішень мережевої безпеки реалізовані у вищих шарах (шари 3-5). Також доступні деякі рішення нижчого рівня, але більшість з них є власними рішеннями, що надаються постачальником. До популярних та стандартизованих рішень належать брандмауер, глибока перевірка пакетів (DPI), системи запобігання вторгнень (IPS) та IEEE 802.1X.

Однією з найпопулярніших систем безпеки є брандмауер. Основне завдання брандмауерів – фільтрація та захист хоста або локальної мережі від небажаного трафіку зв'язку. Термін брандмауер дуже загальний, оскільки існує десятки різних реалізацій. Реалізації відрізняються залежно від того, наскільки глибоко слід досліджувати трафік (рівні 3-5). Простий брандмауер може запобігти трафіку відповідно до спочатку визначених змінних IP-заголовків (рівень 3), і він більше підходить для захисту мережі і трохи більш досконалий брандмауер на хості може слідувати сеансам з'єднання та запобігати спочатку визначеним незаконним з'єднанням.

Іноді брандмауери покращуються функціоналом DPI, який сканує трафік на рівні програми (рівень 5). Це робиться шляхом вивчення вмісту кожного прохідного пакета IP на предмет підозрілих даних. Таким чином, міжмережеві стіни з DPI мають можливість ідентифікувати програми та атаки протоколів вищого рівня.

DPI також використовується в IPS. Завдання IPS – досліджувати трафік, що проходить, і запобігати вторгненню в систему чи мережу. Система IPS порівнює вхідний трафік із відомою базою даних атак і тим самим захищає трафік від зловмисних проникнень.

Хоча вищезазначені рішення забезпечують високий рівень зв'язку,

вони часто використовуються під час впровадження Ethernet LAN. Незалежно від того, IEEE 802.1X є стандартизованим рішенням для підвищення рівня безпеки Ethernet за допомогою функції аутентифікації. Оскільки хост підключається до комутатора Ethernet, він повинен надіслати свої облікові дані, щоб отримати зв’язок. Потім комутатор підтверджує справжність хоста від віддаленого сервера аутентифікації. Таким чином, лише аутентифіковані хости можуть спілкуватися в мережі.

Наразі дана технологія перебуває у кризовому стані. Первинними функціями технології Ethernet, що була винайдена декілька десятиліть тому, були гнучкість та децентралізованість. З часом технологія набула розвитку, обслуговуючи розширені високо потужні мережі, але питання безпеки архітектури Ethernet і досі залишається відкритим. Більш того, популярні мережеві технології, такі як віртуалізація мережі, використовують розширені сегменти Ethernet. Через відсутність маршрутизації в Ethernet в таких технологіях спостерігається від неефективна передача даних.

З іншого боку, мережева безпека невпинно розвивається в останній час. З розвитком нових протоколів та технологій для захисту з’єднання між хостами та у хостах з іншими мережевими приладами досі залишаються невирішеними питання безпеки. Експерти з мережевих технологій постають перед численними проблемами, оскільки їхні рішення обмежені концепцією того, як функціонує застаріла мережа. Такі мережі, як правило, побудовані з розподіленою інформацією. Ця проблема постає, коли вузли мережі обробляють свою поведінку відповідно до сусіднього вузла. Це дає змогу експертам з безпеки вирішувати проблеми безпеки мережі лише обмеженим чином. Проблеми безпеки було б легше вирішити, якщо контроль над усією мережею буде наданий суб’єкту безпеки.

Основна унікальна особливість, яка робить SDN популярною і випливає з її сутності – це можливість програмного конфігурування мережі. Це нова технологія, що розвивається, і має підстави стати революційною в галузі мереж. Тим не менш, аспекти безпеки повинні бути керовані та

оцінені, щоб забезпечити надійність мереж. Тому необхідні дослідження можливостей SDN для подолання проблем безпеки Ethernet.

У мережі Ethernet площини контролю та даних сумісні, що дає нападнику більше можливостей для пошуку можливих шляхів атаки мережі. У мережі SDN рівень контролю відділений від рівня даних, що ізолює потоки інформації від управління. Це ускладнює проникнення до мережі навіть при послідовних атаках [11].

Щоб атакувати мережу, нападник повинен отримати над нею контроль або фізично, або контролюючи існуючи ресурси мережі. Після отримання доступу нападник може виконати кілька видів атак.

Однією з проблем є неавторизований доступ до мережі. Під’єднатися до Ethernet досить легко через те, що дана технологія керується принципом мінімальної адміністрації. Це означає, що якщо несанкціонована особа отримує доступ до комутатора чи наявного хоста, що підключені до цільового сегмента, можливе несанкціоноване з’єднання. Виходячи з архітектури мережі Ethernet, розширення мережі можливо за допомогою підключення комутатора до ще одного перемикача. Тому, якщо несанкціонований користувач отримує доступ до мережі, то дозволяється розширювати мережу для надання декількох точок доступу (провідних або безпровідних) до цільової мережі [4]. Спочатку проблему можна вирішити, обмеживши точки доступу. У SDN рішення було б подібним, але вимагало б менше зусиль. Починаючи з централізованого контролера, порти комутаторів можуть бути відключені та включені з однієї точки, підтримуючи стан підключення до усіх хостів. Крім того, авторизація хостів може бути надана програмою-контролером, яка також авторизує комутатори під час підключення до них (тобто використовує OpenFlow).

Ethernet комутатори можуть адмініструвати VLAN з повідомленнями протоколу управління VLAN з будь-якого зі своїх портів. Це дозволяє несанкціонованій особі зробити хост або компрометований комутатор, щоб приєднатися до всіх VLAN та отримати доступ до обмеженого сегмента

мережі. У випадку, коли використовується SDN, VLAN це не конфігурації, створені в комутаторах, але програмні засоби, визначені в контролері, і потоки створюються в комутаторах відповідно. Власне, SDN усуває необхідність використання VLAN, як це робиться традиційно. Тому в SDN немає загроз VLAN, як у застарілому Ethernet.

Несанкціонована особа може надсилати в мережу повідомлення для дослідження відповідей та виявлення топології та послуг, що надаються господарями. Можна також визначити використані IP-адреси хостів, сервери та шлюзи, до яких хости підключені. Використовуваний діапазон IP також може бути визначений з відповідей DHCP. За допомогою даної інформації можна сканувати всю мережу та виявляти вразливі місця хостів, дозволяючи отримувати доступ для подальших атаки [22]. За допомогою SDN можна видалити широкомовні повідомлення з мережі, не впливаючи на продуктивність або функціональність. Отже, оскільки SDN може перекваліфікувати мережу та дати їй змогу функціонувати як брандмауер, з можливістю IPS та IDS можливостей, це показує, що програмованість надає багато переваг для налаштування мережі під власні потреби користувача.

Якщо наявна інформація про вразливість у хості, атака може бути виконана за допомогою Ethernet як носія. Атака маскується під звичаний сеанс і залишається невиявленою. Тут, як і у випадку виявлення топології та вразливості, для запобігання може використовуватися брандмауер, IDS, IPS. Як правило, за допомогою SDN хостинг-проби також важко виявити.

Коли встановлюються нові перемикачі, для них діє стандартний пароль для доступу до адміністративних інструментів та послуг. Якщо отримати пароль або виявити вразливість комутатора, можна отримати доступ до комутатора та керувати ним.

Зловмисник може перенаправляти трафік. Тим не менш, застарілі Ethernet комутатори, як правило, обмежені своїм обладнанням та операційною системою, щоб виконувати лише дії комутації та управління потоком. Перемикач бажано розміщувати в надійному та секретному місці.

Адміністративні інструменти та сервіси завжди повинні зберігатись за паролями, а краще, якщо це можливо, за послугами аутентифікації з сертифікатами, такими як TLS, SSL, або чим-небудь подібним [23]. У SDN перемикач пов’язаний з його контролером. Без контролера перемикач не може діяти в будь-якій мережі. Апаратне забезпечення комутатора SDN є більш потужним, ніж у застарілому комутаторі, оскільки обробка потоку та відповідність пакетів вимагає більшої потужності для обчислення.

Систематизуємо теоретичні дані у вигляді таблиці 2.1.

Таблиця 2.1 – Порівняльна характеристика традиційних методів захисту Ethernet та використання SDN для захисту Ethernet

Проблема	Традиційне рішення	Рішення за допомогою SDN
1	2	3
Неавторизований доступ до мережі	Обмеження точок доступу, утримання вимикачів та інших мережевих пристроїв у захищених та таємних місцях	Порти комутаторів можна керувати і вмикати з однієї точка, авторизація хостів може бути надана додатком, який також авторизує комутатори при підключенні до них
Під’єднання до VLAN	Ретельна конфігурація індивідуальної сегментації VLAN в мережі	Нема потреби у використанні VLAN, а отже, нема подібної загрози
Віддалений доступ до LAN	Соціальна інженерія – консультація користувачів з приводу використання антивірусів тощо	Аналогічно до традиційного методу

Продовження таблиці 2.1

1	2	3
Виявлення топології та слабких місць	Нема прямого методу вирішення проблеми	Можливість видаляти широкомовні повідомлення з мережі, не впливаючи на продуктивність чи функціональність
Вторгнення до мережі	Використання брандмауерів, спам-фільтрів, антивірусного та антишпигунського програмного забезпечення	Аналогічно до традиційного методу
Встановлення контролю над перемикачем	Розміщення перемикачів у надійному та секретному місці	Фізично ізолювати мережу управління від керованої мережі даних

Таким чином, SDN пропонує багато рішень для підвищення безпеки застосування технології Ethernet. Було проаналізовано традиційну технологію Ethernet, проблеми безпеки даної технології, її сучасний стан та засоби вирішення цих проблем за допомогою SDN. Отримані дані було систематизовано у вигляді таблиці.

2.2 Алгоритм рою сальп для управління трафіком SDN

У 2017 році С. Мірджалі запропонував новий стохастичний алгоритм оптимізації під назвою алгоритм Salp Swarm – алгоритм рою сальп [19]. Алгоритм рою сальп імітує пошуки їжі та океанічну навігацію риб роду Salpidae. Такі риби мають пляшкоподібне тіло. Сальпи утворюють ланцюги, як зграя або рої в глибоководній частині океану для навігації та пошуку

кормів. Ланцюг сальп має два рівні ієрархії: лідер і послідовники. Ведучий знаходиться попереду і керує роєм (послідовниками) під час навігації в багатовимірному просторі пошуку, знаходячи найкраще рішення (джерела їжі) проблеми оптимізації. У загальному вигляді ланцюг сальп зображено на рисунку 2.2.

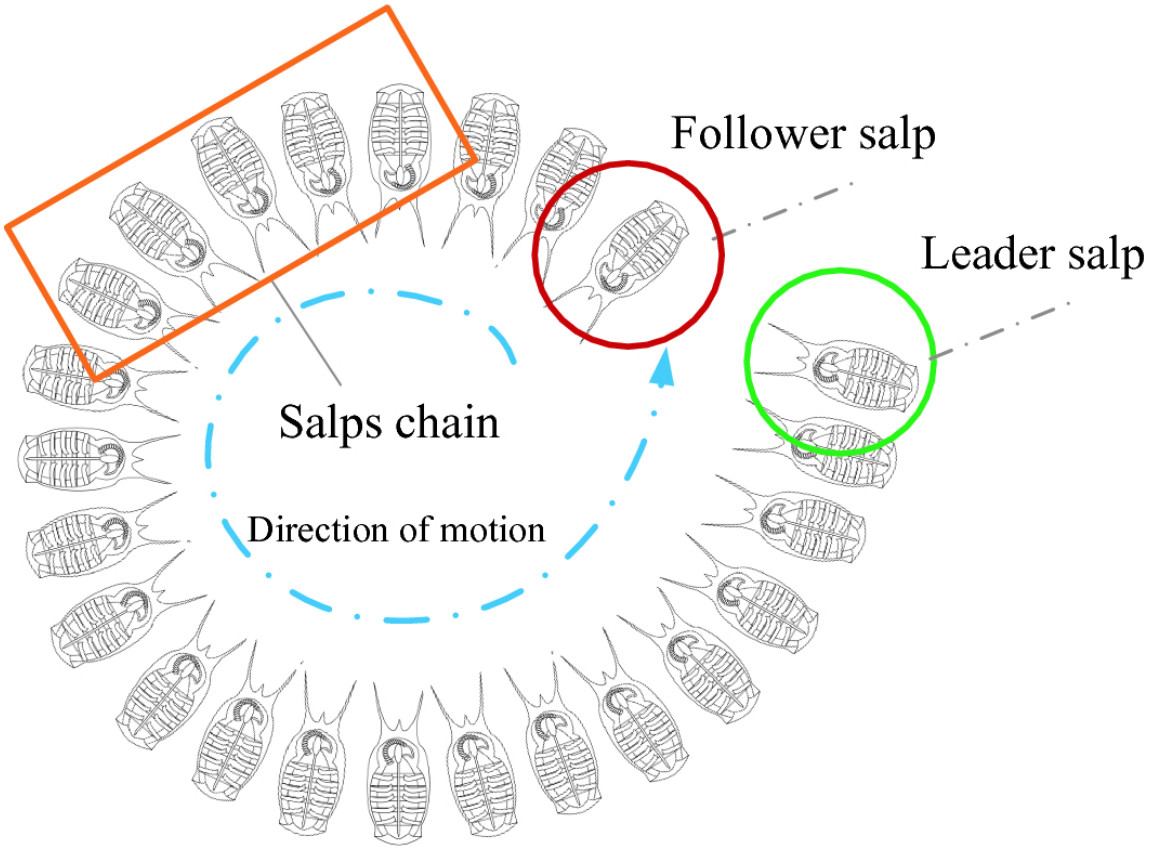


Рисунок 2.2 – Схематичне зображення ланцюгу сальп

Для математичного моделювання ланцюгу сальп рій спочатку поділяється на дві групи: лідер і послідовники. Лідером є сальп на передній частині ланцюга, тоді як решта сальп вважаються послідовниками. Як випливає з назви цих категорій, лідер веде рій, а послідовники йдуть один за одним (і керівник формально також є частиною ланцюгу).

Аналогічно іншим методам, що базуються на рою, положення сальп визначається в n-мірному просторі пошуку, де n - кількість змінних даної задачі. Тому положення всіх сальп зберігаються в двовимірній матриці, що називається X. Також передбачається, що в пошуковому просторі є джерело

їжі, яке називається F, як ціль рою.

Місцезнаходження лідера сальп визначається наступним виразом 2.1.

$$X_i^1 = \begin{cases} F_i + C_i\left((ub_i - ul_i)C_2 + lb_i\right), C_3 \geq 0 \\ F_i - C_i\left((ub_i - ul_i)C_2 + lb_i\right), C_3 < 0 \end{cases}, \quad (2.1)$$

де X_i^1 – визначає місце розташування головного сальпи в i-тому вимірі,

F_i – розташування джерела їжі в i-тому вимірі;

ub_i –верхня межа в i-тому вимірі;

lb_i – нижня межа в i-тому вимірі;

C_1, C_2 і C_3 – коефіцієнти моделі.

Ці коефіцієнти є випадковими числами, які використовуються для певних цілей. Перший коефіцієнт C_1 вводиться для забезпечення балансу між пошуком і безпосереднім використанням джерела корму, і являє собою найбільш важливий параметр в алгоритмі. C_2 і C_3 - випадкові величини, які генеруються відповідно до рівномірним розподілом з інтервалу від 0 до 1 [19].

C_1 визначається наступним чином (2.2):

$$C_1 = 2e^{-\left(\frac{4t}{T_{max}}\right)^2}, \quad (2.2)$$

де t – поточна ітерація;

T_{max} – максимальна кількість дозволених ітерацій.

Місцезнаходження сальп-членів ланцюга визначається у відповідності з наступним рівнянням (2.3):

$$X_i^k = \frac{1}{2}\left(X_i^k + X_i^{k-1}\right) \quad 2 \leq k \leq n, \quad (2.3)$$

де X_i^k - позиція k-ого члена рою в i-тому вимірі,

n - загальне число сальп.

Така стратегія детально описується у прикладі 2.1.

```
Algorithm 1 SSA algorithm
1: Initialize the number of salps, Max iterations
2: while Max iterations do
3:  for  do
4:  Calculate the fitness of each salp
5:  end for
6:  F is the food source (best agent)
7:  Update
8:  for  do
9:  if  then
10:   Update the position of the leading salp using Eq. (1)
11:  else
12:   Update the position of the follower salp using Eq. (4)
13:
14:  end if
15:  end for
16: end while
17: return F as the best solution.
```

Приклад 2.1 – Алгоритм роя сальп

У даному алгоритмі сальпи утворюють ланцюги для процесу годівлі. Сальпи орієнтуються за реактивним рушійним механізмом. Ведучий оновлює свою позицію щодо джерела їжі на основі наявної інформації про сусідство. У цьому алгоритмі відсутня можливість пошуку потенційного рішення для глобального пошуку через відсутність параметра інерції. Інерція допомагає контролювати швидкість та напрямок об’єкта. Однак ця стратегія добре працює для одномодальних та мультимодальних проблем другорядного виміру. Але для проблем з вищими вимірами це дає незадовільні результати із зображенням поганої конвергенції. Тому рівняння оновлення позиції необхідно змінювати в кожній ітерації для досягнення оптимального рішення.

Пропонується зважений алгоритм рою сальп на основі оновлення положення зваженої відстані для підвищення ефективності класичного алгоритму Salp Swarm. Для підвищення можливостей пошуку класичного алгоритму Salp Swarm стратегія оновлення місця змінюється як зважена сума найкращих позицій, а не середня. Ця стратегія досягається шляхом

обчислення ваги на основі коефіцієнтів векторів.

Ця зважена стратегія оновлення позицій забезпечує баланс між можливостями пошуку та експлуатації пошукових агентів. Крім того, це покращує швидкість конвергенції, що призводить до підвищення продуктивності SSA. Ця стратегія детально пояснюється в наступному прикладі 2.2.

```
Algorithm 2 WSSA algorithm
1: Initialize the number of salps, Max iterations
2: while Max iterations do
3:  for eachsalp do
4:  Calculate the fitness of each salp
5:  end for
6:  F is the food source (best agent)
7:  Update r1
8:  for eachsalp do
9:  if i==1 then
10:   Update the position of the leading salp
11:  else
12:   Update the position of the follower salp using
13:   t=t+1
14:  end if
15:  end for
16: end while
17: return F as the best solution.
```

Приклад 2.2 – Зважений алгоритм рою сальп

Щоб побачити, наскільки запропонована зважена модель та алгоритм Salp Swarm є ефективними при вирішенні проблем оптимізації управління трафіком SDN, розглянемо деякі їх переваги:

- алгоритм Salp Swarm зберігає найкраще рішення, отримане дотепер, і призначає його в якості змінної джерела їжі, тому значення ніколи не втрачається;

- алгоритм Salp Swarm оновлює положення лідера сальп тільки щодо джерела їжі, що є найкращим рішенням, отриманим на даний момент, тому лідер завжди досліджує та експлуатує простір навколо нього;

- алгоритм Salp Swarm оновлює положення послідовників сальп відносно один одного, тому вони поступово рухаються до лідера сальп;

- поступові рухи послідовних сальп перешкоджають застоюванню в

локальних оптимах;

- алгоритм Salp Swarm порвівняно простий і легкий в реалізації.

Ці переваги роблять алгоритм теоретично і потенційно здатним вирішити одноцільові проблеми оптимізації управління трафіком SDN. Адаптивний механізм дозволяє цьому алгоритму уникати локальних рішень і, врешті-решт, знаходить точну оцінку найкращого рішення, отриманого під час оптимізації. Тому його можна застосувати як до одномодальних, так і до мультимодальних проблем.

2.3 Математична модель мульти контролера SDN

Для того, щоб сформулювати завдання оптимізації і визначити функцію корисності, перш за все, розглянемо модель мережі. Мульти контролер SDN управляє M комутаторами, які використовують відповідний інтерфейс SDN, наприклад, OpenFlow. Комутатори OpenFlow з'єднані з N контролерами, розподіленими по мережі, і складовими мультиконтролера мережі SDN. Позначимо набір контролерів як C_i (i = 1,2, ..., N), а набір комутаторів як S_j (j = 1,2, ..., M).

Для виконання будь-якої мережевої операції, кожен комутатор OpenFlow запитує обслуговування у відповідного контролера. Контролер передає на комутатор вимоги по комутації та таблицю переадресації [16].

Оскільки ресурси контролера, включаючи пам'ять, пропускну здатність і обробку інформації, природно, обмежені, контролер Ci може керувати тільки обмеженим числом комутаторів OpenFlow K_i. K_i – це число комутаторів, пов'язаних з певним контролером C_i в конкретний момент часу, яке може визначено з матриці комутаторів мульти контролера шляхом підсумовування числа комутаторів в ряду контролера. рядки матриці представляють різні контролери, а стовпці - карти комутаторів. Контролери не повинні бути перевантажені. Це пов'язано з тим, що ймовірність збою значно збільшується зі збільшенням навантаження на контролер, особливо в

умовах перевантаження [24]. Приклад матриці комутаторів мульти контролера з N = 5 і M = 7 представлений формулою (2.4), де матриця K включає в себе число підключених комутаторів до кожного з контролерів.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \gg\gg [K] = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix} \quad (2.4)$$

Одним із способів оцінки продуктивності контролера може служити тривалість часу відповіді контролера на запит комутатора, що робить вирішальний вплив на затримку. Мульти контролер може бути представлений моделлю з чергами виду M/M/s [26]. На рисунку 2.3 показана структура системи з чергою, в якій передбачається, що кожен контролер має s станів.

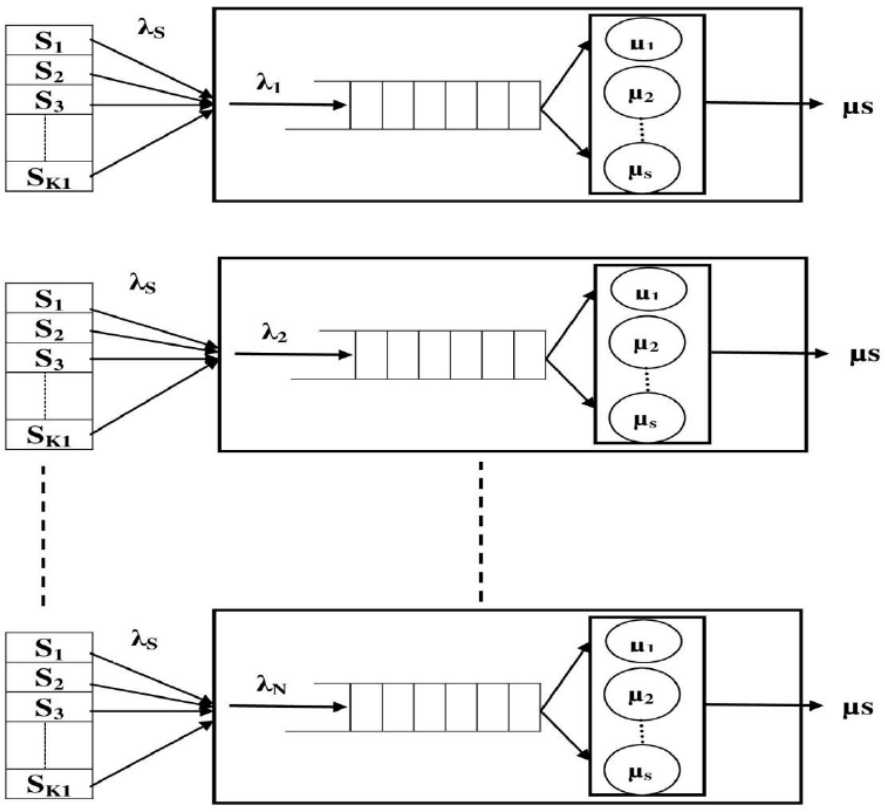


Рисунок 2.3 – Структура системи мульти контролера з чергою

Передані пакети надходять на контролер з інтенсивністю, яка визначається пуасонівським процесом, і формують просту чергу на обслуговування конкретним контролером.

На рисунку 2.4 представлена діаграма станів для моделі черг M/M/s, яка визначається марківським процесом загибелі-розмноження з безперервним часом.

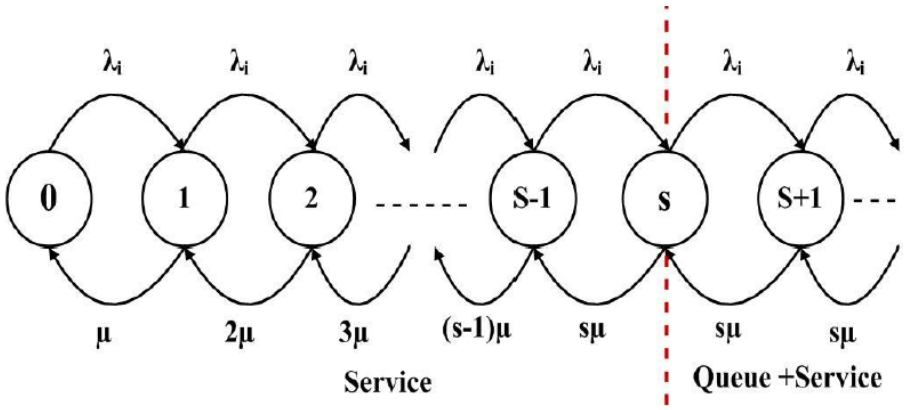


Рисунок 2.4 – Діаграма переходів для моделі M/M/s

Середній час відповіді T_i контролера C_i визначається як сума тривалості очікування в черзі і часу обробки, і може бути обчислено з використанням формули С. Ерланга як функція інтенсивності надходження λ_i і інтенсивності обслуговування μ. Обчислюється за наступною формулою:

$$T_i(\lambda) = \frac{C(S_j \frac{\lambda_i}{\mu}}{s\mu_i - \lambda_i} + \frac{1}{\mu}, \tag{2.5}$$

де C (s,λ/μ) – ймовірність того, що всі сервери в системі знаходяться в стані обслуговування і будь-який надходить пакет буде поставлений в чергу.

Цю ймовірність можна обчислити відповідно до таких формул.

$$C\left(S, \frac{\lambda}{\mu}\right) = \frac{\left(\frac{(s\rho)^c}{s!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{s-1}\frac{(s\rho)^k}{k!} + \left(\frac{(s\rho)^c}{s!}\right)\left(\frac{1}{1-\rho}\right)} \tag{2.6}$$

$$\rho = \frac{\lambda_i}{s.\mu}, \tag{2.7}$$

де ρ – це завантаження сервера, що вказує на стійкість системи.

Система має стійкий розподіл тільки тоді, коли ρ менше 1. Це можна побачити на діаграмі переходів на рисунку 2.4, коли отриманих заявок виявляється більше, ніж можливостей обробки у контролерів, інтенсивність обслуговування як і раніше sμ, але максимальні можливості контролера вже будуть завантажені повністю.

Інтенсивність надходження заявок λ_i на контролер C_i може бути обчислена як сума середніх інтенсивностей заявок на комутатор, сполучений з контролером (2.8):

$$\lambda_i = \sum k_i \lambda_s, \tag{2.8}$$

Відповідно, середнє навантаження на контролер C_i визначається як середнє число заявок, що знаходяться в черзі і в процесі обробки. Використовуючи формулу С Ерланга, нескладно отримати формулу для визначення навантаження на контролер L_i (2.9):

$$L_i(\lambda) = s\rho + \frac{\rho}{1-\rho} C(S, \frac{\lambda_i}{\mu}), \tag{2.9}$$

Отже, модель мульти контролера для мережі SDN може бути представлена у вигляді системи з чергою. Контролери не повинні бути перевантажені, що можна забезпечити за допомогою формули С. Ерланга. Було досліджено обчислення оптимального завантаження сервера. Коли кількість отриманих заявок є більшою за кількість можливостей обробки, що має кожен з контролерів, для уникнення повної завантаженості необхідно за можливості збільшити інтенсивність обслуговування.

2.4 Методи вирішення задачі мультиплікативного рюкзака для мережі SDN

Дані в мережі SDN можна розглядати як потік пакетів даних, які складаються із заголовка протоколу та корисного навантаження даних. Аналіз окремих пакетів даних не може освоїти основні вимоги запитів додатків, такі, як відео в режимі реального часу та онлайн-ігри, які мають основні вимоги до пропускання даних у режимі реального часу.

Якщо основні вимоги не виконані, а пакет, що надходить, знаходиться в недоступному стані, краще відмовитися від пакета безпосередньо. Розробка QoS допомагає певною мірою вирішити цю проблему за умови недостатньої інформації в традиційних мережах. У середовищі мережі SDN є можливість зібрати більше можливостей інформації про запит, використовуючи резервування ресурсів для отримання кращої продуктивності [18].

У цій роботі мережеві запити розглядаються з трьох аспектів, включаючи обсяг даних, час та парний зв'язок між запитами. Обсяг даних може бути представлений активною пропускною здатністю та активним часом у більш детальних вимогах. Час відображає тривалість запиту і може мати зв’язок із самим раннім часом початку та останнім часом завершення відповідно до більш детальних вимог. Парний зв’язок між запитами має різні форми виконання, такі як два запити, розділені на визначений час, або перший запит з невеликим обсягом даних, що використовуються для узгодження, і другий запит з великою кількістю даних, що використовуються для передачі.

Стан мережі можна спростити до вигляду (T_0, T_2,…, T_m), де T_i – тривалість простою і-го сегмента. Можна ввести додаткову тривалість простою (m + 1) -го відрізка T_{m+1}, але можна опустити T_{m+1} через те, що останній час простою T_{m+1} є нескінченно довгим, розглядаючи питання про те, щоб поставити запит у режим очікування мережі максимально близько до початку послідовності. Набір мережевих запитів зменшується до (t_1, t_2,…, t_n),

а t_i – час, зайнятий $_i$-м запитом.

Як показано на рисунку 2.5, метою оптимізації моделі обробки запитів SDN є розміщення запитів у вигляді неактивної мережі таким чином, щоб загальна довжина запиту, що був розміщений у неактивній мережі, завжди була найбільшою.

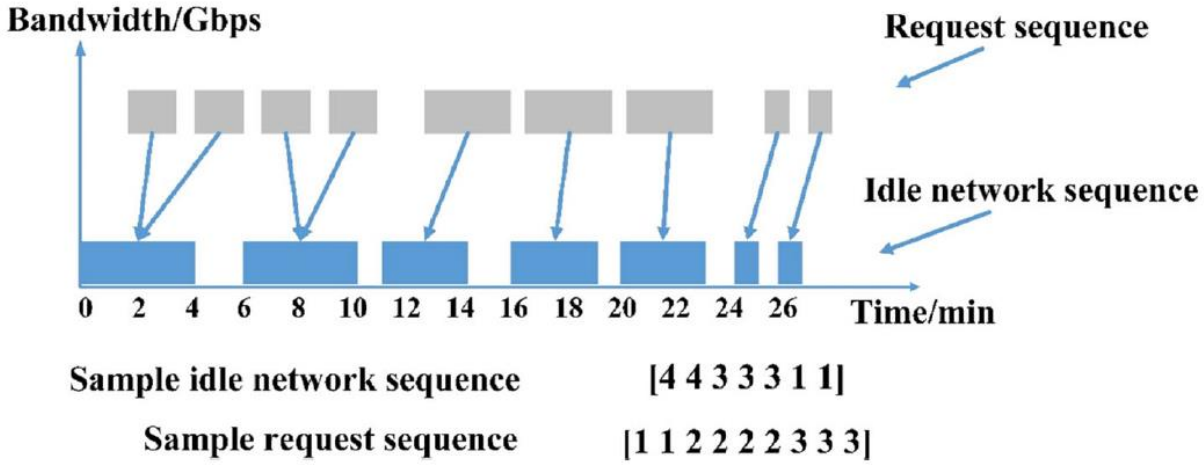


Рисунок 2.5 – Модель обробки запитів у SDN

На рисунку 2.5 загальна довжина неактивної мережі для першої групи - 19, як і загальна довжина запиту 19, тому всі запити розміщуються в режимі очікування. Однак неактивну мережу неможливо заповнити в деяких випадках.

Таким чином, цей документ моделює мережеві запити таким чином:

- активна пропускна здатність B.
- активний час T.
- найранніший час початку t1
- найпізніший час завершення t2.
- зв'язкове відношення між запитами P [21].

Тому мережевий запит виражається у вигляді запиту з 5 кортежами: {B, T, t1, t2, R}.

Можливо перетворити модель та підвищити її ефективність. Що стосується пропускної здатності, для запиту може знадобитися смуга пропускання B, а загальна смуга пропускання лінії - B_t. Дискретизуємо

суцільну пропускну здатність і спрощуємо стан пропускної здатності до моделі 0–1. Один запит, що споживає більшу пропускну здатність, ніж інші, може розглядатися як кілька одночасних запитів, що споживає меншу пропускну здатність.

До того ж, мережа з більшою пропускною здатністю може складатися з декількох мереж пропускної здатності 0–1. Можна замінити оригінальний запит на кілька зв'язаних запитів, не призначаючи пропускну здатність окремо. Ситуація стає набагато складнішою, коли алгоритми пропонуються безпосередньо за згаданою вище моделлю. Можна спочатку спростити модель, а потім поступово змінювати алгоритм, який пропонується у спрощеній моделі, нарешті застосувати алгоритм у складній моделі.

Для запиту одного типу 0-1 досить легко зіставити його на задачі з кількома рюкзаками, враховуючи лише тривалість часу заняття. Найпростіша модель спочатку відображається в задачі мультиплікативного рюкзака, а потім поширюється на складніші випадки.

Проблема з рюкзаком була запропонована з великою кількістю доповнень як модель оптимізації, однією з яких є проблема мультиплікації. Проблема мультиплікації може бути описана як задані n об'єктів розміром w_1, w_2, ⋯, w_n та m умовними рюкзаками розміру c_1, c_2, ⋯, c_n (де w_i і c_j - це додатні цілі числа). Потрібно знайти m роз'єднаних підмножини w таким чином, щоб m пакетів було заповнено максимально, а це означає, що всі об'єкти в пакетах мають найбільшу суму значень. Математична модель описується так:

$$\max\left(\sum_{j=1}^{m}\sum_{i=1}^{n} W_i X_{ij}\right) \qquad (2.10),$$

Дана модель також повинна відповідати наступним умовам (2.11):

$$\begin{cases} \sum_{j=1}^{m} X_{ij} \le 1, \ \forall i \epsilon \{1,2,\dots,n\} \\ \sum_{i=1}^{n} W_i X_{ij} \le C_j, \forall j \epsilon \{1,2,\dots,m\}, \\ X_{ij} \epsilon \{1,0\} \end{cases} \qquad (2.11)$$

X_{ij} = 1 означає, що об'єкт і покладений у рюкзак j, навпаки X_{ij} = 0 означає, що об'єкт і не покладений у мішок.

Тому послідовність запитів може розглядатися як послідовність пункту, тривалість запиту аналогічна розміру елемента, непрацююча мережа моделі пропускної здатності 0–1 аналогічна рюкзаку, і час очікування може розглядатися як розмір рюкзака. Метою оптимізації є розміщення запиту в режимі очікування, завдяки чому загальна довжина запиту, яка була розміщена в мережі очікування, завжди є найдовшою. У практичних програмах залишилися запити можуть бути розміщені після останнього зайнятого часового відрізка, оскільки тривалість запиту, розміщеного в рюкзаку, є найдовшою, а загальна тривалість залишків запитів мінімізована, а також послідовність запитів обробляється в найменший час [20].

Крім того, можна додати інші умови до проблеми багатокутника, щоб відобразити більш складні моделі. Для найбільш раннього часу початку t_1 та останнього часу завершення t_2 у запиті: {B, T, t_0, t_2, R}. Для рюкзака, який вважається мережею простою, записується час початку і закінчення холостого ходу T_0, T_0 відносно рюкзака як {W, T_0, T_2}. Запит, який кладуть у рюкзак, повинен відповідати необхідним умовам: t_0 + T <T_0 && t-T> T_1. Слід зазначити, що коли запит з t_1 або t_2 розміщується в (T_1, T_2), на наступні запити вплине це обмеження, оскільки запит може існувати лише у певному просторі в ранці. Розглядаючи розбиття рюкзаків, будемо використовувати евристичний алгоритм для визначення способу розміщення. Стан рюкзака описується залежно від інших правил даного правил алгоритму.

Необхідно відповідно модифікувати модель рюкзака. Взявши за приклад два запити з фіксованим інтервалом часу, такі два запити можна розглядати як запит, який займає більше часу, включаючи час двох запитів і час інтервалу. Середина інтервалу, природно, утворює непрацюючу мережу, і тоді такі два запити можуть розглядатися як елемент у рюкзаку. Далі, рюкзак також може бути розміщений у виділений період, який може розглядатися як запит із часом початку і часом завершення.

2.5 Алгоритм летючої миші для мережі SDN

Алгоритм летючої миші – це евристичний інтелектуальний алгоритм, запропонований Вангом з Кембриджського університету в 2010 році, який моделює принцип ехолокації, що використовується в процесі полювання летючих мишей. Розробляючи основну ідею алгоритму летючої миші, Ванг запропонував основні припущення алгоритму:

- усі летючі миші використовують свою ехолокацію, щоб сприймати відстань до цілі, одночасно виявляючи різницю між ціллю та фоновою перешкодою;
- положення летючої миші – x_i, політ з будь-якою швидкістю v_i, пошук мішені з фіксованою частотою f_{min}, змінна довжина хвилі l та гучність A_i. Вони можуть визначати відстань між собою та здобиччю та автоматично регулювати довжину хвилі (або частоту) імпульсу, одночасно регулюючи частоту $r \in [0, 1]$ імпульсу в міру наближення до цілі;
- гучність змінюється багатьма способами, припускаючи, що вона змінюється від максимального значення (додатного) A_0 до фіксованого мінімального значення A_{min};
- тривимірна топографія та часові затримки тут ігноруються, хоча це, ймовірно, є однією з особливостей багатовимірних обчислень. Однак невідомо, як це використовувати на даному етапі дослідження, і це значно збільшує кількість обчислень в багатовимірних обчисленнях одночасно [15].

У даній роботі використовується алгоритм для спрощеної моделі запиту даних у мережі SDN з урахуванням зайнятого часу за обставин пропускної здатності 0-1. Стан мережі можна спростити як (T_1, T_2,…, T_m), де T_i - тривалість простою і-го сегменту. Як було згадано вище, можна ввести додаткову тривалість простою (m + 1) -го відрізка T_{m+1}, але можна опустити T_{m+1} через те, що останній час простою T_{m+1} є нескінченно довгим, враховуючи, що запит поставлено у режим очікування мережі максимально

близько до початку послідовності. Набір мережевих запитів зменшується до $(t_1, t_2,\ldots, t_n)$, і t_i це час, який займає і-й запит.

Для визначення місця розташування летючої миші в рюкзаку, ключовим питанням є оновлення місця розташування летючої миші. Необхідно визначити, як оновлюється позиція x_i та швидкість виша в двовимірному просторі пошуку. Спосіб оновлення швидкості v_i^t та положення x_i^t на кроці t показаний нижче:

$$f_i = f_{min} + (f_{max} - f_{min})\beta, \quad (2.12)$$

$$v_i^{t-1} + (x_i^t - x_*)f_i, \quad (2.13)$$

$$x_i^t = x_i^{t-1} + v_i^t, \quad (2.14)$$

Серед усіх значень найближче сусіднє значення найкращого рішення генерує локальне оптимальне рішення. Формула оновлення вказана наступним чином:

$$x_{new} = x_{old} + \varepsilon A^t \quad (2.15)$$

де A_t – це середня гучність усіх летючих мишей у час t,

$\varepsilon\epsilon[-1,1]$ – це напрямки та довжини, що генеруються випадковим чином.

Наведемо приклад використання алгоритму летючої миші (приклад 2.3).

```
      Input: Current network status = (T1, T2, …, Tm), Network requests set =
(t,t2,…,tn)
      Output: Map Sequence = (xa, x2,…,xn)
      Valuation function: g(x) = ∑_{j=1}^{n} y_i * t_i,   y_i = {1 x_i > 0; 0 x_i = 0
      Objective function: f(x)=max(g(x))
      Initialize the Map Sequence and Velocity sequence v = v1, v2, …, vn)
      Set the pulse frequency sequence f = (f1, f2, …, fn)
      Initialize the pulse frequency ri and loudness sequence Ai
      For i=1 to N // Iterate N times
      // Adjust the frequency, produce new solution and update the position
and speed (expression (2.12), (2.13), (2.14))
      if rand>r
      end if
      if rand <A&Fnew<g(x*) // Valuation of the new solution
      end if                                        // Accept the new solution,
Increase ri, decrease Ai
```

```
      // Compare the new solution with the current one and update the current
one
      end for
      return Map Sequence
```

Приклад 2.3 – Приклад застосування алгоритму летючої миші

Отже, враховуючи здатність мережі SDN збирати інформацію, а також вивчати ідею розподілу міток та резервування ресурсів, було запропоновано використання алгоритму летючої миші для SDN.

Поява SDN дозволяє більш точно контролювати мережеві пакети. У даній роботі використовується алгоритм для спрощеної моделі запиту даних у мережі SDN з урахуванням зайнятого часу за обставин пропускної здатності 0-1. Використовуючи загальні можливості планування контролера та характеристики запиту даних різних програм, можна покращити використання мережевих ресурсів.

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ АЛГОРИТМУ РОЮ САЛЬП ТА АЛГОРИТМУ ЛЕТЮЧОЇ МИШІ ДЛЯУПРАВЛІННЯ ТРАФІКОМ SDN

3.1 Застосування хаотичного алгоритму рою сальп для SDN

Переваги застосування метаевристичних моделей та алгоритмів включають масштабованість, простоту та скорочення часу на обчислення. Однак у цих алгоритмів є два основні недоліки: спад у локальних оптимах та низький коефіцієнт конвергенції. Одним зі способів подолання цих проблем та підвищення продуктивності метаевристичних алгоритмів є розгортання теорії хаосу. Хаотичні карти використовуються замість випадкових чисел в алгоритмах на основі оптимізації рою часток для посилення конвергенції [27]. Таким чином, доцільно ввести алгоритм рою сальп на основі теорії хаосу, який замінює випадкові змінні хаотичними. Хаотичний алгоритм рою сальп використовує хаотичні карти для регулювання значення другого коефіцієнта C_2. Значення C_2 можна замінити значенням відповідної хаотичної карти при поточній ітерації таким чином:

$$C_2^t = \omega(t), \tag{3.1}$$

де ω(t) - значення хаотичної карти при t-ітерації.

Рівняння (3.1) можна переписати, використовуючи нове значення C_2, наступним чином:

$$X_i^1 = \begin{cases} F_i + C_1\left((ub_i - ul_i)\omega(t) + lb_i\right), C_3 \geq 0 \\ F_i - C_1\left((ub_i - ul_i)\omega(t) + lb_i\right), C_3 < 0 \end{cases}, \tag{3.2}$$

Теорія хаосу – це поширений математичний підхід, що використовується для аналізу поведінки динамічних систем з критичними

початковими умовами.

Одним із способів відображення такої поведінки є використання хаотичних карт, які є або дискретними, або безперервний. Хаотичні карти можна розгорнути лише для детермінованих систем із передбачуваною поведінкою. Останнім часом теорія хаосу стає більш привабливою для багатьох систем у різних галузях, таких як інформатика, робототехніка, фізика та мікробіологія.

Хаотичні карти стають найпотужнішим рішенням для підвищення продуктивності метаевристичних алгоритмів за рахунок підвищення їх параметрів випадковості. Ці випадкові параметри вибираються на основі рівномірного або гауссового розподілу, тому вони можуть бути краще керовані хаотичною картою, що має ту саму характеристику з кращими показниками. Контроль цих параметрів за допомогою хаотичних карт зменшує локальну оптимуму та збільшує конвергенцію [25].

На основі результатів, отриманих в роботі, оптимальною хаотичною картою для нашого оптимізатора є логістична карта. Загальне рівняння для логістичної хаотичної карти:

$$\omega(t+1) = a\omega(t)[1-\omega(t)],\ a = 4, \tag{3.3}$$

Початковий стан хаотичної карти вважається рівним 0,7 (ω (0) = 0,7).

У класичному алгоритмі рою сальп позиція їжі, яка є найкращою позицією сальпи, являє собою рішення оптимізованої проблеми. Для модельованої проблеми необхідно встановити оптимальну кількість контролерів та оптимальне з'єднання для кожного вимикача в наборі комутаторів S. Ця проблема вважається NP-складною задачею .

Одним із способів вирішення подібних проблем є використання метаевристичних алгоритмів, оскільки детерміновані алгоритми в такому разі використати не вдається. Для вирішення цієї проблеми доцільно використовувати алгоритм рою сальп, який є своєрідним алгоритмом рою

часток. Алгоритм рою сальп –- це найсучасніший алгоритм на основі оптимізації рою часток, який вводить більш високу продуктивність, ніж інші алгоритми на основі оптимізації рою часток.

Основна ідея алгоритму рою сальп полягає в тому, що декілька сальп одночасно шугають оптимальне рішення. Оптимальне рішення у випадку SDN – це оптимальна кількість контролерів та оптимальний розподіл контролерів між перемикачами [14]. Отже, два алгоритми у вкладеному циклі розгорнуті для отримання найкращих рішень; два алгоритми засновані на алгоритмі рою сальп. Алгоритм 1 з прикладу 3.1 вказує псевдо-код для алгоритму рою сальп, що вводиться для визначеної проблеми, де кожен сальп представляє кількість контролерів у мережі, на виході з цього алгоритму отримуємо оптимальну кількість контролерів.

```
Algorithm 1 CSSA for optimal number of controllers
1: Initialize ub, lb, tmax, d, n
2: Initialize positions of salps xi (i = 1,2,3,. . .,n)
3: While (t _ tmax)
4: Calculate the fitness function of each salp position
5: F = The best salp position
6: Update the value of C1
7: Get the value of chaotic map (Logistic) w(t)
8: For (i = 1 : i _ n) do
9: if (i == 1)
10: Update the position of leading salp
11: else
12: Update the position of follower salp
13: end if
14: end for
15: Adjust salps based on the upper and lower bounds
16: Calculate the best connections of switches for the best
salp (call Algorithm 2)
17: Update the best salp based on the results of Algorithm2
18: t t + 1
19: Return F
```

Приклад 3.1 – Псевдо-код для алгоритму рою сальп

Алгоритм 2 вказує псевдо-код для хаотичного алгоритму рою сальп, розгорнутого для пошуку оптимальних з'єднань для всіх комутаторів, виходячи з оптимальної кількості контролерів, обчисленої за алгоритмом 1

(приклад 3.2).

```
Algorithm 2 CSSA for optimal switches to controllers connections
1: Initialize ub, lb, tmax, d, n
2: Initialize positions of salps xi (i = 1,2,3,. . .,n)
3: While (t _ tmax)
4: Calculate the fitness function of each salp position
5: F = The best salp position
6: Update the value of C1
7: Get the value of chaotic map (Logistic) w(t)
8: For (i = 1 : i _ n) do
9: if (i == 1)
10: Update the position of leading salp
11: else
12: Update the position of follower salp
13: end if
14: end for
15: Adjust salps based on the upper and lower bounds
16: t t + 1
17: Return F
```

Приклад 3.2 – Псевдо-код хаотичного алгоритму рою сальп

Кожен сальп в алгоритмі 2 представляє всі наявні з'єднання для всіх комутаторів з їх виділеними контролерами, і це M-мірний вектор з кожен вимір являє собою перемикач. Вихід другого алгоритму вказує на найкращий розподіл контролерів між комутаторами.

Система починає працювати при встановленні початкових параметрів хаотичного алгоритму рою сальп, що включають нижню межу, верхню межу та максимальну кількість ітерацій. Наступним кроком є ініціалізація n кількості сальпів випадковим чином з кожним сальпом, що представляє кількість доступних контролерів у мережі. Придатність кожної сальпи обчислюється наступним чином (3.4):

$$U = \sum_A U_{ci}/|A|, \tag{3.4}$$

де A – загальна кількість контролерів.

Сальп з найвищою придатністю вважається найкращим рішенням на даний момент. Місце розташування найбільш підходящої сальпи вважається місцем харчування. Параметри алгоритму рою сальп оновлюються, і

відповідно до них оновлюються позиції сальп.

Процес оновлення позицій сальп та оцінювання придатності кожної сальпи повторюється до тих пір, поки не буде знайдене оптимальне рішення або не досягнуто максимальної ітерації.

Оцінка ефективності запропонованої моделі проводиться у імітаційному середовищі. Запропонований алгоритм реалізований для різних топологій, використовуючи Matlab. Запропонований алгоритм реалізований та протестований за допомогою Matlab над комп'ютері, що має процесор Intel Core i5 та 8 ГБ оперативної пам'яті. Потік вважається випадковою змінною, яка слідує за розподілом Пуассона. Параметри моделювання, включені в процес оцінювання, вказані у таблиці 3.1.

Таблиця 3.1 – Параметри моделі хаотичного алгоритму рою сальп для тестування

Параметр	Опис	Значення
λ_S	Середня кількість запитів комутатора	[1500,3000]
μ	Швидкість обслуговування контролера	30 000 запитів у секунду
T	Порогове значення затримки	2 мс
U_{ub}	Верхня межа значення індексу придатності контролера	0,9
$t_{max}(1)$	Максимальна кількість ітерацій для Алгоритму 1	50
$t_{max}(2)$	Максимальна кількість ітерацій для Алгоритму 2	18
δ_C	Коефіцієнт зважування витрат	18
δ_T	Коефіцієнт зважування часу	25

Для першого розробленого контролера SDN NOX кількість запитів середнього потоку, що обробляються в секунду, становлять близько 30 000. Таким чином, середній рівень обслуговування контролера встановлюється на рівні 30 000. Розглядаються два основні сценарії для десяти топологій. Топології, що використовуються для моделювання, наведені у таблиці 3.2.

Таблиця 3.2 – Топології, що використовуються для моделювання

Категорія	Топологія	Кількість пристроїв переадресації
1	ARPANET 1969_12	4
1	MREN	6
1	GetNet	7
1	Sprint	11
1	NSF	13
2	Claranet	15
2	IBM	18
2	Oxford	20
2	FCCN	23
2	AGIS	25

У першому сценарії метою є аналіз впливу зміни порогового часу s на рішення (тобто оптимальну кількість контролерів). У цій ситуації індекс використання верхньої межі кожного контролера U_{ub} вважається постійним і дорівнює значенню в таблиці параметрів моделювання (тобто U_{ub}=0,9). Запропонований алгоритм представляє випадок, в якому враховується певне значення порогу відгуку часу. У другому сценарії перевіряється вплив зміни верхнього індексу використання кожного контролера при постійному значенні порогової затримки s. Верхній показник використання в основному вважається для кожного доступного контролера для компенсації будь-якого раптового збільшення мережевого трафіку та уникнення роботи контролера

на максимальній потужності.

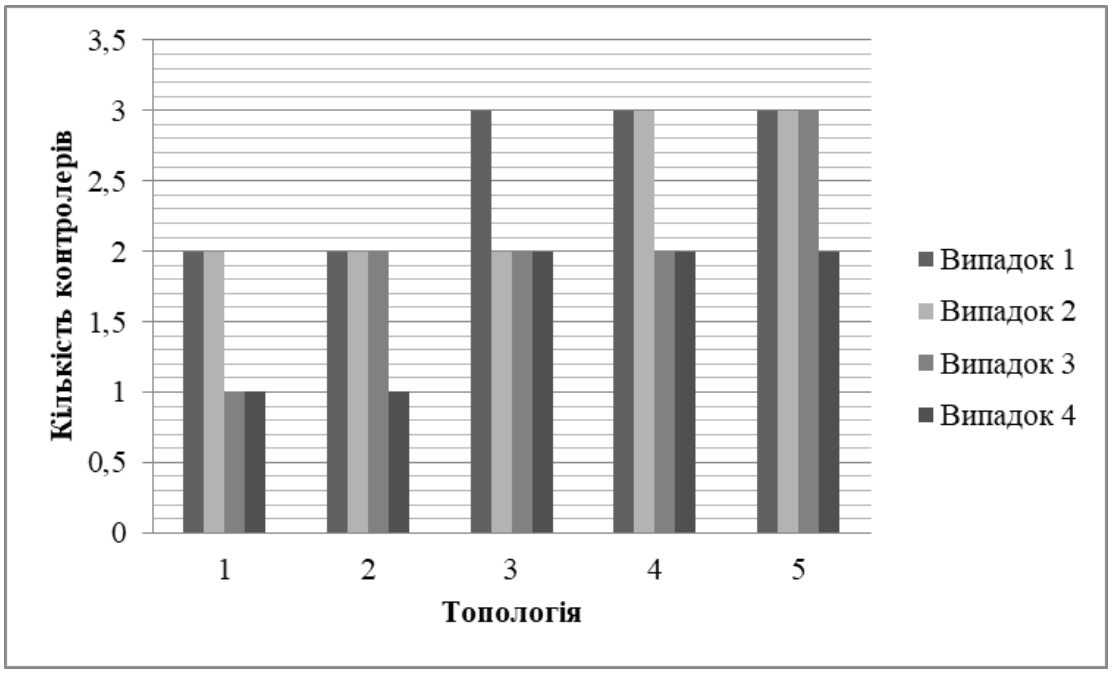


Рисунок 3.1 – Оптимальна кількість контролерів для першого сценарію у першій категорії топологій

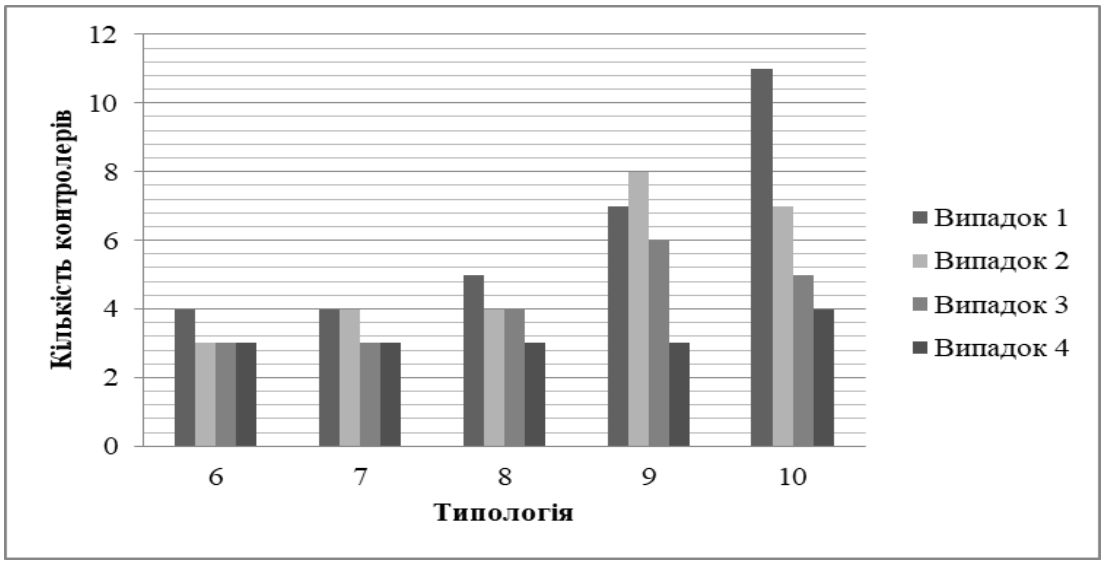


Рисунок 3.2 – Оптимальна кількість контролерів для першого сценарію у другій категорії топологій

Запропонований алгоритм реалізований для кожної топології мережі кілька разів; щоразу враховується нове значення U_{ub}. Для всіх випадків у сценарії порогова характеристика часу τ встановлюється на значення,

визначене в таблиці параметрів моделювання (тобто $\tau = 2$ мс).

Рисунок 3.1 ілюструє результати для різних випадків у першому розглянутому сценарії, арисунок 3.2 – для другого.

Розглянемо результати роботи мережі у другому сценарію для різних типів топологій.

Рисунок 3.3 ілюструє оптимальну кількість контролерів для кожного випадку в другому сценарії для топологій першої категорії.

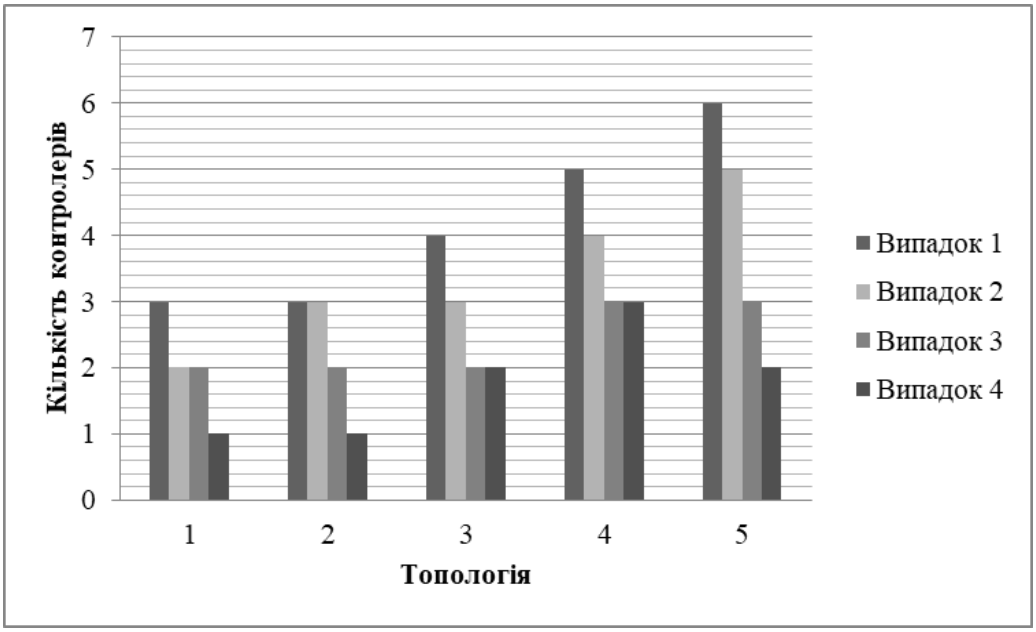


Рисунок 3.3 – Оптимальна кількість контролерів для кожного випадку в другому сценарії для топологій першої категорії

Це можна інтерпретувати наступним чином: збільшення порогового часу τ дозволяє збільшувати час відгуку кожного розгорнутого контролера; таким чином, контролер може працювати з більшою кількістю пристроїв переадресації. Це відбувається за рахунок затримки, яка повинна підтримувати необхідний QoS системи. Результати другого сценарію для другої категорії топології представлені на рисунку 3.4.

Результати показують, що зі збільшенням верхньої межі індексу використання контролера оптимальна кількість контролерів, необхідних для підтримки мережі, зменшується. Для кожної топології першої або другої категорії, у міру збільшення верхнього індексу використання кожного

розгорнутого контролера U_{ub} необхідна кількість контролерів зменшується. Переходячи від випадку до випадку, кількість доступних контролерів або залишається незмінною, або зменшується. Збільшення верхнього індексу використання контролера дозволяє контролеру запропонувати більше ресурсів і, таким чином, виконувати більше завдань. Це головна причина скорочення кількості розгорнутих контролерів із зростанням U_{ub}. Найбільш оптимальне значення для верхнього індексу використання пов'язане з характером мережі та ймовірністю швидкого зміни потоку.

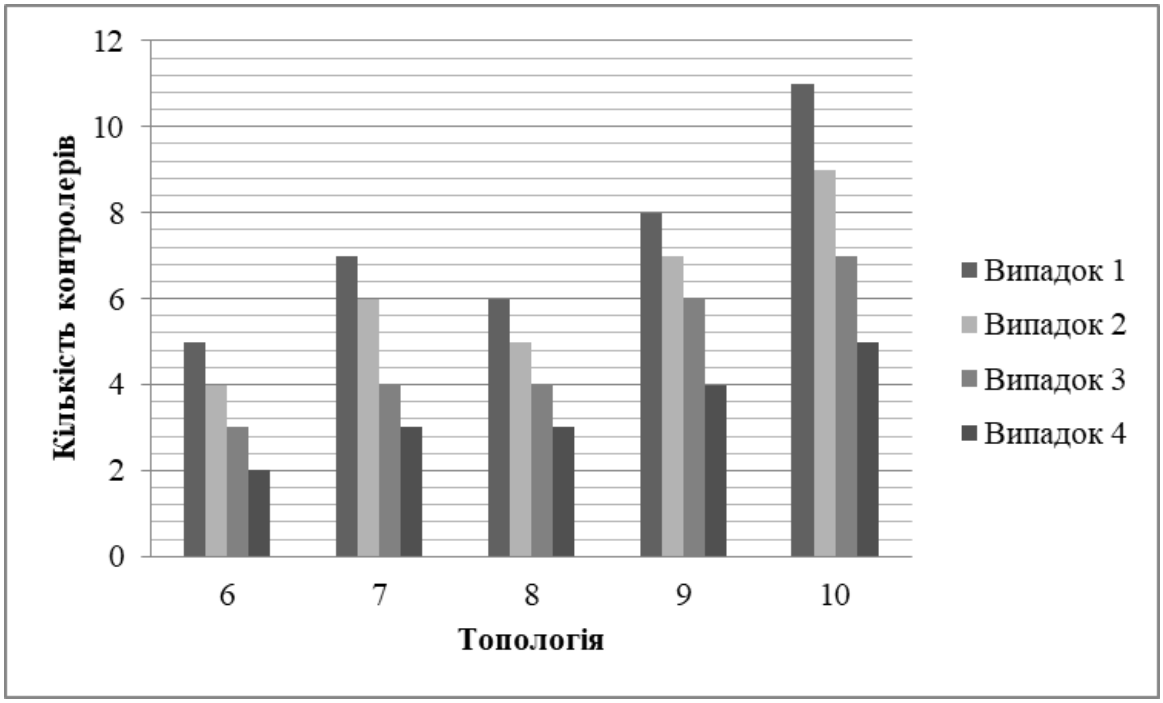


Рисунок 3.4 – Оптимальна кількість контролерів для кожного випадку в другому сценарії для топологій другої категорії

Найбільш поширене значення U_{ub} 0,9, що вказує на те, що 90 відсотків ресурсів контролера використовуються, а десять відсотків ресурсів відокремлено для різких змін. У таблиці 3.3 вказані обидва значення τ та U_{ub}.

Для кращої оцінки введеного алгоритму та для перевірки впливу змін параметрів пропонований алгоритм реалізується для випадкової мережі з 30 комутаторів. Система перевіряється на зміну порогового часу τ, а також верхнього індексу використання Uub у більшій шкалі, ніж у попередніх моделюваннях.

Таблиця 3.3 – Варіація параметрів для випадкової мережі

Значення U_{ub}	Значення τ
U_{ub1}=0,8	τ_1=1 мс
U_{ub2}=0,85	τ_2=5 мс
U_{ub1}=0,9	τ_3=10 мс
U_{ub1}=0,95	τ_4=15 мс

Це пов’язано з важливістю цих параметрів та їх впливом, особливо для широкомасштабних мереж. Тому алгоритми повинні вестись для щільних мереж для підтвердження запропонованої роботи. Розглядаються два основні випадки; у першому випадку алгоритм реалізований для різних значень порогового часу τ з чотирма різними значеннями максимального індексу використання контролера U_{ub}. Другий випадок перевіряє систему на широкий діапазон значень U_{ub} з чотирма різними значеннями порогового часу τ.

Рисунок 3.5 ілюструє вплив зміни порогового часу затримки контролера τ на оптимальну кількість контролерів, необхідних для мережі, у чотирьох розглянутих випадках.

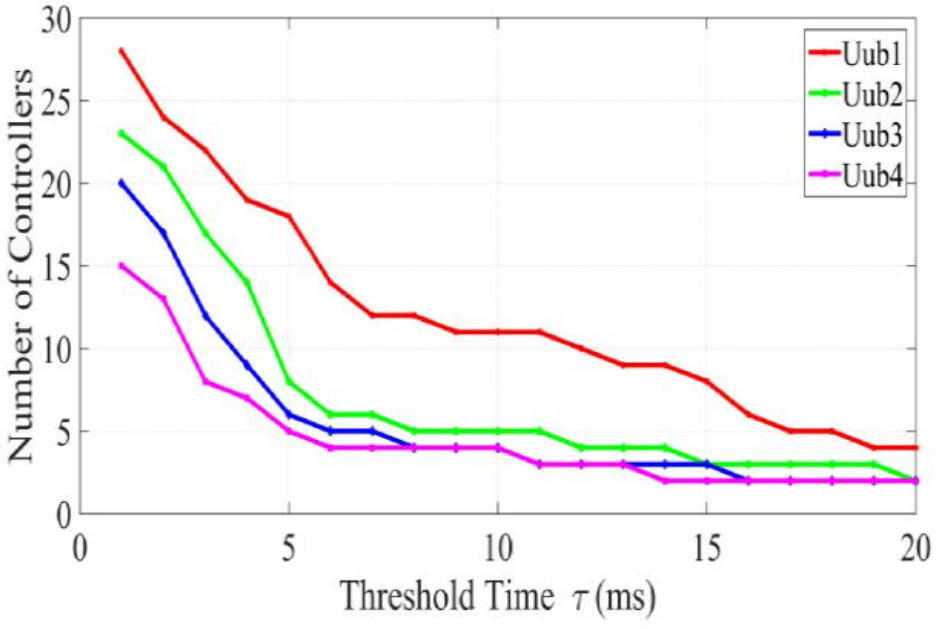


Рисунок 3.5 – Вплив зміни порогового часу затримки контролера τ на оптимальну кількість контролерів, необхідних для мережі

Кожен випадок представляє певне значення максимального індексу використання контролера U_{ub}. Чотири значення вибрані таким чином, що вони охоплюють можливий діапазон U_{ub} і, таким чином, дають надійну присутність. Перша крива представляє зміну при індексі використання 80 відсотків, що вважається мінімальним коефіцієнтом використання. Таким чином, крива представляє найгірший випадок між чотирма кривими, оскільки вона відповідає найгіршому значенню U_{ub}. З іншого боку, четверта крива являє собою варіацію при максимальному індексі використання 95 відсотків, що є найкращим випадком між чотирма графами. Для чотирьох кривих зі збільшенням порогового часу (тобто переміщення зліва направо) необхідна кількість контролерів зменшується. Найгірші значення кількості необхідних контролерів присвячені першій області кривих, при якій пороговий час s невеликий, і, таким чином, контролери не в змозі впоратися з великими завданнями. Найкращі значення кількості контролерів знаходяться в останніх частинах всіх кривих, при яких значення τ досить високе для вирішення багатьох завдань і служить для багатьох комутаторів.

Рисунок 3.6 ілюструє на зміну оптимальної кількості контролерів зі зміною максимального показника використання кожного контролера для чотирьох різних значень порогового часу τ.

Перша крива являє собою найгірший варіант, оскільки вказує на найвищі значення кількості необхідних контролерів. Це пояснюється тим, що мале значення порогового часу заважає контролеру виконувати багато завдань, і, таким чином, контролер може служити лише для мінімальної кількості пристроїв переадресації. На відміну від першої кривої, четверта крива характеризує найкращі результати за кількістю необхідних контролерів. Це можна трактувати як високе значення порогового часу τ дозволяє контролеру підтримувати більше комутаторів.

Ще один важливий параметр, який слід враховувати, - це зміна запиту потоку. Зміна швидкості запиту потоку комутаторів призводить до зміни ситуацій розгорнутих контролерів.

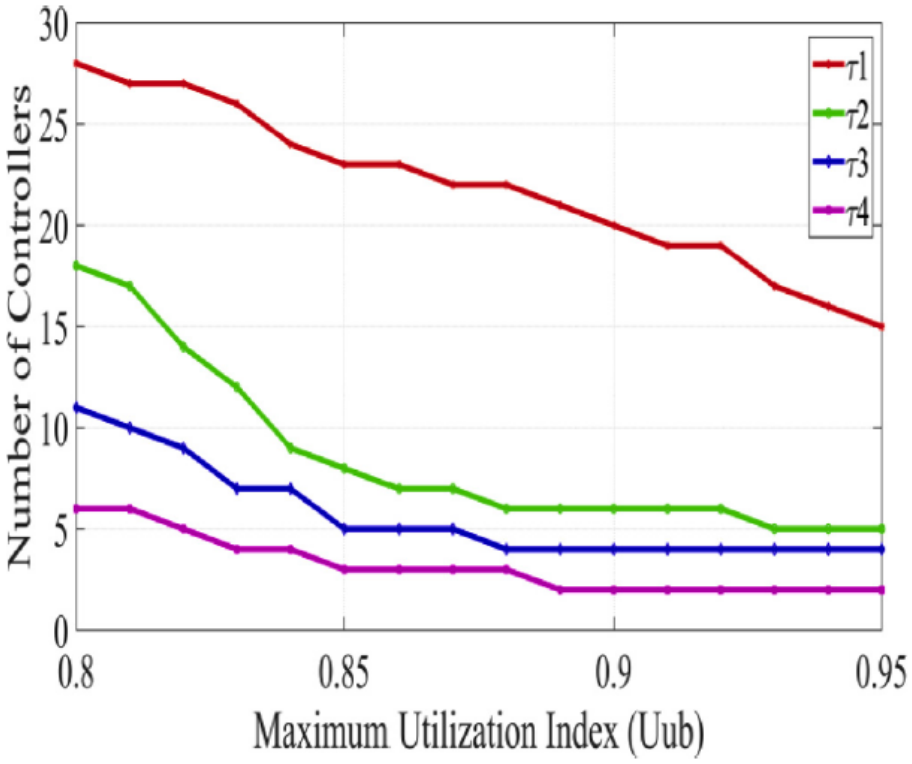


Рисунок 3.5 – Вплив зміни максимального значення показника придатності на оптимальну кількість розгорнутих у мережі контролерів зі зміною порогового часу затримки контролера

Кількість розгорнутих контролерів слід збільшувати, якщо витрата збільшується. Таким чином, більше навантаження на мережу повинно зіткнутися з активацією більшої кількості контролерів, тоді як зменшення навантаження в мережі повинно створювати меншу кількість активних контролерів.

Отже, мережа починається з певної кількості активних контролерів і до того часу кількість активних контролерів зростає і падає, виходячи з навантаження мережі. Динамічна зміна навантаження в мережу вносить динамічну зміну кількості активних контролерів. Для перевірки впливу динамічних змін запитів потоку на кількість активних контролерів розглядаються чотири випадки з різними діапазонами швидкості потоку кожного пристрою пересилання. У кожному випадку розглядаються три значення порогового часу τ та максимального використання кожного контролера U_{ub}.

3.2 Дискретний алгоритм летючої миші для управління трафіком SDN

Алгоритм летючої миші, описаний у розділі 2, не може бути застосований безпосередньо до мережі SDN, оскільки рішення про розташування летючої миші є безперервним. У спрощеній моделі запиту SDN відповідність між запитом і простою мережею дискретна. Тому алгоритм слід дискретизувати.

Існує запропонували бінарна версія алгоритму бінарних летючої мишы для вирішення проблем вибору особливостей у розпізнаванні візерунків та розфарбуванні графіків. Серед них ключовим для перетворення безперервних значень у бінарне є введення сигмоїдної функції:

$$S(x) = \frac{1}{1+e^{-x}}, \tag{3.5}$$

Для певних x∈ (−∞, + ∞) маємо S(x)∈(0, 1). Крім того, призначається значення x до 0 або 1, порівнюючи функцію rand з S(x), таким чином змінюючи безперервне значення на дискретне бінарне значення.

$$x = \begin{cases} 1 & \text{if rand} < S(x) \\ 0 & \text{otherwise} \end{cases}, \tag{3.6}$$

Однак, що стосується послідовності карти (x_1, x_2,…, x_n), $x_i \in [0, m]$, m - кількість незадіяних мережевих сегментів, 0 являє собою запит, розміщений не в першій m простої мережі. Більше того, x_i не може просто прийняти значення 0 або 1, оскільки дискретного бінарного значення недостатньо. Для отримання безперервного дискретного значення між [0, m], можна звернутися до методу в задачі на забарвлення графіка та перетворити x_i у вектор довжини m. Таким чином, послідовність карт (x_1, x_2, ..., x_n) перетворюється в матрицю n * m карт. Матриця 9 * 3 задовольняє наступним умовам:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Для простою мережі m сегментів і n запитів матриця карт є матрицею n*m. Кожен стовпець представляє запит, а кожен рядок представляє непрацюючу мережу.

Значення, розташоване в (i, j) в матриці, є «1», означає, що j-й запит вводиться в і-ту мережу, що не працює.

Кожен стовпець має максимум «1», що означає, що запит розміщується в одній непрацюючої мережі. Кожна точка може отримати 0–1 значення дискретно бінарним способом.

Однак у цього підходу є дві проблеми. По-перше, важко переконатися, що кожен стовпчик дискретний лише з одним значенням. По-друге, матриця занадто рідка і витрачається простір, коли m стає більшим. Перетворимо x_i у вектор довжини k, де $2^{k-1} \leq m < 2^k$, k - довжина двійкової форми приблизно m. Тоді матриця карти перетворюється в матрицю n*k. Як показано, матриця 9*2 задовольняє наступним умовам:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ & & & & - & & & & \end{bmatrix}$$

- для неактивної мережі m сегментів і n запитів матриця карт - матриця n * k, де $2^{k-1} \leq m < 2^k$.

- кожен вектор стовпчика є двійковим числом. Крім того, $a_i = v$ означає, що і-й запит вводиться в v неактивну мережу.

- значення кожного стовпця задовольняє $a_i = v \leq m$. Крім того, $a_i = 0$ означає, що і-й запит не розміщено в колишній мережі очікування. Таким чином, 0–1 значення можна отримати шляхом двійкової дискретизації від кожної точки.

Обчислюємо x_{ti} за формулою (2.14) та x_{new} за формулою (3.4), а потім перетворюємо його у матрицю n*k. Тепер дані значення можуть бути використані як дискретні.

Симуляційний експеримент у цій роботі проводився на ПК із процесором Intel (R) Core (TM) 2 Due CPU E7500E при 2,94 ГГц та 8,00 ГБ пам'яті. Алгоритм оцінки використовує дискретний алгоритм летючої миші і порівнює його з результатами жадібного алгоритму, алгоритму динамічного програмування, а також оптимального рішення відповідно. Використаємо MATLAB для запуску коду експерименту.

Оскільки для процесу підбору вводиться випадковий фактор, для рішення, яке не задовольняє умові обмеження, необхідно відкинути і повторно обрати його. Для того, щоб отримати рішення, що задовольняє обмеженню, необхідно зробити безліч випадкових відборів у кінцевій конвергенції, щоб отримати оптимальне дискретне рішення. В іншому випадку витрата часу дуже велика і неконтрольована, тому вводиться максимальна кількість ітерацій, завдяки чому алгоритм отримує результат за відносно розумний час. Стратегія жадібного алгоритму виправлена. У таблиці 3.4 наведено приклад наборів даних для неактивних мереж та запитів.

Таблиця 3.4 – Набори послідовних даних для неактивних мереж та запитів

№з/п	Послідовність неактивної мережі	Послідовність запитів
1	[4,4,3,3,3,1,1]	[1,1,2,2,2,2,3,3,3]
2	[4,7,2,5,3,1,2]	[2,1,3,4,1,2,3,5,2]
3	[7,6,1,6,1,2,5]	[2,1,6,3,2,6,5,5,5]
4	[1,4,3,3,3,1,1]	[1,1,2,2,2,2,3,3,3]
5	[6,8,3,5,3,1,1]	[3,1,2,3,1,2,3,5,3]
6	[4,6,2,5,3,1,1]	[2,1,3,2,1,2,3,5,2]

Таблиця 3.4 – приклад створених людиною даних з різними характеристиками. Для запиту одного типу 0-1, можна легко зіставити його на багатопроцесорні проблеми, враховуючи лише тривалість часу заняття. Тому послідовність запитів може розглядатися як послідовність пункту, тривалість запиту аналогічна розміру елемента, неактивна мережа моделі пропускної здатності 0–1 аналогічна рюкзаку, і час простою може вважатися розміром рюкзака.

Метою оптимізації кількості контролерів є розміщення запиту в режимі очікування, завдяки чому загальна довжина запиту, яка була розміщена в мережі очікування, завжди є найдовшою.

Результати жадібного алгоритму, алгоритму динамічного програмування, дискретного алгоритму летючої миші та оптимального можна обчислити для кожного набору даних. Перерахуємо максимальний час запиту, який можна розмістити в режимі очікування мережі, і обчислимо послідовність відображення на основі жадібного алгоритму, дискретного алгоритму летючої миші і, нарешті, обчислимо відношення максимального часу запиту до оптимального рішення.

Для дискретного алгоритму летючої миші параметри встановлюються наступним чином: кількість сукупності n дорівнює 10, амплітуда A - 0,25, частота пульсу r - 0,5, а кількість ітерацій N – 20, як показано в таблиці 3.5.

Кожне число x з підписним індексом y у послідовності рішення кожного алгоритму являє собою y-й запит, що вводиться в x-ту мережу очікування, а x = 0 означає, що y-й запит не розміщується в жодній мережі простою. Співвідношення між оптимальним рішенням та рішенням при застосуванні жадібного алгоритму наведено у таблиці 3.5.

Також можна навести дані обчислення за алгоритмом летючої миші та порівняти результати із жадібним алгоритмом.

Співвідношення між оптимальним рішенням та рішенням при застосуванні алгоритму летючої миші наведено у таблиці 3.6.

Таблиця 3.5 – Порівняння оптимального рішення та жадібного алгоритму

№	Оптимальне рішення	Послідовність жадібного алгоритму	Рішення жадібного алгоритму	Співвідношення
1	19	[1,2,4,5,0,0,1,2,3]	15	78,89%
2	23	[3,0,4,1,6,7,5,2,4]	22	91,67%
3	27	[6,1,1,0,0,2,4,7,0]	25	89,28%
4	15	[1,2,5,0,0,0,2,3,4]	13	81,25%
5	24	[3,7,0,2,7,2,4,1,5]	22	78,57%
6	21	[3,1,1,4,2,5,4,2,0]	19	86,36%

Таблиця 3.6 – Порівняння оптимального рішення та алгоритму летючої миші

№	Оптимальне рішення	Послідовність алгоритму летючої миші	Рішення алгоритму летючої миші	Співвідношення
1	19	[7,6,4,2,2,0,3,5,1]	17	89,47%
2	23	[3,6,2,1,2,7,5,4,2]	23	95,83%
3	27	[6,3,4,0,1,2,7,1,0]	27	96,42%
4	15	[7,6,2,2,4,0,0,3,5]	14	87,5%
5	24	[1,6,2,2,7,1,5,4,3]	24	85,71%
6	21	[5,7,2,3,5,1,2,4,1]	21	95,45%

Алгоритм кажана досяг кращих результатів, ніж жадібний алгоритм за допомогою порівняння. У наборі даних також висвітлюються різні ситуації, коли жадібна стратегія не може отримати оптимальне рішення.

Загальна довжина неактивної мережі для першої групи мереж – 19, загальна довжина запиту – 19, тому всі запити можуть бути розміщені в режимі очікування в оптимальному рішенні. Загальна довжина неактивної мережі для третьої групи становить 28, загальна довжина запиту – 35,

загальна довжина запиту більше, ніж загальна довжина неактивної мережі, і неактивну мережу неможливо заповнити. Загальна довжина мережі, що працює в режимі очікування для четвертої групи – 16, загальна довжина запиту – 19, загальна довжина запиту – більше, ніж загальна довжина мережі простою, і мережа, що не працює, не може бути заповнена цими значеннями. Загальна довжина мережі роботи в режимі очікування для шести груп становить 22, загальна довжина запиту – 21, і нарешті всі запити розміщуються в режимі очікування в оптимальному алгоритмі рішення та в алгоритмі летючої миші.

Алгоритм летючої миші досяг кращих результатів порівняно з жадібним алгоритмом. Імовірність отримання кращих результатів корелює з його параметрами.

Можемо зробити висновок, що у випадку однакової кількості ітерацій, обґрунтовано, що зміна чисельності n швидше викликає зміну результату. Оскільки амплітуда A і частота імпульсів r більше впливають на швидкість конвергенції, вони впливають на конвергенцію алгоритму летючої миші, але їх вплив на результат порівняно невеликий.

Алгоритм, наведений у даній роботі, не передбачає зміну амплітуди A і частоту імпульсів r. При виборі відповідних параметрів дискретний алгоритм летючої миші дає задовільні результати.

ВИСНОВКИ

В атестаційній роботі було досліджено принципи функціонування SDN-мереж, їх архітектуру. Було проведено порівняльний аналіз SDN-мереж та традиційних мереж. Було досліджено галузі застосування галузі застосування SDN-мереж, адже даний тип мережі залишається новою технологією, призначеною замінити фізичний дизайн мережі мережевою інфраструктурою, керованою програмним забезпеченням. Зазвичай це виявляється практичним, порівняно економічно ефективним та динамічним рішенням.

SDN-мережа застосовується у різноманітних галузях: програми для спільної роботи, конвергентне сховище, мережевий обмін, організація послуг мобільної мережі, масштабовані мережі центрів обробки даних.

Було запропоновано можливі шляхи підвищення безпеки технології Ethernet за допомогою технології SDN. На даний момент технологія Ethernet перебуває у кризовому стані. Основною перевагою SDN є програмованість, що випливає з її сутності, а отже, це забезпечує майже цілковиту незалежність від фізичних носіїв, які є слабким місцем Ethernet.

У роботі також наведено модель мультиконтролера для мережі SDN. Дана модель може бути представлена у вигляді системи з чергою. Контролери не повинні бути перевантажені, що можна забезпечити за допомогою формули С. Ерланга.

Було наведено алгоритм рою сальп для мережі SDN. Для перевірки впливу динамічних змін запитів потоку на кількість активних контролерів розглядаються чотири випадки з різними діапазонами швидкості потоку кожного пристрою пересилання.

Враховуючи здатність мережі SDN збирати інформацію, а також вивчати ідею розподілу міток та резервування ресурсів, було запропоновано використання алгоритму летючої миші для SDN.

Поява SDN дозволяє більш точно контролювати мережеві пакети. Використовуючи загальні можливості планування контролера та характеристики запиту даних різних програм, можна покращити використання мережевих ресурсів. Також було вдосконалено даний алгоритм за допомогою дискретизації. Дискретний алгоритм летючої миші є доцільним для застосування при управлінні трафіком SDN.

Результати дослідження можуть бути запроваджені при оптимізації локальних мереж. Також результати можуть бути застосовані у навчальному процесі університету, зокрема при вивченні дисциплін «Комп’ютерні системи» та «Комп’ютерні системи та мережі».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Бовда Е.М. Метод управління перерозподілом навантаження в SDN мережах: зб. наук. праць / Військовий інститут телекомунікації та інформатизації. Київ : ВІТІ, 2017. С. 6-16

2. Дуравкин Е. В., Ткачева Е. Б., И. Саад. Архитектура SDN. Анализ основных проблем на пути развития / Е. В. Дуравкин, Е. Б. Ткачева, И. Саад // Системи обробки інформації. 2015. №3. С. 92-99

3. Красов А.В., Левин М. В., Цветков А. Ю. Метод управления трафиком в гибридной програмно-определяемой сети / А. В. Красов, М. В. Левин, А. Ю. Цветков // Информационные технологии и телекоммуникации. 2016. Т. 4 №2. С. 53-63.

4. Курочкин И. И., Гуменный Д. Г. Безопасность сетей SDN. Классификация атак // И. И. Курочкин, Д. Г. Гуменный / Современные информационные технологии и ИТ-образование. 2015. №11. Режим доступу: https://cyberleninka.ru/article/n/bezopasnost-setey-sdn-klassifikatsiya-atak

5. Лунтовський А. О., Семенко А. І. Застосування технологій SDN для програмної реалізації провайдерського ядра систем мобільного зв’язку 5G майбутнього покоління / А. О. Лунтовський, А. І. Семенко // Зв'язок. 2014. №3. С. 13-20

6. Лурджан М. Б., Воропаева А. А., Ступак Г. В. Исследование возможностей внедрения алгоритмов маршрутизации в программно-конфигурируемых сетях / М. Б. Лурджан, А. А. Воропаева, Г. В. Ступак // Наукові праці ДонНТУ. 2015. №1. С. 81-90

7. Маньков В.А., Краснова И.А. Алгоритм динамической классификации потоков в мультисервисной SDN-сети / В. А. Маньков, И. А. Краснова. // T-Comm: Телекоммуникации и транспорт. 2017. Т.11, №12. С. 37-42.

8. Олизарович Е. В. Метод автоматизации построения программно-конфигурируемых сетей / Е. В. Олизарович, А. И. Бражук // Вестник Гродзенского государственного университета им. Я.Купалы. 2013. №3(159).

С. 128-134.

9. Орлов Є. В. Програмно-конфігуровані мережі: архітектура, міжнародна стандартизація // Наукові записки Українського науково-дослідного інституту зв'язку. 2014. №4. С.85-92

10. Преимущества интеграции VMware vSphere с SDN платформой Huawei Cloud Fabric в ЦОД // Режим доступу: https://e.huawei.com/kz/publications/region/ru/product-insights/huawei-digital-transformation/cover/VMware-vSphere-and-SDN-Platform

11. Сергеева Т. П., Тетёкин Н. Н. Методы повышения надежности в сетях SDN / Т. П. Сергеева, Н. Н. Тетёкин // T-Comm. 2014. №16. С. 53-57

12. Смелянский Р. В. Программно-конфигурируемые сети / Р.В. Смелянский // Открытые системы. 2012. № 9. Режим доступу: http://www.osp.ru/os/2012/09/13032491.

13. Солтан Д. Д., Філімончук Т. В. Метод управління трафіком SDN мережі / Проблеми інформатизації : тези доп. 7 міжнар. конф., 13-15 листопада 2019 року // Харків : 2019. С.29

14. Ateya A. A., Muthanna A., A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal. Retriefed from: https://doi.org/10.1016/j.jestch.2018.12.015

15. Djelloul H., Sabba S., Chikhi S. Binary bat algorithm for graph coloring problem // Complex Systems (WCCS), 2014 Second World Conference on 10 – 12 Nov, pp. 481,486, 2014. doi: https://doi.org/10.1109/ICoCS.2014.7060988

16. Krasov A.V., Levin M.V., Shterenberg S.I., Isachenkov P.A. Traffic flow management model in software-defined networks with unequal load metric // H&ES Research. 2016. Vol. 8, №. 4. Pp. 70-74.

17. Limoncelli T. A. OpenFlow: A Radical New Idea in Networking / Thomas A. Limoncelli // Communications of the ACM. N. Y., 2012. Т. 55, № 8. P. 42-47.

18. Lin R., Ye Z. A bat algorithm for SDN network scheduling. EURASIP Journal on Wireless Communications and Networking. https://doi.org /10.1186/s13638-018-1145-y

19. Mirjalili, S. Salp swarm algorithm: a bio-inspired optimizer for engineering design problems / Mirjalili, S., Gandomi, A.H., Mirjalili, S.Z., Saremi, S., Faris, H. and Mirjalili, S.M. // Advances in Engineering Software, vol. 114, pp. 163–191, Dec. 2017.

20. Nakamura N. Haptic information presentation system and method / N. Nakamura, Y. Fukui, M. Sakai, N. Koda, Y. Iizuka // U.S. Patent Application, №. 15/285, 2017.

21. Nakamura R., Pereira L., Costa K., Rodrigues D., Papa J.P., Yang X. A binary bat algorithm for feature selection. In: Graphics, Patterns and Images / (SIBGRAPI), 2012 25th SIBGRAPI Conference On, pp. 291–297 (2012). IEEE

22. Oorschot P., Wan T. A Framework and Comparative Analysis of Control Plane Security of SDN and Conventional Networks. Retrieved from: https://www.researchgate.net/publication/315489682_A_Framework_and_Comparative_Analysis_of_Control_Plane_Security_of_SDN_and_Conventional_Network

23. Raza M., Shyamala C., Robertson B. A Comparison of Software Defined Network Implementation Strategies // Procedia Computer Science, №32, 2014. P. 1051-1055

24. Sakic E. Response Time and Availability Study of RAFT Consensus in Distributed SDN Control Plane / Sakic, E. and Kellerer, W. // IEEE Transactions on Network and Service Management. 2018. Vol. 15, №. 1. P. 304–318

25. Sayed G. I. A novel chaotic salp swarm algorithm for global optimization and feature selection / G. I. Sayed, G. Khoriba, M. H. Haggag // Applied Intelligence, pp. 1–20, 2018.

26. Shortle J.F. Fundamentals of queueing theory / Shortle J.F., Thompson J.M., Gross D., Harris C.M. // John Wiley & Sons., 2018. 324 p.

27. Skiadas, C. H. Handbook of applications of chaos theory / Skiadas, C. H. and Skiadas, C. eds. // CRC Press, Dec. 2017.

ДОДАТОК А

Графічний матеріал атестаційної роботи

Харківський університет радіоелектроніки
Кафедра ЕОМ

Метод управління трафіком SDN-мережі

Атестаційна робота
Другий (магістерський) рівень

Автор:
Солтан Д.Д.,
студ. гр. СПм-18-1

Керівник:
Філімончук Т.В.,
к.т.н., доц. каф. ЕОМ

Мета і задачі роботи

- Метою роботи є дослідження та систематизація теоретичних даних щодо управління трафіком SDN-мереж, існуючих проблем даної технології та моделювання можливих шляхів вирішення цих проблем.
- Для досягнення поставленої мети в роботі сформульовано та вирішено наступні задачі:

- дослідити архітектуру SDN та галузі застосування;
- порівняти технологію SDN та традиційні мережі;
- дослідити алгоритми та методи управління трафіком мережі SDN;
- систематизувати отримані дані;
- навести результати практичного використання досліджених алгоритмів та методів для управління трафіком мережі SDN.

Архітектура SDN

Площина управління	• управління пристроєм
Площина контролю	• управління трафіком
Площина передачі	• передача трафіку

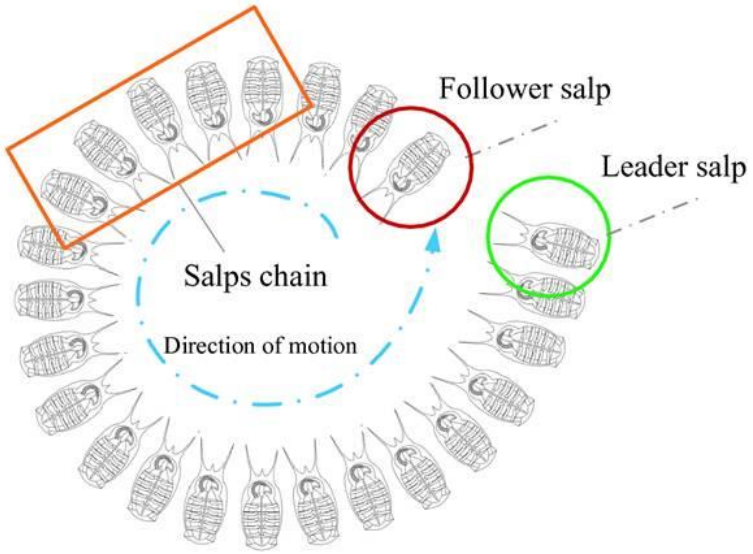
Порівняння технології SDN та традиційних мереж

Параметр	Звичайні мережі	SDN
Вигляд мережі	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
Адаптація нових технологій	Відповідно до потреб постачальника	Відповідно до потреб користувача

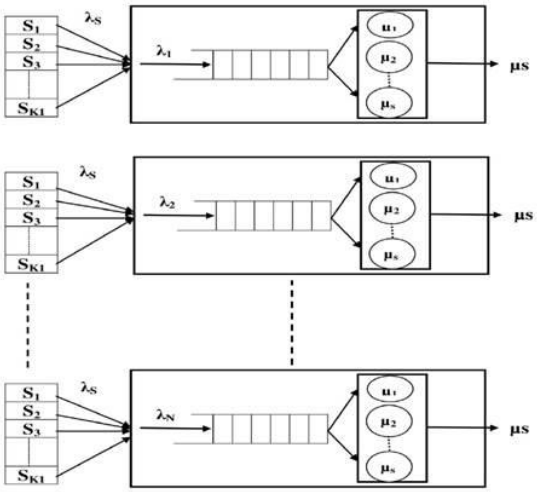
Застосування SDN для підвищення безпеки Ethernet

- Можливість видаляти широкомовні повідомлення з мережі, не впливаючи на продуктивність чи функціональність
- Нема потреби у використанні VLAN
- Можливість фізично ізолювати мережу управління від керованої мережі даних

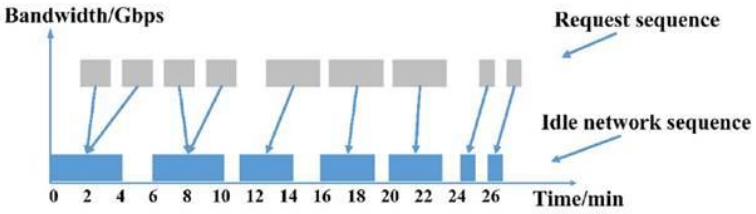
Алгоритм рою сальп для управління трафіком SDN



Математична модель мульти контролера SDN



Методи вирішення задачі мультиплікативного рюкзака для мережі SDN



Sample idle network sequence [4 4 3 3 3 1 1]

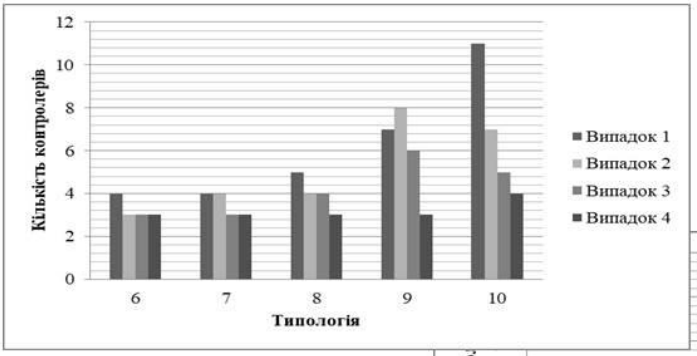
Sample request sequence [1 1 2 2 2 2 3 3 3]

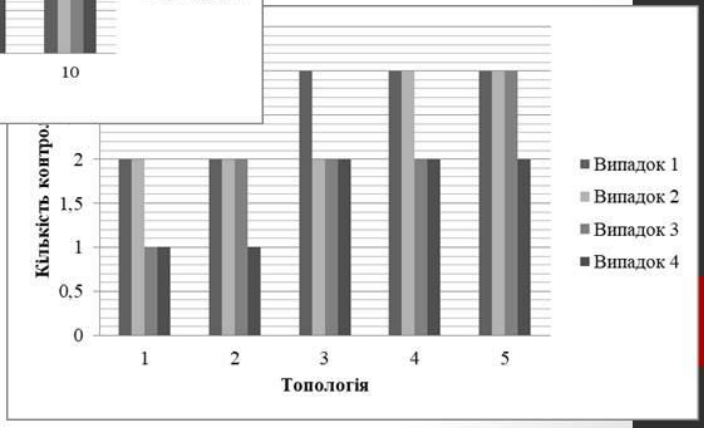
Алгоритм летючої миші для мережі SDN

Основні припущення алгоритму:

- усі летючі миші одночасно виявляють різницю між ціллю та фоновою перешкодою за допомогою ехолокації;
- положення летючої миші – x_i, політ з будь-якою швидкістю v_i, пошук мішені з фіксованою частотою f_{min}, змінна довжина хвилі l та гучність A_i;
- гучність змінюється багатьма способами, припускаючи, що вона змінюється від максимального значення (додатного) A_0 до фіксованого мінімального значення A_{min};
- тривимірна топографія та часові затримки ігноруються.

Застосування хаотичного алгоритму рою сальп для SDN





Модифікований алгоритм летючої миші для управління трафіком SDN

№	Оптимальне рішення	Послідовність алгоритму летючої миші	Рішення алгоритму летючої миші	Співвідношення
1	19	[7,6,4,2,2,0,3,5,1]	17	89,47%
2	23	[3,6,2,1,2,7,5,4,2]	23	95,83%
3	27	[6,3,4,0,1,2,7,1,0]	27	96,42%
4	15	[7,6,2,2,4,0,0,3,5]	14	87,5%
5	24	[1,6,2,2,7,1,5,4,3]	24	85,71%
6	21	[5,7,2,3,5,1,2,4,1]	21	95,45%

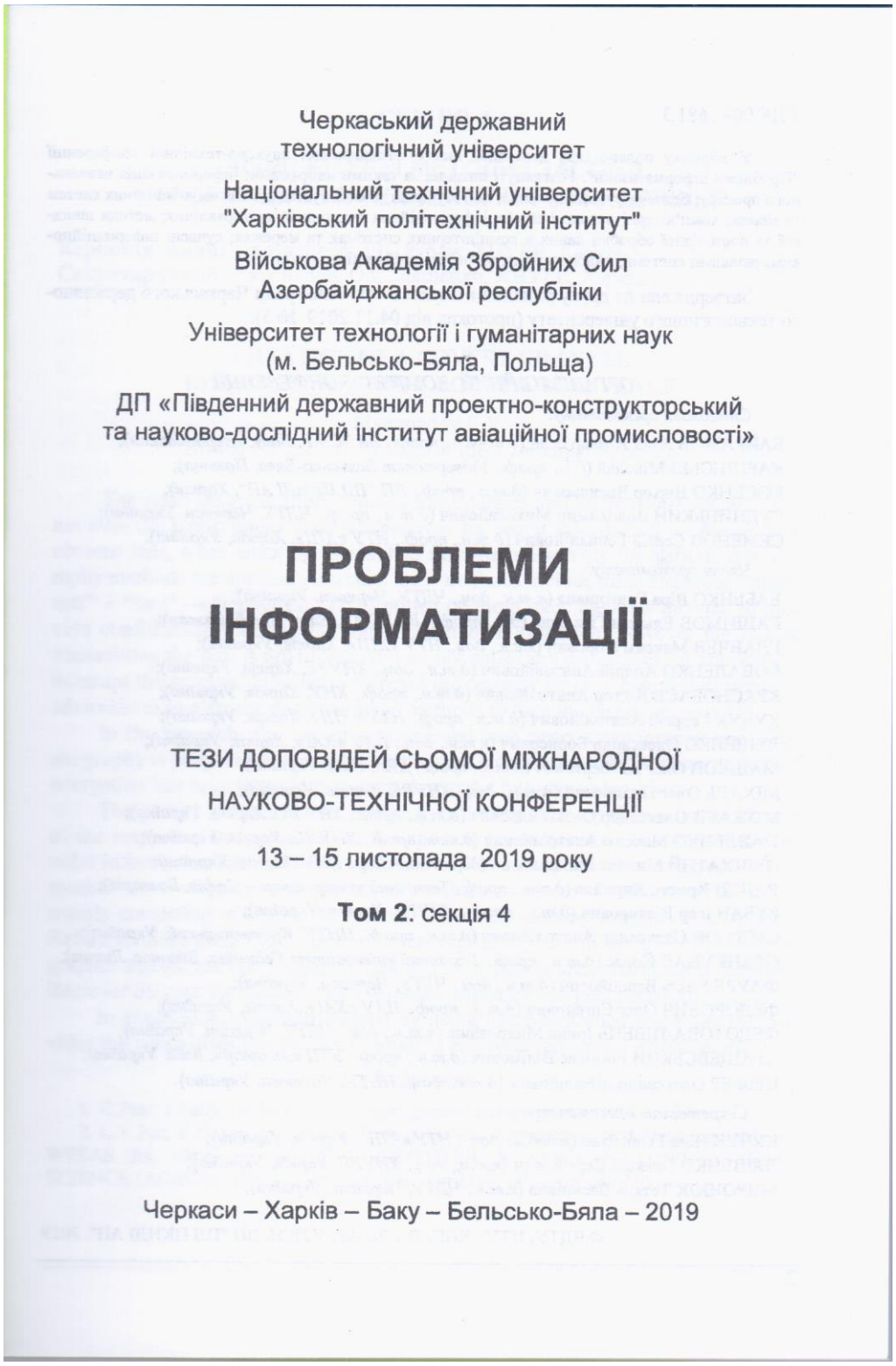
Висновки

Було запропоновано можливі шляхи підвищення безпеки технології Ethernet за допомогою технології SDN, модель мульти контролера для мережі SDN та алгоритм рою сальп для мережі SDN, застосування хаотичного алгоритму рою сальп та модифікованого алгоритму летючої миші для управління трафіком SDN-мережі.

Результати дослідження можуть бути запроваджені при оптимізації локальних мереж. Також результати можуть бути застосовані у навчальному процесі університету, зокрема при вивченні дисциплін «Комп'ютерні системи» та «Комп'ютерні системи та мережі».

ДОДАТОК Б

Напрацьовані матеріали за даною тематикою



Черкаський державний
технологічний університет

Національний технічний університет
"Харківський політехнічний інститут"

Військова Академія Збройних Сил
Азербайджанської республіки

Університет технології і гуманітарних наук
(м. Бельсько-Бяла, Польща)

ДП «Південний державний проектно-конструкторський
та науково-дослідний інститут авіаційної промисловості»

ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ

ТЕЗИ ДОПОВІДЕЙ СЬОМОЇ МІЖНАРОДНОЇ
НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

13 – 15 листопада 2019 року

Том 2: секція 4

Черкаси – Харків – Баку – Бельсько-Бяла – 2019

МЕТОД УПРАВЛІННЯ ТРАФІКОМ SDN МЕРЕЖІ

Філімончук Т.В., Солтан Д.Д.
Харківський національний університет радіоелектроніки, Харків, Україна

Програмно-конфігурована мережа (SDN) – це архітектура, що дозволяє контролювати комп'ютерну мережу за допомогою програмного забезпечення [1]. За допомогою SDN оператори можуть послідовно та комплексно управляти всією мережею незалежно від технології, яка використовується в ній. За управління трафіком в мережі SDN відповідає контролер, який передає відповідні вказівки комутаторам для організації політики управління трафіком [2]. Але на даний час в SDN мережі не існує готових політик управління трафіком, тому що все залежить від завдань, які накладаються на мережу та можливостей самої мережі.

Метою доповіді є впровадження нових підходів для розподілу мережного трафіка в комп'ютерних мережах за рахунок використання комплексної метрики для визначення найкоротшого маршруту передачі даних.

В доповіді розглядаються існуючі методи управління трафіком, їх недоліки та переваги. На даний час для організації обслуговування в мультисервісних мережах використовують вендори, які реалізують стандартизовані протоколи. Ці протоколи засновані на визначенні найкоротшого маршруту або його мінімальної вартості. Алгоритм визначення метрики в цих алгоритмах використовує один параметр для прийняття рішення, наприклад, довжина лінії зв'язку, затримка вздовж фізичного каналу, завантаження фізичного або логічного канал та ін. Але це призводить до низької ефективності використання протоколів маршрутизації в мультисервісних мережах. Для вирішення цієї проблеми пропонується використовувати концепцію маршрутизації із забезпеченням необхідної якості обслуговування. Ця концепція полягає у визначенні такого маршруту між джерелом та адресатом, при якому будуть виконуватись вимоги по максимізації якості обслуговування комп'ютерної мережі (використання комплексної метрики) [3, 4].

Список літератури

1. Лозинская В.Н., Долгих И.П. Особенности управления сетью оператора связи на основе SDN-решений. Сборник научных трудов ДОНИЖТ. 2017. №46. С. 10–17.

2. Бовда Е.М. Метод управління перерозподілом навантаження в sdn мережах. Збірник наукових праць ВІТІ. 2017. №2. С. 6–15.

3. Филимончук Т.В., Волк М.А. Разработка модифицированного метода обратного заполнения Backfill для консервативного резервирования. Системи обробки інформації. Харків: ХУПС, 2017. №1(147). С. 33–37.

4. Филимончук Т.В., Волк М.А., Казмина Д.Р., Ольшанская Т.И., Рисухин М.В. Модифицированная информационная технология распределения заданий на ресурсы для систем облачных вычислений. Сучасний стан наукових досліджень та технологій в промисловості. 2019, №1(7). С. 121-128. DOI: https://doi.org/10.30837/2522-9818.2019.7.121.

ЗМІСТ

Наукове видання

ПРОБЛЕМИ ІНФОРМАТИЗАЦІЇ

Тези доповідей
сьомої міжнародної науково-технічної конференції
13 – 15 листопада 2019 року
Том 2: секція 4

Відповідальний за випуск *В. М. Рудницький*
Технічний редактор *І. А. Лєбедева*
Комп'ютерне складання та верстання *Н. Г. Кучук*

Підписано до друку 06.11.2019 Формат 60 × 84/16
Ум.-вид. арк. 8,25. Тираж 200 пр. Зам. 1107-19
Адреса оргкомітету: бульвар Шевченка 460, м. Черкаси, 18006, Україна
Черкаський державний технологічний університет

Віддруковано з готових оригінал-макетів у друкарні ФОП Петров В.В.
Єдиний державний реєстр юридичних осіб та фізичних осіб-підприємців.
Запис № 24800000000106167 від 08.01.2009.
61144, м. Харків, вул. Гв. Широнінців, 79в, к. 137, тел. **(057) 778-60-34**
e-mail: **bookfabrik@mail.ua**