Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод моніторингу мережного трафіку на основі технологій Big Data

(тема)

Виконав:

студент II курсу, групи КСМм-20-1

Гнип А.К.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: проф. Торба А.А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 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 Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Гнипу Артему Костянтиновичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод моніторингу мережного трафіку на основі технологій Big Data

затверджена наказом по університету від “ 05 ” листопада 2021 р. № 1656 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи 1) технологія обробки: розподілена; 2) підтримувані системи: Apache Hadoop; 3) режими: пакетний, потоковий.

4. Перелік питань, що потрібно опрацювати у роботі
1) огляд технологій використання великих даних у моніторингу комп’ютерних мереж;
2) аналіз особливостей застосунків моніторингу і аналізу мережного трафіка;
3) розробка методу моніторингу;
4) експериментальні дослідження;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд – презентація – 12 слайдів

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 технологій використання великих даних	09.11.21-12.11.21	
	для моніторинга комп'ютерних мереж		
2	Аналіз особливостей застосунків NTMA	13.11.21-18.11.21	
3	Розробка методу моніторингу	19.11.21-29.11.21	
4	Проведення експериментів	30.11.21-03.12.21	
5	Оформлення матеріалів кваліфікаційної роботи	04.12.21-07.12.21	
6	Подання кваліфікаційної роботи керівникові та	08.12.21-09.12.21	
	її попередній захист		
7	Подання кваліфікаційної роботи на	10.12.21-11.12.21	
	рецензування		

Дата видачі завдання 08 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Торба А.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 72 с., 22 рис., 10 табл., 1 дод., 24 джерела.

АНАЛІЗ ТРАФІКУ, ВЕЛИКІ ДАНІ, МАШИННЕ НАВЧАННЯ, МЕРЕЖЕВІ ВИМІРЮВАННЯ, ПЛАТФОРМИ ВЕЛИКИХ ДАНИХ

Метою кваліфікаційної роботи є розробка методу моніторингу трафіку комп’ютерної мережі на основі технологій великих даних.

У ході виконання кваліфікаційної роботи запропоновано метод моніторингу на основі збору великих даних та формування статистики трафіку за допомогою значень лічильників. Для підтвердження ефективності цього методу виконана реалізація з використанням спеціальних інструментів Big Data. Також досліджується використання методів аналізу великих даних для аналізу в реальному часі великої кількості збережених історичних даних. Результати аналізу великих даних можуть допомогти при розв’язанні задач ТЕ, таких як балансування навантаження комутатора та відмовостійкість площини даних.

ABSTRACT

Master’s thesis: 72 pages, 22 figures, 10 tables, 1 appendix, 24 sources.

BIG DATA, BIG DATA PLATFORMS, MACHINE LEARNING, NETWORK MEASUREMENTS, TRAFFIC ANALYSIS.

The purpose of the qualification work is to develop a method of monitoring computer network traffic based on big data technologies.

During the qualification work, a detailed method of monitoring for the Big Data collection and generation of traffic statistics using measured values is proposed. To confirm the effectiveness of this method, the implementation was performed using special tools Big Data. The Big Data analysis methods for real-time analysis of a large amount of stored historical data is also investigated. The results of Big Data analysis can help solve TE problems such as switching load balancing and data plane fault tolerance.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ER – тип формального опису «сутність–відношення» (англ., Entity-Relation)

HDFS – розподілена файлова система Hadoop (англ., Hadoop Distributed File System)

KDD – процес видобування знань з великих даних (англ., Knowledge Discovery in Data)

NTMA – моніторинг та аналіз мережевого трафіку (англ., Network Traffic Monitoring and Analysis)

OD – взаємодія за типом «пункт відправки–пункт призначення» (англ., Origin–Destination)

OF – протокол OpenFlow

QoE – якість сприйняття послуг (англ., Quality of Experience)

QoS – якість послуг (англ., Quality of Service)

SDN – програмно-визначена мережа (англ., Software-defined networking)

TE – інженерія трафіку (англ., Traffic Engineering)

TM – матриця трафіку (англ., Traffic Matrix)

ВСТУП

Моніторинг та аналіз мережевого трафіку (англ., NTMA – Network Traffic Monitoring and Analysis) є ключовим компонентом для управління мережею, особливо для забезпечення правильної роботи великих мереж, таких як Інтернет. Оскільки складність Інтернет-послуг і обсяг трафіку продовжують зростати, стає важко розробляти масштабовані програми NTMA. Нерідко вимогами до таких програм є робота у реальному часі та підтримка масштабованості. Механізми виявлення аномалій і інформаційної безпеки вимагають швидко виявляти та реагувати на непередбачувані події під час обробки мільйонів різнорідних подій. Нарешті, система має збирати, зберігати та обробляти величезні набори історичних даних для ретроспективного аналізу. Це саме та сукупність обставин, з якою стикається методологія великих даних: обсяг, швидкість, різноманітність і правдивість. В даній кваліфікаційній роботі поєднуються NTMA та великі дані. Перш за все потрібно визначити, які використовуються підходи до великих даних, щоб зрозуміти, наскільки потенціал великих даних враховано у NTMA. На цьому етапі доцільно зосередитися на підходах та технологіях для управління великими даними NTMA, а також коротко обговорити аналітику великих даних (наприклад, машинне навчання) заради NTMA.

Об’єктом дослідження є програмно-визначені мережі (англ., SDN – Software-defined networking). Ця парадигма долає обмеження традиційних мереж завдяки відокремлення площини керування від площини даних. Логіка мережі переміщується до компонента, що називається контролером, який керує пристроями в площині даних. Для реалізації цієї архітектури стало нормою використовувати протокол OpenFlow (OF), який визначає кілька лічильників, що обслуговуються мережевими пристроями. Ці лічильники є відправною точкою в інженерії трафіку (англ., TE – Traffic Engineering). TE контролює декілька параметрів мережі, включаючи використання пропускної

здатності мережі. Важливим завданням для ТЕ є збір і генерація статистичних даних про використання пропускної здатності для реалізації моніторингу та аналізу трафіку. Це завдання ускладнюється, якщо потрібен детальний моніторинг. Подібний «дрібнозернистий» моніторинг є корисним у завданнях керування мережею, таких як надання мережі, планування потужності, балансування навантаження і виявлення аномалій.

Оскільки лічильники оновлюються для кожного пакету, що перетинає комутатор, їх потрібно отримувати потоковим способом. Цей сценарій передбачає використання методів потокової передачі великих даних для збору та обробки значень лічильника. Тому пропонується підхід, заснований на детальному методі моніторингу великих даних для збору та створення статистики трафіку за допомогою значень лічильників. При цьому може суттєво використовуватися ТЕ. Такий підхід може надати більш детальне уявлення про використання мережевих ресурсів, завдяки індивідуальному та агрегованому статистичному аналізу споживання пропускної здатності. Результати експерименту свідчать про ефективність запропонованого способу.

1 ВИКОРИСТАННЯ МЕТОДОЛОГІЇ ВЕЛИКИХ ДАНИХ ДЛЯ МОНІТОРИНГУ МЕРЕЖЕВОГО ТРАФІКА

1.1 Поєднання NTMA та Big Data

Розуміння того, як використовуються Інтернет-послуги та як вони працюють, має вирішальне значення для життя людей. Одне з найважливіших місць у цьому посідають проблеми моніторингу та аналізу мережевого трафіка (NTMA). Діапазон застосунків варіюється від надання огляду мережевого трафіку до виявлення аномалій і невідомих атак під час функціонування систем. Такі застосунки збирають історичні дані, необхідні для підтримки розробки трафіку та усунення несправностей, допомагаючи спланувати еволюцію мережі та визначити першопричину проблем. Можна вважати програми NTMA наріжним каменем, який гарантує, що послуги, які підтримують наше повсякденне життя, завжди є доступними та працюють, як того очікують.

Моніторинг і аналіз трафіку є складним завданням. Величезний обсяг трафіку, швидкість систем передачі, природна еволюція послуг і атак, а також різноманітність джерел даних і методів для отримання вимірювань – це лише деякі з проблем, з якими стикаються програми NTMA. Із зростанням складності мережі більше точок спостереження стають доступними дослідникам, потенційно дозволяючи збирати й оцінювати різнорідні дані. Ця тенденція ускладнює розробку масштабованих і розподілених застосунків і вимагає ефективних механізмів для онлайн-аналізу великих потоків вимірювань. Більше того, зі зниженням цін на зберігання стає можливим створення масивних історичних наборів даних для ретроспективного аналізу.

Ці проблеми є саме характерними для парадигми великих даних (англ., Big Data), тобто коли обсяг даних, швидкість, правдивість і різноманітність є ключовими проблемами, які дозволять видобувати значення з даних. Дійсно,

моніторинг та аналіз трафіку були одними з перших прикладів джерел великих даних, які з’явилися, і це створює проблеми з великими даними як ніколи.

Тому не дивно, що дослідники вдаються до технологій великих даних для підтримки застосунків NTMA [1, 2, 3]. Деякими прикладами технологій, що допомагають програмам обробляти набори даних, які в іншому випадку були б нерозв’язними є: розподілені файлові системи, наприклад, HDFS – Hadoop Distributed File System; платформи великих даних, наприклад, Hadoop і Spark; розподілені механізми машинного навчання та обробки графіків, наприклад, MLlib і Apache Giraph.

Проте не зовсім зрозуміло, чи повністю використовують програми NTMA потенціал технологій великих даних. Отже, можна вважати доцільним об’єднати дослідження NTMA та великі дані. Різні дослідники задокументували досягнення в області технологій великих даних [4, 5, 6], методів, що підтримують NTMA (наприклад, машинне навчання для NTMA) [7, 8], або конкретні програми NTMA [9, 10, 11], але ж немає систематичних досліджень, які б описували, як NTMA та великі дані поєднуються для повного використання потенціалу мережевих даних. Виходячи з цього, потрібно визначити, наскільки дослідниками NTMA використовується потенціал технологій великих даних.

1.2 Методологія великих даних

1.2.1 Основні визначення

У літературі з’явилося багато визначень великих даних. Ху та ін. [12] стверджують, що «великі дані означають не тільки великий обсяг даних, але й інші характеристики, такі як різноманітність і швидкість передачі даних». Вони організовують визначення в трьох категоріях: архітектурні, порівняльні або атрибутивні.

І архітектурне, і порівняльне визначення є абстрактними. Відповідно до архітектурного визначення, можна зіткнутися з проблемою великих даних, коли набір даних буде таким, що «традиційні підходи» не здатні будуть зберігати та обробляти його. Аналогічно, порівняльні визначення характеризують великі дані, порівнюючи властивості даних з можливостями традиційних інструментів баз даних. Вони покладаються на атрибутивні визначення [13, 14] та описують великі дані за характерними ознаками як даних, так і процесу аналізу самих даних. Вони виступають за те, що зараз відомо як «V’s», – наприклад, обсяг, швидкість, різноманітність, правдивість, значимість (англ., відповідно, Volume, Velocity, Variety, Veracity, Value).

Інші дослідження, спрямовані на технології великих даних [6, 12, 15] та їхню аналітику [4, 5] розділяють цей погляд. В даній роботі за основу буде взято визначення «5-V», оскільки воно надає конкретні критерії, які характеризують проблеми з великими даними, з якими має справу NTMA. Таким чином, можна вважати, що проблема належить до класу великих даних, якщо набори даних великі (тобто об’єм) і їх потрібно фіксувати та аналізувати з високими темпами або в режимі реального часу (тобто швидкість). Дані потенційно надходять з різних джерел (тобто різноманітних), які поєднують структуровані дані (такі як колончаті бази даних), неструктуровані дані (такі як журнали сервера) і напівструктуровані дані (такі як документи XML або JSON). Більше того, дані можуть бути різної якості, з невизначеністю, яка характеризує дані (тобто правдивість). Нарешті, аналіз даних приносить переваги користувачам і підприємствам (тобто цінність) – нові ідеї витягуються із вхідних даних, щоб збільшити їх цінність, бажано з використанням автоматичних методологій.

1.2.2 Видобування знань з великих даних

Процес видобування знань з великих даних (англ., KDD – Knowledge Discovery in Data) зазвичай розглядають як послідовність операцій над

даними: збір, відбір, попередня обробка, перетворення, видобуток, оцінка, інтерпретація та візуалізація. З системної точки зору, окремі автори [6] реорганізували ці операції на три групи: введення, аналіз і виведення.

Введення даних охоплює процес управління даними, від збору необроблених даних до доставки даних у відповідних форматах для подальшого видобутку. Воно включає попередню обробку, яка є початковими кроками для підготовки даних, з інтеграцією різнорідних джерел і очищенням від помилкових даних.

Методи аналізу даних отримують підготовлені дані та витягують інформацію, тобто моделі для класифікації, приховані закономірності, відносини, правила тощо. Методи варіюються від статистичного моделювання та аналізу до машинного навчання та алгоритмів аналізу даних.

Виведення даних завершує процес, перетворюючи інформацію в знання. Він включає кроки для вимірювання якості інформації, відображення інформації в стислому форматі та надання допомоги аналітикам в інтерпретації результатів. Цей крок не оцінюється в даній роботі.

Визначені «класичні» стадії можна далі згрупувати в управління даними та аналітику (тобто аналіз даних і виведення). Слід зауважити, що на відміну від [5], в цій кваліфікаційній роботі термін «аналіз даних» означатиме вилучення інформації з даних, тоді як «аналітика даних» означатиме весь процес, від аналізу даних до отримання знань. Відповідна схема показана на рисунку 1.1 і буде використовуватися для характеристики документів NTMA відповідно до кількох етапів процесу KDD.

Процес KDD, описаний вище, застосовується до будь-якої аналітики даних. Проте характеристики великих даних накладають фундаментальні проблеми для методологій на кожному кроці. Наприклад, процес управління даними, природно, зіштовхнеться з набагато складнішими завданнями з великими даними, враховуючи обсяг, швидкість і різноманітність даних. Таким чином, управління даними має вирішальне значення для великих даних, оскільки воно відіграє ключову роль у зменшенні обсягу та складності

даних. Крім того, це впливає на фази аналізу та результатів, зокрема з точки зору швидкості аналізу та цінності результатів.

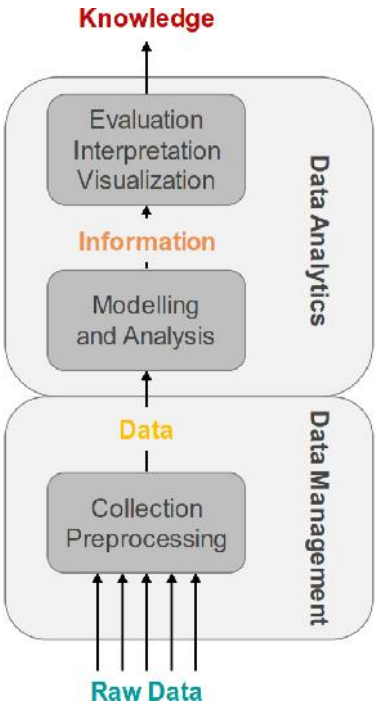

Рисунок 1.1 – Схема процесу KDD

Проблеми з великими даними (наприклад, «5-V») вимагають підходу, який розглядає весь процес KDD у комплексній аналітичній структурі. Така структура повинна включати моделі програмування, які дозволяють реалізувати логіку програми, що охоплює повний цикл KDD.

1.2.3 Моделі та платформи програмування

Дотримуючись існуючих таксономій [6, 15], можна виділити чотири типи моделей обробки великих даних:

- загального призначення – платформи для обробки великих даних, які роблять невеликі припущення щодо характеристик даних і виконуваних алгоритмів;
- SQL-подібні платформи, що фокусуються на масштабованій обробці

структурованих і табличних даних;

- обробка графів – платформи, які зосереджені на обробці великих графів;

- потокова обробка – платформи, які обробляють великомасштабні дані, які постійно надходять до системи потоковим способом.

Рисунок 1.2 демонструє цю таксономію з прикладами систем. MapReduce, Dryad, Flink і Spark належать до першого типу. Hive, HAWQ, Apache Drill і Tajo належать до типу SQL. Pregel, GraphLab дотримуються моделей обробки графів і, нарешті, Storm і S4 є прикладами останнього типу.

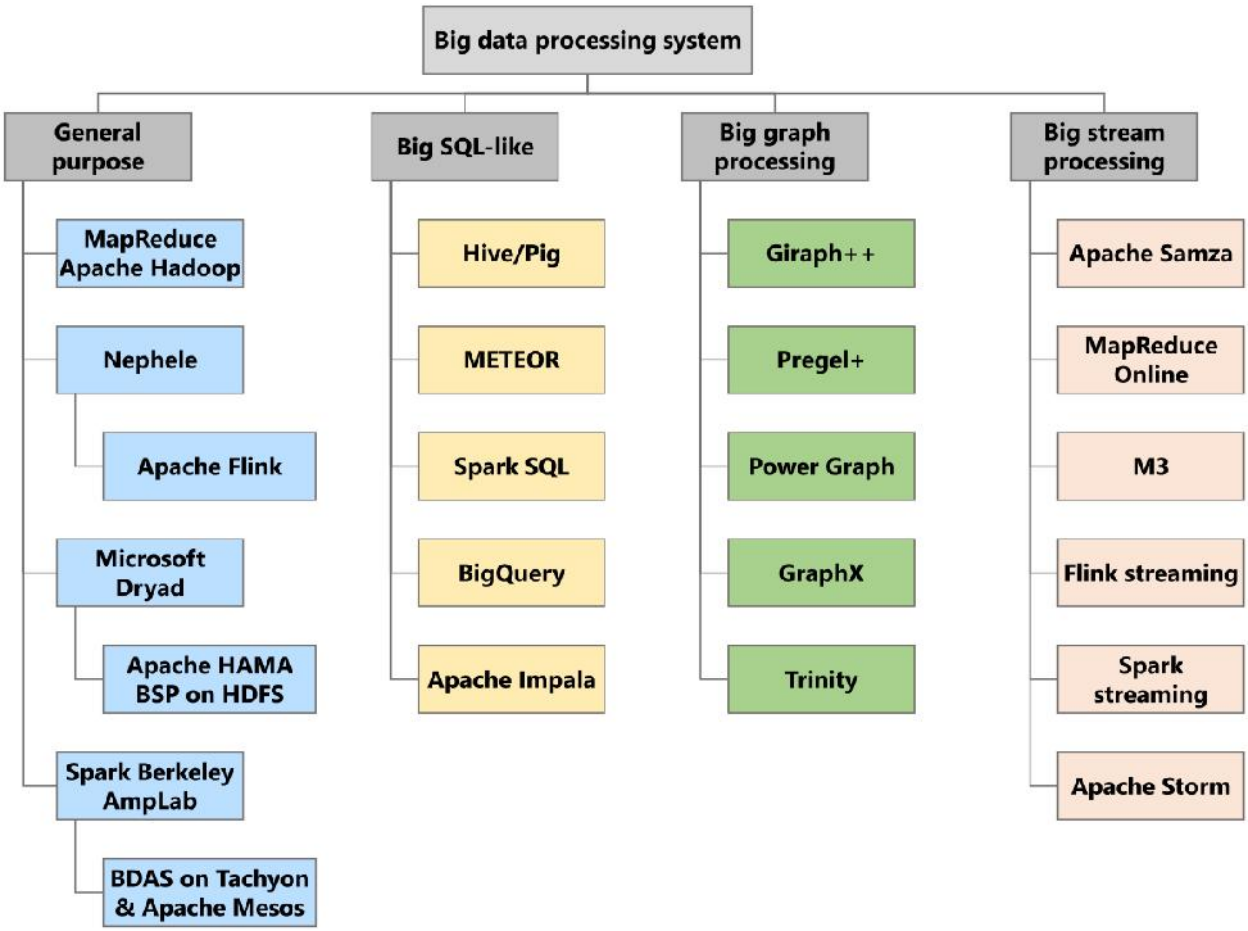

Рисунок 1.2 – Системи обробки великих даних

Найпоширенішим рішенням серед універсальних платформ великих даних є Hadoop. Враховуючи його важливість, на рисунку 1.3 наведено деякі

деталі про його компоненти. Система Hadoop складається з ядра Hadoop, MapReduce і розподіленої файлової системи Hadoop (HDFS). YARN – це менеджер ресурсів за замовчуванням, який надає доступ до ресурсів кластера для кількох конкуруючих завдань. Інші менеджери ресурсів (наприклад, Mesos) і механізми виконання (наприклад, TEZ) також можна використовувати, наприклад, для надання ресурсів Spark.

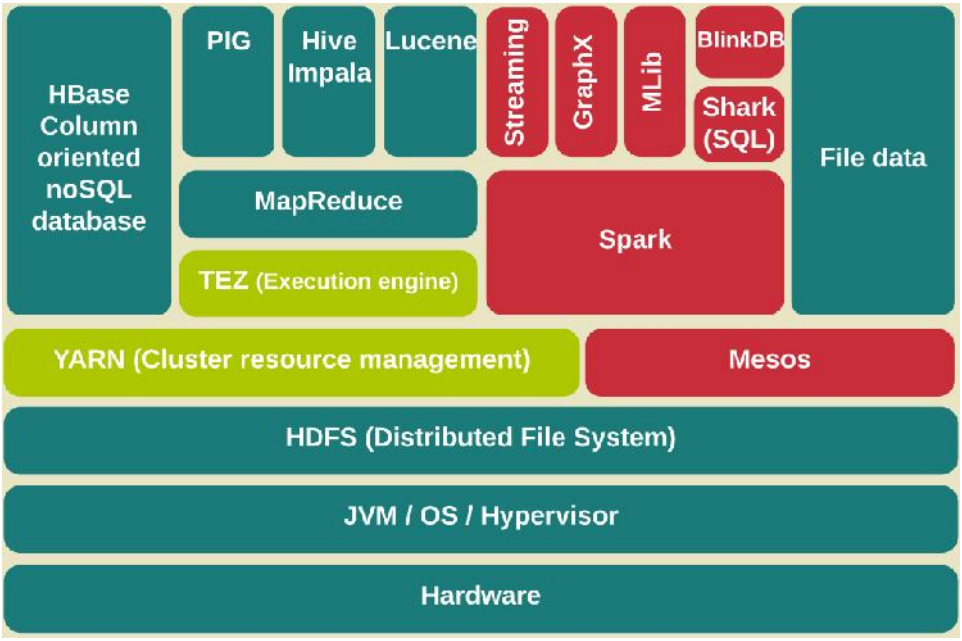

Рисунок 1.3 – Стек Hadoop 2.0 із Spark (на основі [14])

Spark представлено в Hadoop, починаючи з версії 2.0, для подолання обмежень в парадигмі MapReduce. Spark заснований на представленнях даних, які можна трансформувати в кілька кроків, ефективно перебуваючи в пам’яті. На відміну від цього, парадигма MapReduce покладається на основні операції («Map» та «Reduce» – зведення та зменшення), що застосовуються до пакетів даних, які читаються та зберігаються на диску. Spark набирає обертів у непакетних сценаріях, наприклад, ітераційних застосунках із великими даними в режимі реального часу, а також у пакетних програмах, які неможливо вирішити за кілька етапів операцій зі зведенням/зменшенням.

Існують пропозиції щодо запуску поверх Hadoop різних мов і систем

високого рівня, наприклад Sawzall від Google, Pig Latin від Yahoo, Hive від Facebook, а також SCOPE від Microsoft. Крім того, спільнотою були запроваджені кілька бібліотек, таких як Mahout для MapReduce і MLlib для Spark, з метою вирішення проблем або заповнення прогалин в оригінальній екосистемі Hadoop. Нарешті, екосистема була доповнена інструментами, орієнтованими на конкретні моделі обробки, такими як GraphX і Spark Streaming, які підтримують обробку графів і потоків, відповідно.

Окрім дистрибутивів Apache Hadoop, власні платформи пропонують різні функції для обробки даних та керування кластерами. Деякі з таких рішень включають Cloudera CDH, Hortonworks HDP і MapR Converged Data Platform.

1.3 Застосунки NTMA

1.3.1 Таксономія

Щоб класифікувати застосунки NTMA, які використовують підходи до великих даних, взято підмножину таксономії програм керування мережею [16]. У вказаному дослідженні наведено вісім категорій, які визначені відповідно до кінцевої мети програми керування, а саме: прогнозування трафіку, класифікація трафіку, керування помилками, безпека мережі, контроль перевантаження, маршрутизація трафіку, керування ресурсами та керування QoS/QoE.

Слід врахувати роботи, які підходять до перших чотирьох категорій з двох причин. По-перше, тоді як систематика [16] підходить для опису програм керування мережею, рівень залежності таких додатків від NTMA значно різниться. Маршрутизація трафіку та управління ресурсами здаються менш залежними від великомасштабних вимірювань, ніж, наприклад, прогнозування трафіку та безпека. По-друге, інформаційні джерела щодо використання підходів великих даних для контролю перевантажень,

маршрутизації трафіку, управління ресурсами та управління QoS/QoE майже відсутні. Можна припустити, що або великомасштабні набори даних, націлені на проблеми в цих категоріях, недоступні, або дослідники не виявили потенціалу для застосування підходів до великих даних у цих сценаріях.

1.3.2 Програми NTMA

Таблиця 1.1 показує категорії, використані в цій роботі.

Прогноз трафіку складається з оцінки майбутнього стану мережевих посилань. Він служить будівельним блоком для розробки трафіку, допомагаючи визначити, наприклад, найкращий момент для розгортання більшої потужності в мережі, з метою збереження рівня QoS.

Прогноз трафіку часто розглядається як проблема прогнозування часових рядів [16]. Тут використовуються як класичні методи прогнозування (наприклад, методи ARIMA або SARIMA), так і машинне навчання (наприклад, глибокі нейронні мережі). Задача зазвичай формулюється як оцінка обсягів трафіку на основі попередніх вимірювань. Однак зміни в поведінці мережі роблять такі оцінки дуже складними. Нові послуги або раптові зміни в конфігураціях послуг (наприклад, розгортання нових програм, що потребують пропускної здатності) створюють серйозні проблеми для підходів прогнозування трафіку.

Класифікація трафіку спрямована на визначення послуг, які створюють трафік. Це важливий крок для управління та моніторингу мережі. Операторам потрібна інформація про послуги, наприклад, щоб зрозуміти їхні вимоги та їхній вплив на загальну продуктивність мережі.

Класифікація трафіку добре спрацьовувала, коли перевірялась інформація в мережевих і транспортних протоколах. Наприклад, Інтернет-послуги раніше ідентифікувалися просто перевіркою номерів портів TCP/UDP. Однак класифікація трафіку вже не є простим завданням. По-

перше, кількість Інтернет-сервісів велика і продовжує збільшуватися. По-друге, послуги необхідно ідентифікувати, спостерігаючи за невеликою кількістю інформації в мережі. По-третє, у пакетах залишається мало інформації, оскільки велика частка Інтернет-сервісів працює на основі кількох протоколів шифрування (наприклад, HTTPS через TCP). Нарешті, Інтернет-сервіси динамічні та постійно оновлюються.

Таблиця 1.1 – Категорії застосунків NTMA

Категорія	Опис
Прогноз трафіку	Обслуговування та планування мереж. Історично за допомогою прогнозування часових рядів.
Класифікація трафіку	Класифікація та розпізнавання потоків трафіку для різних цілей, наприклад, QoE, безпеки тощо.
Управління несправностями	Прогнозування та ізоляція несправностей і небажаної поведінки в мережах.
Безпека мережі	Захист мережі, реагування на зловмисні дії та загальні атаки (та їх запобігання).

До цього часу запропоновано багато підходів для виконання класифікації трафіку. Їх можна згрупувати відповідно до стратегії, яка використовується для класифікації пакетів:

- перевірка пакетів, при якій аналізується вміст пакетів з метою пошуку попередньо визначених повідомлень або «відбитків пальців» протоколу;

- керовані методи машинного навчання, які виділяють функції з трафіку і на етапі навчання створюють моделі для зв’язування значень функцій із послугами;

- методи машинного навчання без нагляду, які об’єднують трафік без попередніх знань про служби. Вони доречні для вивчення послуг, для яких навчальні дані недоступні;

- поведінкові методи, що визначають послуги на основі поведінки кінцевих вузлів [17]. Алгоритми спостерігають за трафіком для побудови моделей для вузлів, на яких працюють певні служби. Моделі описують, наприклад, до яких серверів звертаються клієнти, з якою швидкістю передачі даних, порядок зв’язку із серверами тощо.

Управління несправностями (збоями) – це набір завдань для прогнозування, виявлення, ізоляції та усунення несправностей у мережах. Мета – мінімізувати час простою. Управління збоями може бути проактивним, наприклад, коли аналітика прогнозує збої на основі вимірювань, щоб уникнути збоїв, або реактивним, наприклад, коли трафік і журнали системи оцінюються для розуміння поточних проблем. У будь-якому випадку ключовим кроком в управлінні несправністю є локалізація основної причини проблеми [16].

Збої у великих мережах можуть вплинути на різні елементи: наприклад, несправний маршрутизатор може перевантажити інші маршрути, створюючи таким чином ланцюжок збоїв у мережі. Різноманітні елементи мережі створюватимуть системні журнали, пов’язані з проблемою, і поведінка мережі може змінюватися в різних аспектах. Виявлення несправності часто досягається за допомогою методів виявлення аномалій, які визначають ненормальну поведінку в трафіку або незвичайні події в системних журналах. Однак аномалії можуть бути викликані також інцидентами безпеки або звичайними змінами моделей використання. Алгоритми аналітики часто використовують у поєднанні оцінку трафіку, системні журнали і активні вимірювання для того, щоб збільшити видимість мережі та спростити ідентифікацію основних причин проблем.

Для забезпечення кібербезпеки було запропоновано багато програм NTMA. Найпоширенішою метою є виявлення недоліків безпеки, вірусів і шкідливих програм, що дозволяє ізолювати заражені машини та вжити контрзаходів для мінімізації збитків. Грубо кажучи, існує два основних підходи до пошуку шкідливої мережевої активності:

- на основі сигнатур атак;
- на основі виявлення аномалій.

Методи на основі сигнатур базуються на тій ідеї, що для атак можна визначити «відбитки пальців». Рішення для моніторингу перевіряє вихідний трафік/журнали/події, шукаючи відомі повідомлення, якими обмінюються віруси, шкідливі програми чи інші загрози, або типові комунікаційні моделі атак, тобто подібні до поведінкових методів класифікації трафіку. Методи на основі сигнатур ефективні для блокування добре відомих атак, які є незмінними або повільно змінюються. Однак ці методи вимагають, щоб атаки були добре задокументовані.

Методи, засновані на виявленні аномалій [9], виходять з припущення, що атаки змінять поведінку мережі. Вони будують моделі, щоб узагальнити нормальну поведінку мережі за результатами вимірювань. Потім відстежується живий трафік і запускаються сповіщення, коли поведінка мережі відрізняється від базової лінії. Методи виявлення аномалій є привабливими, оскільки вони дозволяють завчасно виявляти невідомі загрози (наприклад, експлойти нульового дня). Однак ці методи можуть не виявляти прихованих атак (тобто помилкових негативів), які недостатньо великі, щоб порушити роботу мережі. Іноді вони також страждають від великої кількості помилкових результатів.

1.3.3 Проблеми з великими даними у застосунках NTMA

Можна стверджувати, що програми NTMA, що належать до наведених вище категорій, отримуватимуть прибуток від підходів до великих даних. Обробка таких вимірювань створює типові проблеми з великими даними (наприклад, «5-V»). Це підтверджується наступним.

Обсяг і швидкість (англ., Volume and Velocity) на прикладі класифікації трафіку: вона повинна виконуватися «на льоту» (наприклад, для пакетів), а вхідні дані зазвичай великі. Тут дослідники покладаються на різні стратегії

для виконання класифікації трафіку на високошвидкісних каналах, а деякі роботи застосовують алгоритми до сотень потоків Гбіт/с.

Різноманітність (англ., Variety): оскільки все більше і більше трафіку передається зашифрованим, алгоритми для виконання класифікації трафіку або управління помилками, наприклад, мають гіршу інформацію для роботи. Сучасні алгоритми спираються на різноманітні джерела – наприклад, поєднання DNS і вимірювань потоків. З метою отримання різноманітних сигналів про аномалії, методи виявлення аномалій зазвичай також застосовуються до різноманітних джерел (наприклад, траси трафіку, інформація про маршрути тощо).

З точки зору правдивості (англ., Veracity), знову можна посилатися на кібербезпеку. Зразки атак необхідні для навчання класифікаторів для визначення атак на реальні мережі. Виготовлення таких зразків є складним завданням. У той час як прості атаки можна відтворити в лабораторії, складні атаки важко відтворити. Або, що найгірше, вони моделюються нереалістичним способом, що обмежує аналіз.

Нарешті, значимість (англ., Value) є критичним у всіх вищезазначених програмах. Наприклад, у сфері кібербезпеки, одна атака, яка залишається непоміченою, може бути критичною для роботи мережі.

2 КОНЦЕПЦІЯ ПОБУДОВИ СИСТЕМИ МОНІТОРИНГУ ДАНИХ В МЕРЕЖІ SDN

2.1 Особливості мережі SDN

Основна ідея програмно-визначеної мережі (SDN) полягає в тому, щоб відокремити площину керування від площини даних. Мережні пристрої (комутатори та маршрутизатори), що належать до площини даних, працюють лише як пристрої пересилання, зберігаючи правила пересилання у своїх власних таблицях потоків. Площина керування, що працює на іншому пристрої, має завдання керувати таблицею потоків у кожному пристрої. Це дає можливість динамічно конфігурувати пристрої та надає глобальне уявлення про мережу. Рисунок 2.1 показує розгортання площин управління та даних у традиційних мережах та SDN.

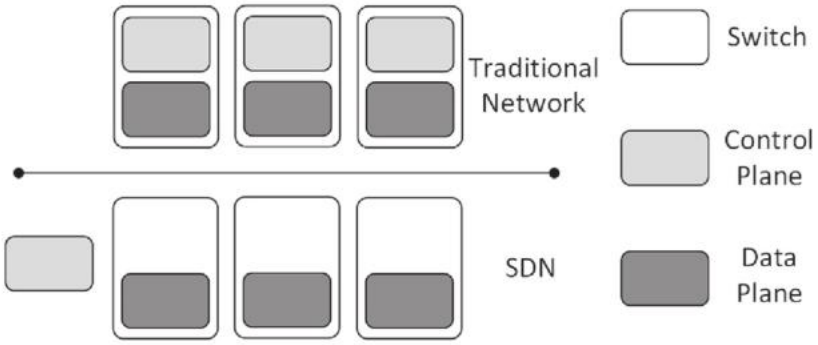

Рисунок 2.1 – Ілюстрація високого рівня розгортання площини керування та даних

SDN забезпечує наступні переваги в порівнянні з традиційними мережами:

- глобальний контроль – контролер має глобальне уявлення про топологію мережі, стан мережі та вимоги до програми;

- програмованість і гнучкість – площину даних можна динамічно програмувати для покращення розподілу мережевих ресурсів;

- відкритість – зв’язок між контролером і пристроями пересилання не залежить від постачальників пристроїв.

Для реалізації SDN необхідно надати протокол, який забезпечує безпечний зв’язок між площинами керування та даних. Цій вимозі відповідає протокол OpenFlow (OF) [18]. OF також надає необхідні API, для того щоб дозволити контролеру змінювати конфігурації площини даних. Специфікація OF визначає серію лічильників, які повинен забезпечити пристрій OF. Ці лічильники відображають трафік в пристрої. Архітектура OF складається з комутаційного обладнання, яким керує контролер OF (рисунок 2.2). OF визначається за допомогою Open Networking Foundation (ONF) у «Специфікації комутатора OpenFlow» [18]. У кількох дослідженнях розглядалося використання протоколу в якості базового рівня [19].

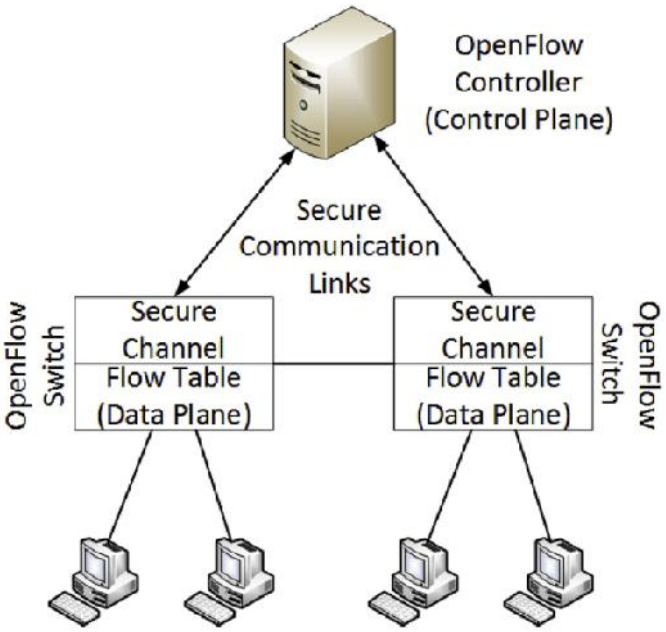

Рисунок 2.2 – Базова архітектура OpenFlow

2.2 Інженерія трафіка

Traffic Engineering (TE) відіграє важливу роль в оптимізації продуктивності мережі шляхом аналізу трафіку в реальному часі, прогнозування трафіку та розробки механізмів маршрутизації для покращення використання мережевих ресурсів [20].

Інжинірінг трафіка SDN може використовувати характеристики SDN для своєї реалізації. Вказані характеристики включають такі, що зображені на рисунку 2.3: управління потоком; стійкість до відмов; оновлення топології; аналіз і характеристика трафіку [21]. Керування потоками вирішує проблеми, пов’язані з накладними витратами трафіку в площинах керування та даних. Відмовостійкість має справу з можливістю швидкого відновлення у разі збоїв у мережевих компонентах (контролерах, комутаторах або каналах). Компонент оновлення топології керує запланованими змінами, такими як оновлення правил мережевої політики. Аналіз і характеристика трафіку відіграють особливо важливу роль у виявленні збоїв у мережі або каналі та прогнозуванні перевантаження каналів або вузьких місць.

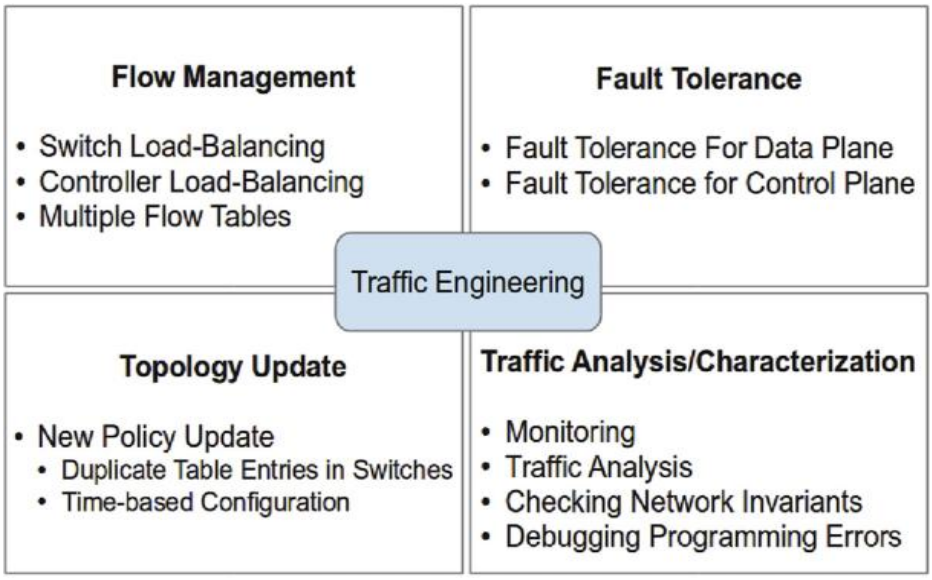

Рисунок 2.3 – Інженерія трафіка

Внесок даної роботи полягає в локалізації діяльності з аналізу трафіку та його характеристик. Тому основна увага приділяється моніторингу пропускної спроможності мережевих компонентів (комутаторів і каналів).

2.3 Проблеми та можливості

Надання статистики в режимі реального часу відіграє важливу роль у моніторингу мережі для раннього виявлення перевантажень каналів і маршрутів, що дає змогу адміністраторам мережі вживати активних дій для усунення перевантажень. Розробка та впровадження рішення для моніторингу трафіку в реальному часі стикаються з кількома проблемами через необхідність обробки великих обсягів даних. Це твердження можна підтвердити, переглянувши серію попередніх рішень для моніторингу, які зазвичай вибирають конкретне посилання, комутатор і маршрут для моніторингу.

Загальною проблемою, з якою стикалися попередні рішення для моніторингу трафіку, було агрегування зібраної статистики. Для вимірювання в реальному часі, залежно від обсягу зібраних даних, обчислення агрегованої статистики для з’єднань, комутаторів і маршрутів може зайняти багато часу, що робить рішення неспроможними задовольнити обмеження реального часу. Процес агрегації починається зі зберігання зібраних даних, які надходять у потоковому режимі, поки попередні дані все ще обробляються. Збережені дані потім групуються за заздалегідь визначеним ключем, оскільки вони надходять у випадковому порядку. Бажані результати обчислюються відповідно до згрупованих ключів. Кожен із попередніх кроків має бути вирішений будь-яким рішенням, і в більшості випадків ці рішення залежать від програми.

Методи великих даних забезпечують надійні та стабільні інструменти для вирішення проблеми агрегації. Ці інструменти забезпечують стандартизований, але гнучкий підхід, який можна використовувати в

широкому діапазоні застосунків, які мають працювати з потоковими даними та обробкою в реальному часі. Ці інструменти надають широкий спектр API для роботи з різними вимогами та керування горизонтальною масштабованістю інфраструктури. Використовуючи методи Big Data, можна безперешкодно обробляти великі обсяги даних, а статистичні дані можна надавати майже в реальному часі.

3 МЕТОД МОНІТОРИНГУ ТРАФІКУ У МЕРЕЖАХ SDN З ВИКОРИСТАННЯМ ВЕЛИКИХ ДАНИХ

У цьому розділі представлено запропонований метод моніторингу з використанням методології великих даних.

3.1 Концептуальна модель моніторингу

Комутатор OF підтримує кілька лічильників, кожен з яких відображає свій власний трафік, і які оновлюються при кожному новому пакеті. Здійснюється періодичне опитування контролера SDN з метою отримання значень лічильників для ресурсів, які контролюються в комутаторі. Лічильники можуть вестися для кількох ресурсів [18], але в рамках цієї кваліфікаційної роботи відстежуються такі ресурси:

- порти комутатора,
- таблиці потоків,
- записи потоків.

Базові значення, що надаються лічильниками в комутаторі OF, можна застосовувати для забезпечення використання пропускної здатності мережі на рівнях портів і потоків.

Метод моніторингу включає три основних етапи, як показано на рисунку 3.1.

Етап 1 – збір даних. Завантажується інформація про конфігурацію мережі і дані про трафік з площин управління та даних за допомогою контролера SDN.

Етап 2 – агрегація даних. Обробляються отримані дані, обчислюючи параметри мережі [22] в оперативному режимі (наближеному до режиму реального часу).

Етап 3 – зберігання даних. Зберігає та надає конфігураційні дані і

параметри мережі для систем аналізу трафіку.

Ці три етапи докладніше описані в наступних підрозділах.

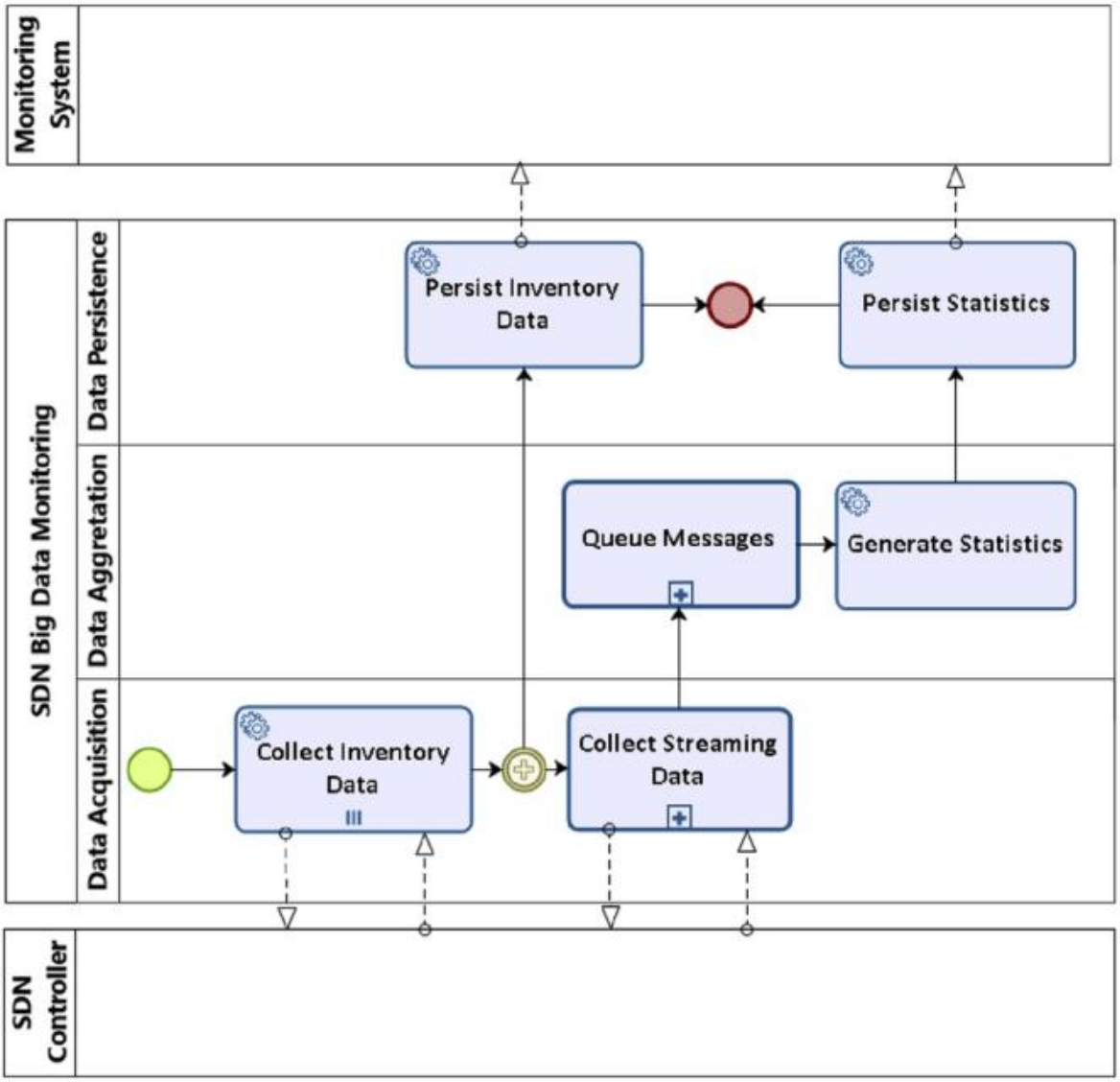

Рисунок 3.1 – Концептуальна модель моніторингу

3.2 Збір даних

Ця дія має за мету збирання всіх даних, необхідних для реалізації методу моніторингу. Зібрані дані можна класифікувати як інвентаризаційні (конфігураційні) та потокові. Дані інвентаризації заповнюють сховище, тоді як потокові дані збираються та негайно надсилаються для подальшої

обробки. Дані інвентаризації відображають інформацію про мережеві ресурси і можуть використовуватися для різних типів функціональних можливостей залежно від реалізації. Наприклад, для того щоб обчислити відсоток власної доступної пропускної здатності, яку споживає порт, інформація про доступну пропускну здатність зберігається в даних інвентаризації. Рисунок 3.2 показує модель «сутність-відношення» (ER) для сховища даних інвентаризації. Дані інвентаризації зберігаються у сутностях Host, Interface, Switch Port, Switch, FlowTable, Flow. Дані інвентаризації можна визначити як такі, що менш схильні до змін з часом, оскільки вони пов’язані з основною ідентифікацією та конфігурацією мережевого обладнання (хостів і комутаторів). Потокові дані пов’язані з лічильниками, що надаються пристроями OF. Відповідно до специфікації OF, ці лічильники оновлюються при кожному пакеті, який перетинає комутатор, і, отже, швидкість їх зміни вища, ніж у даних інвентаризації. Якщо ці лічильники використовуються для моніторингу або будь-якого іншого типу аналізу, їх значення необхідно періодично збирати. Ця вимога передбачає, що ці дані не повинні зберігатися в будь-якому сховищі, а після збору їх слід негайно пересилати для обробки.

Типи сутностей, зображених на рисунку 3.2:

- сутність Host («ведучий») – визначає всі комп’ютери, підключені до мережі;
- сутність Interface («інтерфейс») – описує всі мережеві інтерфейси, доступні в мережі;
- сутність Switch Port («порт комутатора») – описує всі порти, доступні в комутаторі;
- сутність Switch («комутатор») – представляє всі комутатори в мережі;
- сутність FlowTable («таблиця потоків») – представляє всі таблиці потоків, доступні в комутаторі;
- сутність Flow («потік») – описує всі потоки, створені в комутаторі.

Модель, представлена на рисунку 3.2, визначає такі відносини між сутностями:

- відносина Has («Має»). Сутність Host може встановити та налаштувати декілька мережевих інтерфейсів Interface для спілкування через мережу. Interface підключено лише до одного хосту. Варто зазначити, що, незважаючи на те, що кратність цього відношення визначається як 1xN, фактична кількість інтерфейсів, які може встановити хост, обмежена апаратурою;

- відносина Connects to («Підключено до»). Interface підключено лише до одного порту (Port), а порт можна підключити лише до одного Interface;

- відносина Links to («Зв’язано з»). Це рекурсивне співвідношення означає, що Port можна підключити тільки до одного іншого порту;

- відносина Provides («Забезпечує»). Сутність Switch може надати один або кілька Switch Port до мережі, а Switch Port може бути надано лише одним Switch;

- відносина Maintains («Підтримує»). Сутність Switch може підтримувати одну або кілька Flow Table, а кожна Flow Table підтримується лише одним Switch;

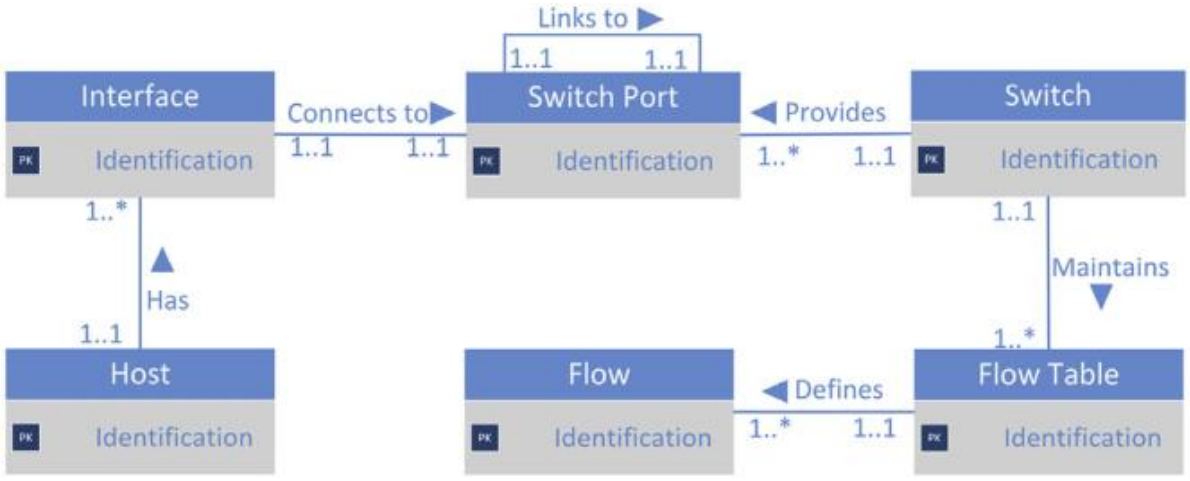

Рисунок 3.2 – ER-модель контрольованих даних

- відносина Defines («Визначає»). Сутність Flow Table визначає один або кілька Flow, а Flow визначається лише однією Flow Table.

Атрибути, які описують входження кожної сутності у Host і Interface, не визначені специфікацією OF. Ці атрибути залежать від контролера SDN і, отже, не описані в цьому розділі. Єдиним атрибутом, визначеним для цих сутностей, є identification, який однозначно ідентифікує одне входження сутності. Цей атрибут встановлюється як первинний ключ в обох сутностях Host і Interface. Таблиця 3.1 надає список атрибутів, визначених для сутності Switch, згідно специфікації OF.

Таблиця 3.1 – Атрибути сутності Switch

Атрибут	Опис
Identification	Унікальна ідентифікація комутатора в мережі
MAC address	MAC-адреса комутатора
Buffers	Максимальна кількість пакетів, що буферизуються за один раз
Tables	Кількість таблиць потоку, які підтримуються комутатором
Capabilities [1..9]	Можливості, які підтримує комутатор (наприклад, надати статистику потоку, надати статистику порту)

Список атрибутів, визначених для сутності Switch Port у категорії інвентаризації, наведено у таблиці 3.2.

Основними атрибутами для Flow Table і Flow є їхні ідентифікатори. Можна додати будь-які інші атрибути, підтримувані контролером SDN.

3.2.1 Збір інвентаризаційних даних

Метою цього завдання є збір даних, що відображають конфігурацію мережі. У мережі OF інвентаризація складається з хостів і комутаторів. Це завдання також збирає інформацію про взаємозв’язки між хостами та

комутаторами. Ці взаємозв’язки представлені відносинами Connects to і Links to (рисунок 3.2).

Таблиця 3.2 – Атрибути сутності Switch Port

Атрибут	Опис
Identification	Унікальна ідентифікація порту в комутаторі
Hardware Address	MAC-адреса порту
Name	Ім’я, зрозуміле людині
Configuration	Адміністративні налаштування порту
State	Внутрішній стан порту
Current Feature	Поточна швидкість порту та дуплексність
Supported Features	Підтримувана швидкість порту та дуплексність
Peer Features	Швидкість і дуплексність, проголошені одноранговим пристроєм
Advertised Features	Швидкість і дуплексність, проголошені портом
Current Speed	Поточна швидкість передачі даних в кбіт/с
Maximum Speed	Максимальна швидкість передачі даних в кбіт/с

Схема, представлена на рисунку 3.2, показує відносину Provides між сутностями Switch і Switch Port. Ця відносина передбачає залежність сутності Switch Port від сутності Switch. Те ж саме стосується і відносини Maintains між сутностями Switch і Flow Table. Відносина Defines між Flow Table і Flow передбачає залежність сутності Flow від сутності Flow Table. На основі цих зв’язків, коли при виконанні завдання збираються дані інвентаризації комутаторів, також збираються дані інвентаризації портів, таблиці потоків та ідентифікатори потоків. Псевдокод для цього завдання показано у лістингу 3.1.

Це завдання передбачає спостерігання за контролером SDN з метою оновлення інвентаризації мережі (рядки 3-5 лістинга 3.1). Ця періодична операція оновлюватиме репозиторій відповідно до поточної конфігурації

мережі. Інтервал часу між двома послідовними оновленнями задається змінною n (рядок 7 лістинга 3.1).

Лістинг 3.1 – Алгоритм збору даних

```
1 Procedure CollectData()
2   while TRUE do
3     CollectHostInventory()
4     CollectSwitchInventory()
5     CollectLinkInventory()
6     update Host, Switch, Switch Port, Flow
        Table, Flow repositories
7     sleep(n units of time)
8   end
```

3.2.2 Збір потокових даних

Завдання збору потокових даних, структура якої показано на рисунку 3.3, використовує північні API контролера для збору даних про лічильники ресурсів, що відстежуються в кожному комутаторі. Ця діяльність визначає шість завдань: а саме: старт збору лічильників, збір лічильників портів, збір лічильники таблиць потоків, збір лічильників потоків, форматування даних, і відправлення даних, які описані далі.

3.2.3 Старт збору лічильників

Це завдання періодично запитує інвентаризацію з метою розпочати збирання лічильників для портів і таблиць потоків. Для кожного порту та таблиці потоків запускається завдання збору. Кількість колекторів лічильників визначається:

$$C = \sum_{S=1}^{n} (p + ft), \tag{4.1}$$

де C – кількість колекторів; n – кількість перемикачів; p – кількість портів комутатора, ft – кількість таблиць потоків у комутаторі.

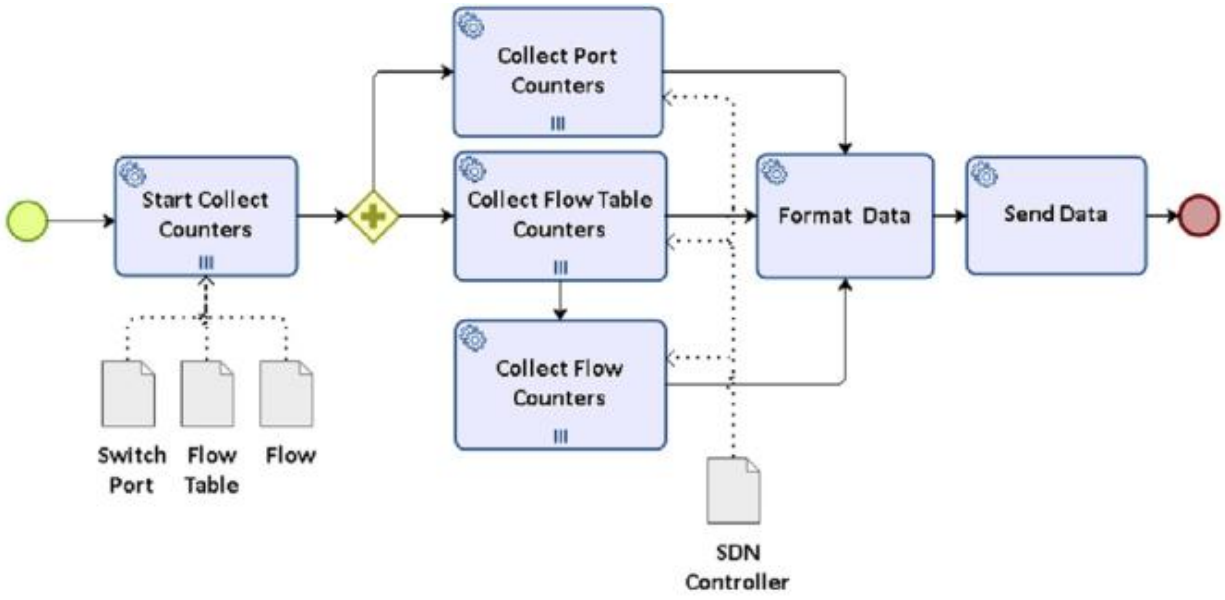

Рисунок 3.3 – Збір потокових даних

Псевдокод для цього завдання показано в лістингу 3.2. Оскільки це завдання безперервно зчитує сховище інвентаризації, очікується, що між двома чи більше зчитуваннями один і той же порт і таблиця потоків будуть у сховищі, і що для збору лічильників потрібно ініціювати лише одне завдання. Щоб уникнути кількох завдань, які збирають лічильники для одного ресурсу, рядки 5 і 11 містять оператор if, щоб забезпечити це обмеження. Наприклад, якщо порт eth0 присутній у двох або більше оновленнях репозиторію, то лише одне завдання буде збирати його лічильники. Те ж саме стосується таблиць потоку.

3.2.1 Збір лічильників портів

Це завдання реалізує збір даних лічильників для портів комутатора. Зібрані атрибути описані в таблиці 3.3. Псевдокод для цього завдання

надається в лістингу 3.3.

Лістинг 3.2 – Алгоритм запуску збирання лічильників

```
1  Procedure StartCollectCounters()
2      while TRUE do
3          read Switch Port repository
4          for each switch port do
5              if collect counters not started then
6                  start collect port counters task
7              end
8          end
9          read Flow Table repository
10         for each flow table do
11             if collect counters not started then
12                 start collect flow table counters
                       task
13             end
14         end
15         sleep (n units of time)
16     end
```

Лістинг 3.3 – Алгоритм збирання лічильників портів

```
1  Procedure CollectPortCounters()
2      while TRUE do
3          CollectPortCounters(switch,port)
4          send counters to Format Data
5          sleep(n units of time)
6      end
```

Функція CollectPortCounters(switch, port) (рядок 3) перевіряє контролер SDN кожні n (рядок 5) одиниць часу, щоб отримати значення лічильника для порту. Параметри комутатора (switch) та порту (port) визначають конкретний порт для контролера SDN.

Таблиця 3.3 – Атрибути для лічильників портів

Атрибут	Опис
Timestamp	Відмітка часу, коли збирання даних завершено
Packets Received	Кількість отриманих пакетів
Packets Transmitted	Кількість переданих пакетів
Bytes Received	Кількість отриманих байтів
Bytes Transmitted	Кількість переданих байтів
Collision Count	Кількість зіткнень
Over RunError Received	Кількість пакетів із перевищенням RX
Drops Transmitted	Кількість пакетів, відкинутих TX
Drops Received	Кількість пакетів, відкинутих RX
Frame Error Received	Кількість помилок вирівнювання кадру
CRC Error Received	Кількість помилок перевірки циклічної надмірності
Seconds	Інтервал часу у секундах, через який порт було ініціалізовано
Nanoseconds	Кількість наносекунд в секунді

3.2.2 Збір лічильників таблиць потоків

Це завдання визначає збирання даних лічильників таблиць потоків. Зібрані атрибути описані в таблиці 3.4.

Таблиця 3.4 – Атрибути для лічильників таблиць потоків

Атрибут	Опис
Timestamp	Відмітка часу, коли збирання даних завершено
Active entries	Кількість активних потоків
Packet Lookups	Кількість переглянутих пакетів у таблиці
Packet Matches	Кількість пакетів, які збігаються в таблиці

Оскільки існує залежність сутності Flow від сутності Flow Table (відношення визначається на рисунку 3.2), це завдання запускає колектор лічильника для кожного потоку. Кількість колекторів лічильника потоку визначається рівнянням:

$$Cf = \sum_{S=1}^{n} \sum_{ft=1}^{m} f, \tag{4.2}$$

де Cf – кількість колекторів лічильників потоків; n – кількість перемикачів; m – кількість таблиць потоків у комутаторі; f – кількість потоків у таблиці потоків.

Лістинг 3.4 надає псевдокод для цього завдання. Функція CollectFlowTableCounters(switch, flow table) (рядок 3) перевіряє контролер SDN кожні n (рядок 10) одиниць часу для того, щоб отримати значення лічильників для таблиці потоків. Параметри перемикача (switch) та таблиці потоків (table flow) визначають конкретну таблицю потоків для контролера SDN.

Лістинг 3.4 – Алгоритм збору лічильників таблиці потоків

```
1  Procedure CollectFlowTableCounters()
2      while TRUE do
3          CollectFlowTableCounters(switch,flow
             table)
4          send counters to Format Data
5          for each flow in the flow repository do
6              if collect counters not started then
7                  start collect flow counters task
8              end
9          end
10         sleep(n units of time)
11     end
```

3.2.3 Збір лічильників потоків

Це завдання визначає збір даних лічильників потоків. Зібрані атрибути описані в таблиці 3.5. Псевдокод для цього завдання показано в лістингу 3.5.

Таблиця 3.5 – Атрибути для лічильників потоків

Атрибут	Опис
Timestamp	Відмітка часу, коли збирання даних завершено
Received Packets	Кількість отриманих пакетів
Received Bytes	Кількість отриманих байтів
Seconds	Кількість секунд, протягом яких було встановлено потік
Nanoseconds	Кількість наносекунд в секунді

Лістинг 3.5 – Алгоритм збору лічильників потоків

```
1 Procedure CollectFlowCounters()
2     while TRUE do
3         CollectFlowCounters(switch,flow
            table,flow)
4         send counters to Format Data
5         sleep(n units of time)
6     end
```

Функція CollectFlowCounters (switch, flow table, flow) (рядок 3) перевіряє контролер SDN кожні n (рядок 5) одиниць часу з метою отримання значень лічильника для потоку. Параметри комутатора (switch), таблиця потоків (flow table) і потоку (flow) визначають конкретний потік для контролера SDN.

3.2.4 Форматування даних

Усі зібрані дані лічильника використовуються для подальшої обробки. Передбачено, що формат, у якому збираються дані, є відмінним від того, у якому вони будуть оброблятися. Також очікується, що рівень, на якому вони зібрані, не є необхідним для обробки. Це завдання визначає форматування даних перед відправленням на обробку. Формат залежить від реалізації. Псевдокод для цього завдання показано в лістингу 3.6.

Лістинг 3.6 – Алгоритм форматування даних

```
1 Procedure FormatData()
2     while TRUE do
3         receive counters data
4         format data
5         forward data to the send data task
6         sleep(n units of time)
7     end
```

3.2.5 Відправлення даних

Це завдання визначає процедуру отримання відформатованих даних від завдання форматування даних, надає позначення для них (порт комутатора, таблиця потоків або лічильники потоків) і пересилає до завдання формування черги повідомлень.

Як показано на рисунку 3.4, комутатор має порти та таблиці потоків, а кожна таблиця потоків має потоки. Для кожного порту, таблиці потоків і потоку завдання збирання лічильників (Collect Port Counters, Collect Flow Table Counters, Collect Flow Counters) перевіряє контролер SDN для отримання даних. Ця процедура представлена в рядку 3 лістингів 3.3, 3.4 і 3.5. Така ж процедура присутня в рядках 3, 4 і 5 лістингу 3.1. Процедура в задачі збирання лічильників є більш інтенсивною в обчислювальному

відношенні з двох основних причин: кількість елементів, що відстежуються, значно більша, а отже, кількість запитів до контролера SDN також значно більше. Швидкість запитів може бути вищою або нижчою в залежності від інтервалу часу між запитами.

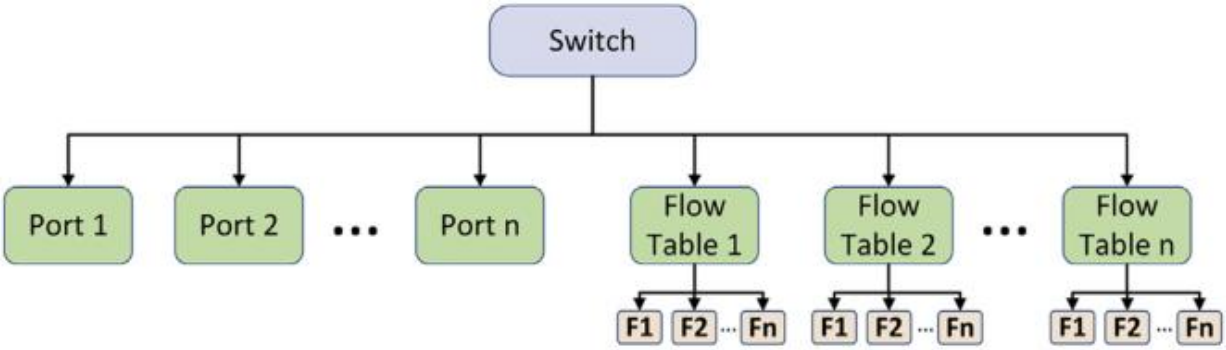

Рисунок 3.4 – Ієрархія перемикачів

Інтервал часу між запитами визначається оператором sleep, присутнім у всіх алгоритмах, згаданих у попередньому абзаці. Значення n в лістингу 3.1 має бути більше, ніж у колекторах лічильників. Вище було зазначено, що дані про конфігурацію, як правило, відносно постійні з часом. Кількість хостів та їх конфігурація змінюються не дуже часто, і те саме стосується комутаторів та мережевих підключень. З іншого боку, дані лічильника постійно змінюються через кількість пакетів, які перетинають комутатор. Кількість запитів, надісланих до контролера SDN за одну ітерацію циклу while у лістингу 3.1 становить три. Це число передбачає, що контролер SDN надає один запит для кожного типу ресурсу (Host, Switch або Link), але це число залежить від північного API, наданого контролером. Наприклад, один контролер може повернути всі дані лише одним запитом. Коли контролер підтримує декілька варіантів запиту даних, вибір того, який варіант використовувати, може бути заснований на затримці запиту. Запит, який повертає всі дані, може мати більшу затримку, тоді як запит для кожного типу ресурсу може мати меншу затримку, але це вимагає більше запитів.

Такий компроміс необхідно вирішити на основі вибраного контролера SDN.

3.3 Агрегація даних

Ця дія має за мету отримувати дані, зібрані дією збору даних, обробляти їх, генеруючи потрібну статистику, та надсилати результат для збереження. Виконання цієї дії визначає підпроцеси формування черги повідомлень і формування статистики, які описані далі.

3.3.1 Формування черги повідомлень

Мета цього підпроцесу – гарантувати збереження повідомлень, отриманих від дії збору даних, та забезпечити їх доступність для подальшої обробки за запитом. Повідомлення зберігаються та пересилаються в тому ж порядку, в якому вони надходять. Завдання, показані на рисунку 3.5 реалізуються з урахуванням цієї вимоги.

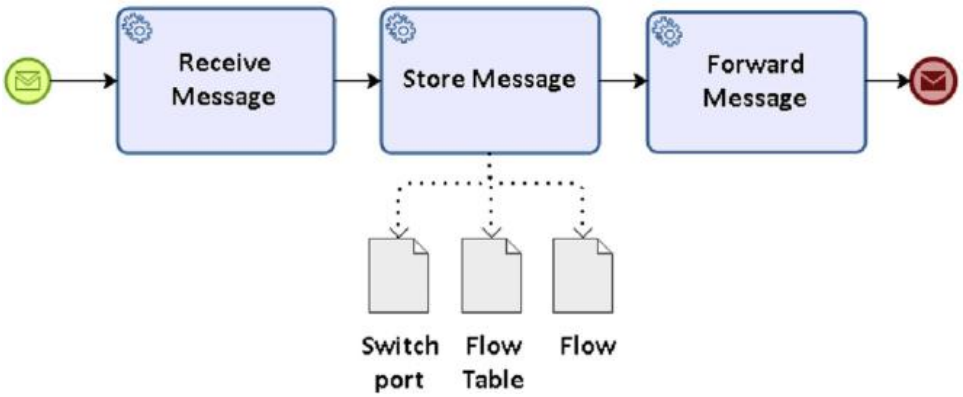

Рисунок 3.5 – Формування черги повідомлень

Завдання Receive message отримує повідомлення, що надходять від дії збору даних. Повідомлення відносяться до порту комутатора, таблиці потоків і лічильників потоків і повинні бути ідентифіковані як такі та переслані для зберігання.

Завдання Store message зберігає всі повідомлення відповідно до їх тегу (порт, таблиця потоку або потік) і надає їх відповідно до запиту, гарантуючи, що повідомлення надаються відповідно до порядку їх надходження, і жодне повідомлення не пропускається.

Метою завдання Forward message є отримання та пересилання збережених повідомлень, коли вони запитуються для подальшої обробки.

3.3.2 Формування статистики

Статистичні дані періодично формуються на основі доступних даних лічильника, збережених попереднім підпроцесом. Завдання формування статистики запитує ці дані (повідомлення), обробляє їх, обчислюючи контрольовані параметри мережі, і надсилає результати в сховище. Псевдокод для цього завдання представлено в лістингу 3.7.

Лістинг 3.7 – Алгоритм формування статистики

```
1 Procedure GenerateStatistics()
2     while TRUE do
3         request data
4         ComputeStatistics()
5         send statistics
6         sleep(n units of time)
7     end
```

Повідомлення обробляються в наборах повідомлень. Кожен набір повідомлень складається з усіх повідомлень, які надійшли до підпроцесу формування черги повідомлень під час одного статистичного обчислення. Рисунок 3.6 показує процедуру GenerateStatistics. Під час обробки пакету 1 (Batch 1) повідомлення, які формують пакет 2 (Batch 2), надходять до підпроцесу формування черги повідомлень. Наприкінці обробки пакету 1 обчислена статистика надсилається на зберігання. Під час обробки пакету 2

повідомлення, які складають пакет 3 (Batch 3), все ще надходять. Це безперервний процес. Функція ComputeStatistics обчислює статистичні дані та агрегує їх відповідно до своєї мети, якою може бути, наприклад, агрегація за комутатором, портом і таблицею потоків. Зведений елемент і параметри мережі залежать від програми.

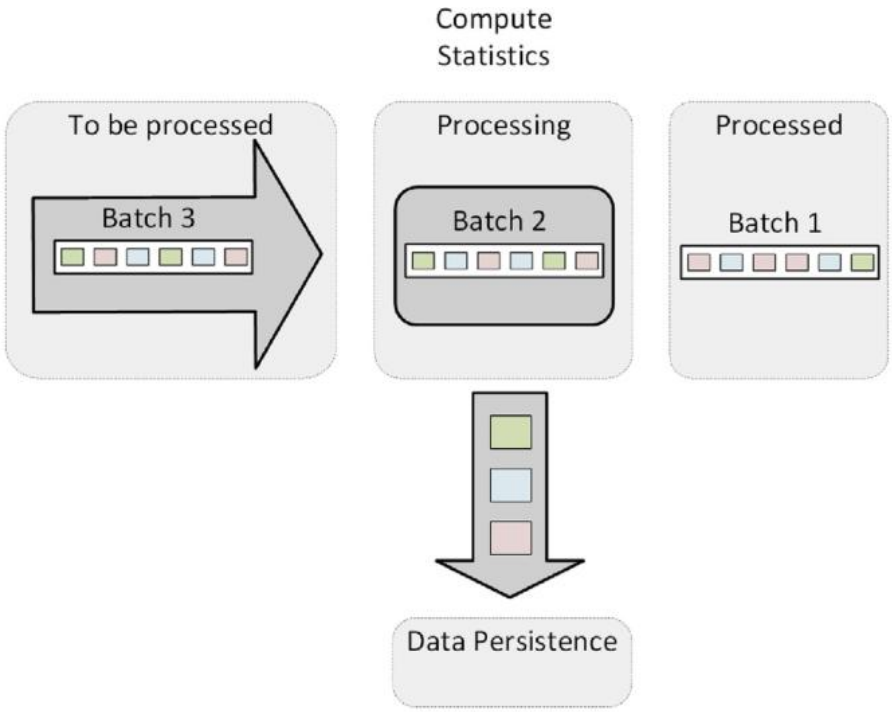

Рисунок 3.6 – Процедура генерації статистичних даних

3.4 Зберігання даних

Зібрані інвентаризаційні дані і згенеровані параметри мережі обробляються у цій дії. Дані можуть використовуватися будь-якою системою моніторингу, аналізу трафіку або візуалізації.

Завдання Persist inventory data визначає зберігання даних для інвентаризації мережі. Ці дані можна використовувати для відтворення топології мережі шляхом визначення всіх її компонентів.

Параметри мережі, згенеровані дією агрегації даних, зберігаються завдяки завданню Persist statistics. Слід зазначити, що достатньо складно

синхронізувати момент, коли система запитує ці параметри, і момент, коли вони доступні. Щоб подолати цю ситуацію, параметри мережі зберігаються в репозиторії і доступні, коли споживаючій системі потрібні останні обчислені значення. Робота системи моніторингу в реальному часі значно збільшує швидкість оновлення цих параметрів. Будь-яка клієнтська система може запитувати збережені значення зі швидкістю, співрозмірною зі швидкістю оновлення параметрів у сховищі, з ризиком пропустити деякі оновлення.

4 ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

У цьому розділі проводиться оцінювання запропонованого методу при реалізації з використанням OpenDaylight в якості контролера SDN та надаються результати вимірювань в реальному часі.

Для розгортання програми було використано чотири машини. Вони описані в таблиці 4.1.

Таблиця 4.1 – Апаратне середовище

id	ЦП	ОЗП
Сервер 1	2 процесора Intel(R) Xeon(R) E5-2630 v2 @ 2,60 ГГц	96 ГБ
Сервер 2	2 процесора Intel(R) Xeon(R) E5-2630 v2 @ 2,60 ГГц	96 ГБ
Сервер 3	2 процесора Intel(R) Xeon(R) E5-2620 v3 @ 2,40 ГГц	96 ГБ
Сервер 4	1 процесор Intel(R) Core(TM) i7-4710MQ @ 2,50 ГГц	16 ГБ

Сервери 1, 2 і 3 підключені до системи зберігання даних HP MSA 2040 SAN 12 ТБ. Сховище було поділено на чотири логічні розділи по 2,5 ТБ кожен.

Усі сервери працювали під керуванням Linux Ubuntu Server 16.04, як і всі налаштовані контейнери. На серверах було активовано опцію Multipath, що дозволило їм мати кілька шляхів введення-виводу до сховища, що забезпечувало паралельний доступ до чотирьох налаштованих розділів.

Рисунок 4.1 представляє топологію, використану в цих експериментах. Елементи S1-S6 представляють мережні комутатори, а елементи h1-h8 – хости. Порти в комутаторах представлені eth1,eth2, тощо.

Для створення топології мережі та проведення експериментів використовувався емулятор Mininet. Відповідно до діаграми планування, показаної на рисунку 4.2, були створені потоки TCP з використанням Iperf3. На цій діаграмі показано трафік, створений між парами хостів, і відповідна

налаштована пропускна здатність. Для того, щоб змінити конфігурацію маршруту за замовчуванням, надану OpenDaylight, були додані деякі записи в таблиці потоків. Ці записи потоків визначають маршрутизацію рівня 3, як показано в таблиці 4.2.

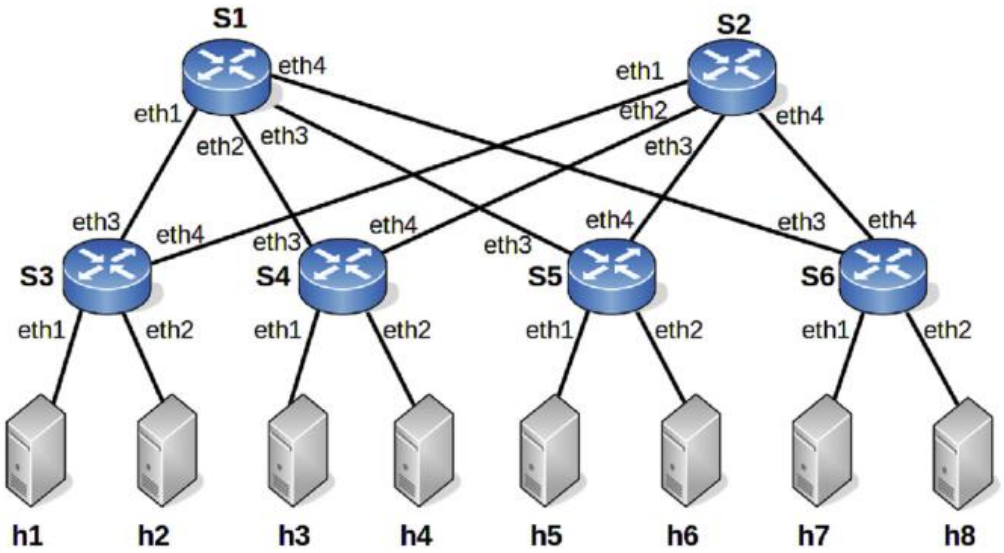

Рисунок 4.1 – Топологія мережі

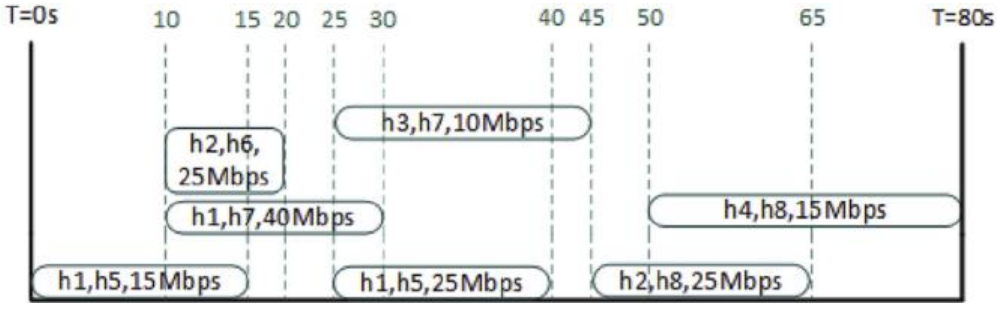

Рисунок 4.2 – Діаграма планування

Під час експериментальної реалізації, з метою збору статистики трафіку, контролер SDN опитувався кожні три секунди. Зібрані дані оброблялися потоковим застосунком Spark, і кожні три секунди вся статистика зберігалася в Elasticsearch. Незважаючи на те, що реалізація забезпечувала всю статистику в режимі реального часу, графіки

створювалися офлайн. Наведені в цьому розділі графіки показують статистику для деяких вибраних ресурсів та елементів із множини тих, що відстежуються. За винятком випадків, коли це явно зазначено, статистика, створена цією реалізацією, позначає кумулятивну пропускну здатність. Схема розкладу, показана на рисунку 4.2 відображає пропускну здатність, вибрану для кожної одиниці створеного трафіку. Значення пропускної здатності зазначено у Мбітах за секунду.

Таблиця 4.2 – Маршрути 3-го рівня

Перемикач	IP-адреса джерела	IP-адреса призначення	Дія
S1	10.0.0.1	10.0.0.5	вихід: eth3
	10.0.0.2	10.0.0.8	вихід: eth4
	10.0.0.3	10.0.0.7	вихід: eth4
S2	10.0.0.1	10.0.0.7	вихід: eth4
	10.0.0.2	10.0.0.6	вихід: eth3
	10.0.0.4	10.0.0.8	вихід: eth4
S3	10.0.0.1	10.0.0.5	вихід: eth3
	10.0.0.1	10.0.0.7	вихід: eth4
	10.0.0.2	10.0.0.6	вихід: eth4
	10.0.0.2	10.0.0.8	вихід: eth3
S4	10.0.0.3	10.0.0.7	вихід: eth3
	10.0.0.4	10.0.0.8	вихід: eth4
S5	*	10.0.0.5	вихід: eth1
	*	10.0.0.6	вихід: eth2
S6	*	10.0.0.7	вихід: eth1
	*	10.0.0.8	вихід: eth2

З метою перевірки наданої статистики, за допомогою tcpdump були захоплені пакети на портах у топології. Використовуючи tshark і сценарій python, кожні три секунди для кожного порту та комутатора обчислювалася

пропускна здатність; потім результати були нанесені на графік. Цей графік мав на меті показати, що статистика, розрахована в цьому дослідженні, дотримується тієї ж схеми, що й розрахована за допомогою tshark. У tshark розраховували статистику кожні три секунди, а значення лічильника, надані стандартом OF, включали секунди та наносекунди, що не дозволяло точного збігу значень на наносекундному рівні. Отже, не можна очікувати, що значення, обчислені за допомогою tshark, і ті, що надаються цією реалізацією, будуть однаковими.

4.1 Пропускна здатність комутаторів і портів

Рисунок 4.3 показує еволюцію з часом пропускної спроможності комутатора S6 і кожного з його портів (eth1-eth4) протягом періоду моделювання. Граф внизу показує пропускну здатність для комутатора, а інші надають пропускну здатність для кожного окремого порту.

Пропускна здатність комутатора спочатку не забезпечувалася контролером SDN і становила сукупністю значень його портів. Вираз для цієї агрегації:

$$S = \sum_{n=1}^{N} T_n, \tag{4.1}$$

де S – пропускна здатність комутатора; T_n – пропускна здатність порта n; N – кількість портів комутатора.

Відповідно до розкладу на рисунку 4.2, 20-секундний сплеск трафіку починається від h1 до h7 на 10-ій секунді. Це відображається на порту eth1, де спостерігалося збільшення пропускної здатності з моменту 10 до 30. Це збільшення тривало до моменту 30, оскільки ще один сплеск трафіку в момент 25 почався з h3 до h7. Порт eth2 відображає як трафік 20-ї секунди, розпочатий з h2 до h8, так і трафік 30-ї секунди, розпочатий з h4 до h8. Порт eth3 показує 20-секундний трафік між h3 і h7 і 20-секундний трафік від h2 до

h8. Порт eth4 показує трафік 20-ї секунди від h1 до h7 і 30-ї секунди трафік від h4 до h8.

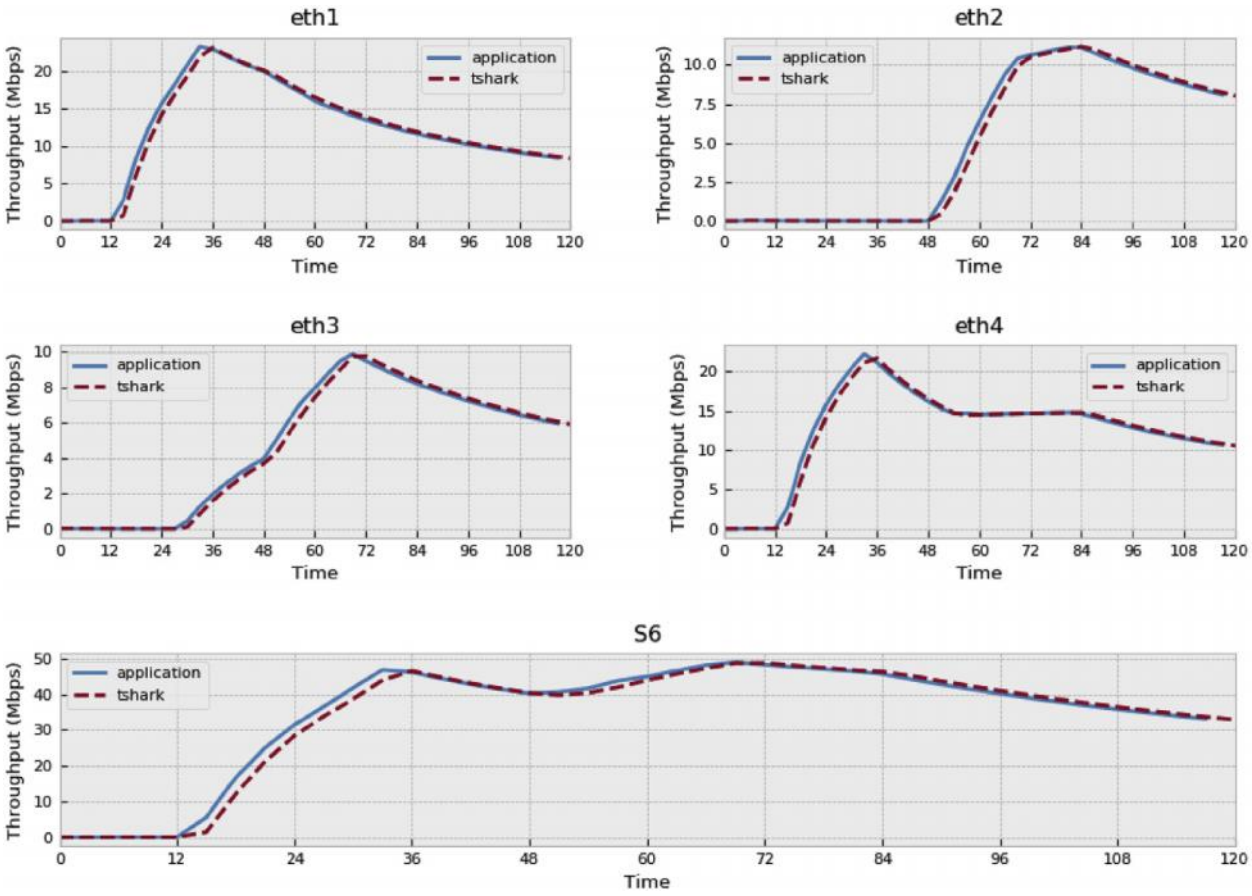

Рисунок 4.3 – Пропускна здатність комутаторів і портів

4.2 Деградація пропускної здатності комутатора

Інша отримана статистика представляє внесок (відсоток) пропускної здатності порту в пропускну здатність комутатора. Це можна побачити на рисунку 4.4. На графіку показано зміну внеску кожного порту в навантаження комутатора з часом. Для кожного порту відсоток розраховували відповідно до виразу:

$$P = \frac{T_p}{T_s} \times 100, \qquad (4.2)$$

де T_p – пропускна здатність порту; T_s – пропускна здатність комутатора.

Рисунок 4.3 показує, що трафік udp починається через 10 секунд у S6. Відсоток, представлений на рисунку 4.4, від 0 до 10-ї секунди пов’язаний з пакетами з інших протоколів, таких як MDNS, LLDP, ARP та ICMP, оскільки моделювання включало перший ping для всіх машин для перевірки підключення. Рисунок 4.4 показує, що з моменту 10-25 порти eth1 і eth4 генерували весь трафік в комутаторі, що підтверджується рисунком 4.3. Крім того, порт eth2 почав свій внесок у навантаження комутатора через 50 секунд.

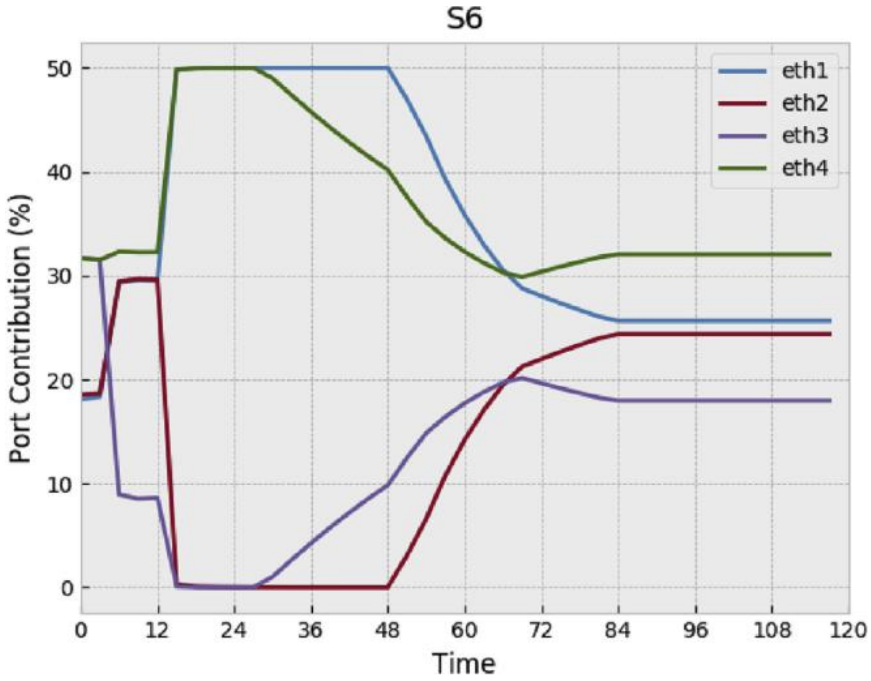

Рисунок 4.4 – Внесок порту в пропускну здатність комутатора

4.3 Пропускна здатність комутаторів

Пропускна здатність кожного комутатора в топології була розрахована на основі (4.1). Ця процедура дозволила відстежувати тенденцію пропускної здатності всіх комутаторів одночасно. Топологія мережі містила шість

комутаторів (S1-S6). Рисунок 4.5 показує виміряну пропускну здатність і перевірку tshark для комутаторів S1, S3 і S4. У перші 15 с комутатори S1 і S3 мають збільшення пропускної здатності. S1 знаходиться на маршруті руху від h1 до h5 протягом 15 с. Через 15 секунд пропускна здатність S1 зменшилася на 10 секунд, але почала зростати через 25 секунд через зростання трафіку з h1 до h5 і через 45 секунд через (збільшення) трафіку від h2 до h8 протягом 20 секунд. Пропускна спроможність S3 зростала швидше, оскільки вона здійснювалася від трафіку від h1 до h5 і від h1 до h7 за 10 секунд і знову від трафіку від h1 до h5 за 25 секунд. Трафік на S4 розпочався лише на 25-й секунді від h3 до h7 і дещо збільшився на 50-й через трафік між h4 та h8.

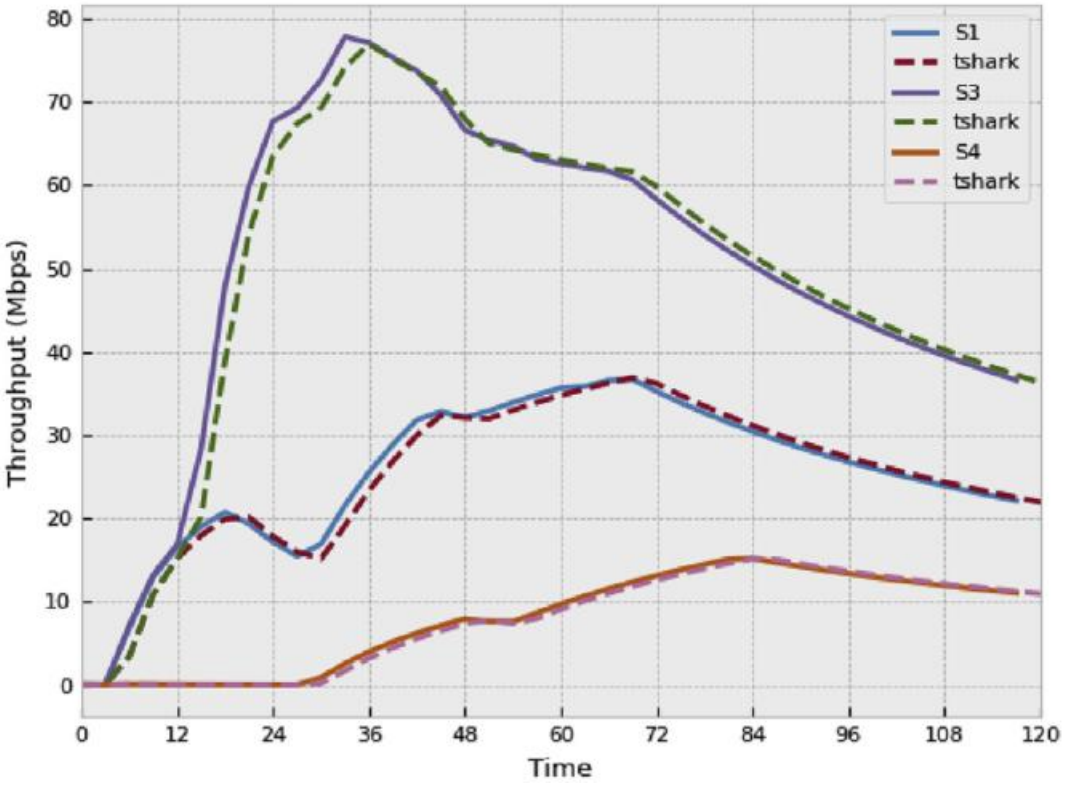

Рисунок 4.5 – Пропускна здатність комутатора

4.4 Використання потужності порту

Також вивчався відсоток власної максимальної пропускної здатності, яка використовувалася кожним портом. Ця статистика наведена на рисунку 4.6. Відсоток розраховано відповідно до рівняння:

$$P = \frac{T_p}{P_c} \times 100, \tag{4.3}$$

де T_p – пропускна здатність порту; P_c – ємність пропускної здатності комутатора.

Змінна P_c визначається Switch Port і представлена в таблиці 3.2 як Current Feature. Значення, надане OpenDaylight для середовища дослідження, становило 1 Гбіт/с. Оскільки це значення було занадто великим для експериментів, було використано значення 100 Мбіт/с.

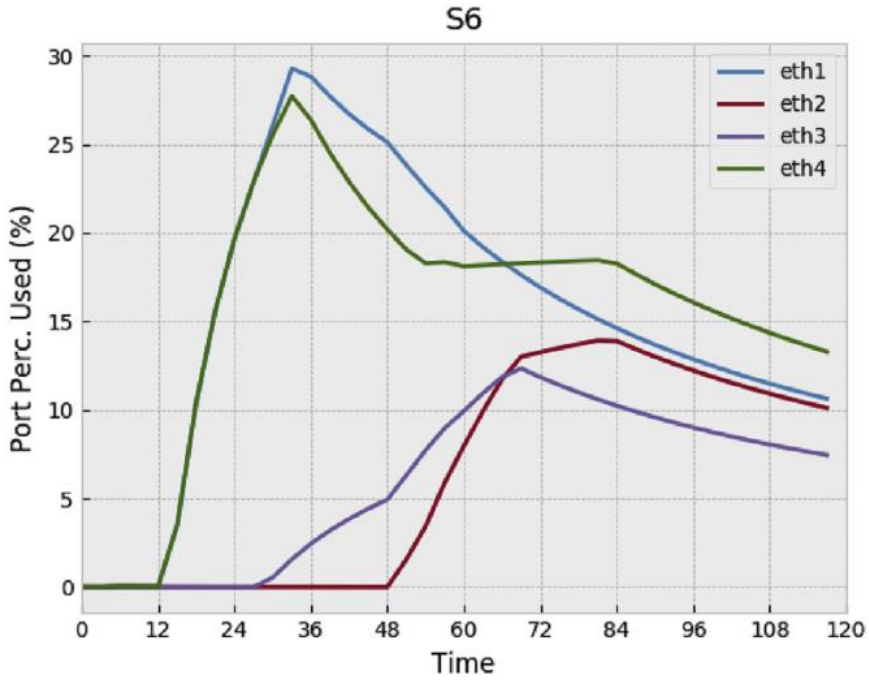

Рисунок 4.6 – Відсоток використаної ємності порту

Рисунок 4.6 показує значення P, розраховані для всіх портів (eth1-eth4) комутатора S6. Порт eth1 досяг свого піку використання між 25 і 30 секундами через трафік від h1 до h7 (10 секунд) і h3 до h7 (25 секунд). Порту eth2 знадобилося більше часу, щоб досягти свого піку, оскільки трафік, який проходив через нього, розпочинався лише з 45 с (h2-h8 і h4-h8). На відсоток використання eth3 впливав трафік від h3 до h7 і від h2 до h8. Нарешті, eth4 мав майже такий саме відсоток використання, як eth3, завдяки трафіку від h1 до h7 і h4 до h8.

4.5 Пропускна здатність з’єднання

Оскільки мережа складається з різних каналів, дана реалізація забезпечує пропускну здатність для кожного каналу в топології. Відповідний графік показано на рисунку 4.7. Зв’язки формуються між хостами і комутаторами, а також між портами комутатора. Якщо це зв’язок між хостом і комутатором, реалізація просто збирає пропускну здатність порту комутатора, оскільки контролер SDN не надає статистику для мережевого інтерфейсу, приєднаного до хоста. Якщо між портами комутатора встановлено зв’язок, реалізація збирає пропускну здатність з обох портів. Як показано у таблиці 3.3, протокол OF визначає поля Bytes Received і Bytes Transmitted для лічильників портів. Якщо реалізація збирає ці лічильники для двох портів, які одночасно (з однаковими значеннями Second і Nanosecond) утворюють посилання (таблиця 4.3), ці значення мають бути однаковими, тобто Bytes Received одного порту має бути таким же, як Bytes Transmitted свого партнера і навпаки. Отже, для реалізації потрібні лише лічильники одного з портів у посиланні. Оскільки поточна реалізація запускає різні потоки для збору лічильників для кожного порту, очікується, що показання між парами Second і Nanosecond не будуть однаковими. Для посилання, утвореного в цій ситуації, використовується така послідовність дій для зв’язків, утворених S_{a_i} і S_{b_j}.

Крок 1. Визначити S_{a_i} в якості порта і комутатора a.

Крок 2. Визначити S_{b_j} в якості порта j комутатора b.

Крок 3. Визначити BR_{a_i} та BT_{a_i} в якості отриманих параметрів Bytes Received і Bytes Transmitted, відповідно, при опитуванні S_{a_i}.

Крок 4. Визначити BR_{b_j} та BT_{b_j} в якості отриманих параметрів Bytes Received і Bytes Transmitted, відповідно, при опитуванні S_{b_j}.

Крок 5. Якщо $(BR_{a_i} + BT_{a_i}) > (BR_{b_j} + BT_{b_j})$, то використати значення S_{a_i}; інакше – використати значення S_{b_j}.

Цей підхід надає найновіші статистичні дані про пропускну здатність зв’язку.

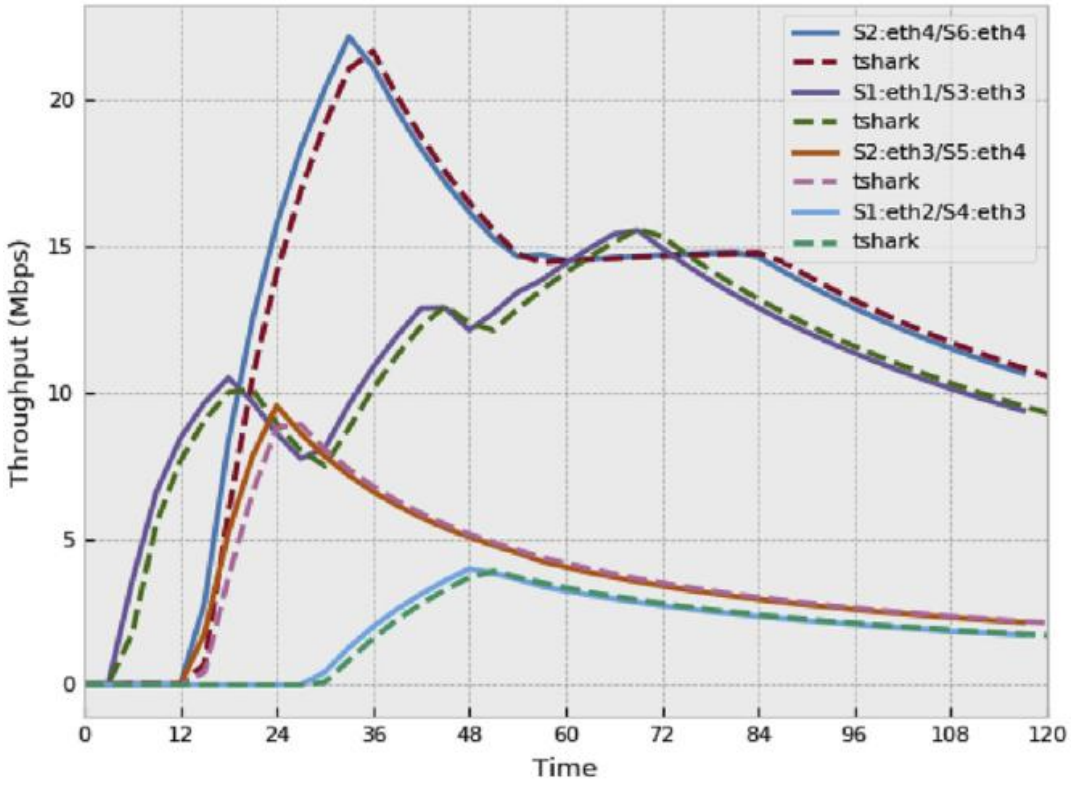

Рисунок 4.7 – Пропускна здатність зв’язку

Рисунок 4.7 показує розраховану пропускну здатність у зв’язках, утворених S2:eth4 і S6:eth4, S1:eth1 і S3:eth3, S2:eth3 і S5:eth4 і S1:eth2 і

S4:eth3. Пропускна здатність у каналі S2:eth4/S6:eth4 збільшується під час трафіку від h1 до h7, показує невелике зниження, а потім перестає зменшуватися через трафік між h4 до h8. Поведінка пропускної здатності, що спостерігається в каналі S1:eth1/S3:eth3, визначається трафіком між h1 і h5 і між h2 і h8. Трафік від h2 до h6 є єдиним екземпляром, який використовує посилання S2:eth3/S5:eth4. Посилання S1:eth2/S4:eth3 використовується трафіком від h3 до h7.

4.6 Пропускна здатність каналу між хостами

Важливою структурою в моніторингу пропускної здатності мережі є матриця трафіку (TM), яка забезпечує обсяг трафіку між парами відправлення та призначення (OD) в мережі [23]. У використаній топології (рисунок 4.1), для того, щоб знайти шляхи між будь-якими двома хостами, потрібно виявити цикли у мережевому графі. Для вирішення цієї задачі, топологія була розділена на дві частини: перша має S1 як корінь, а друга має S2 як корінь (рисунки 4.8, 4.9). За допомогою цього підходу реалізація забезпечує оцінку пропускної здатності між усіма можливими комбінаціями OD, що проходить через S1 або S2, як наведено в таблиці 4.2. Рисунок 4.10 показує виміряну пропускну здатність між пунктами відправлення і призначення.

Таблиця 4.2 – Пункти відправлення та призначення

S1	S2
h1 – h2	h1 – h2
h1 – h3	h1 – h3
h1 – h4	h1 – h4
h7 – h8	h7 – h8

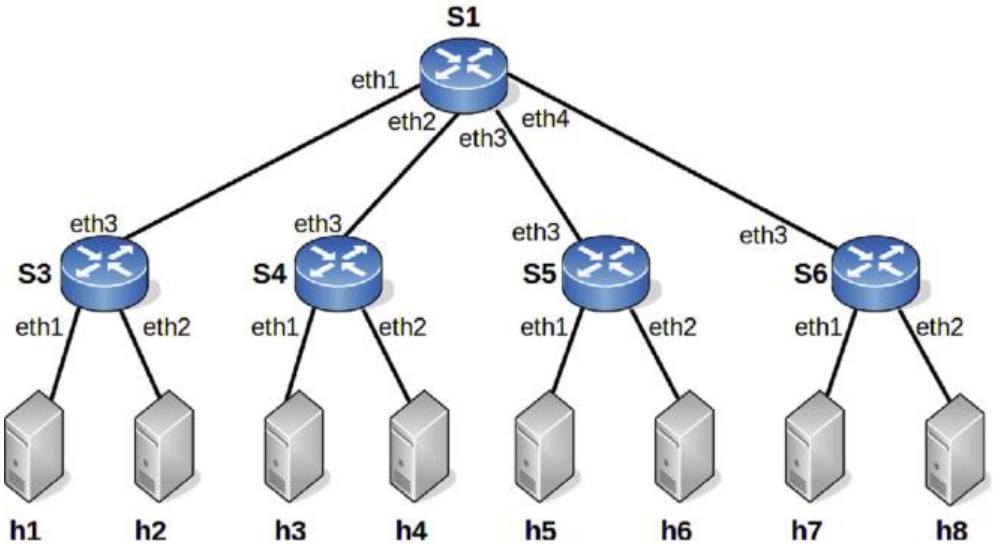

Рисунок 4.8 – Обхід топології S1

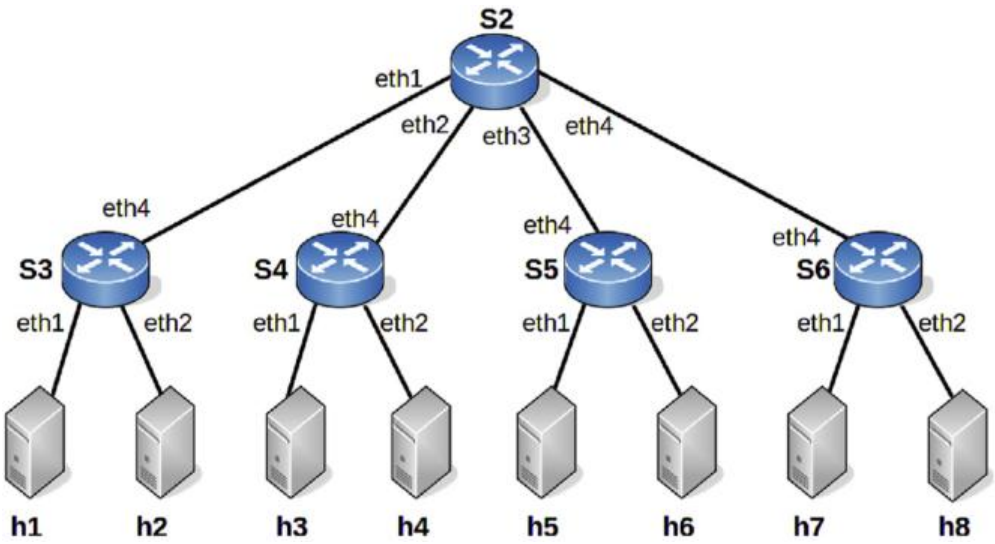

Рисунок 4.9 – Обхід топології S2

Щоб розрахувати пропускну здатність між хостами, необхідно агрегувати отриману пропускну здатність у всіх каналах, які знаходяться на шляху між хостами OD, відповідно до рівняння:

$$T_{OD} = \sum_{n=1}^{N} T_{Ln}, \tag{4.4}$$

де T_{OD} – пропускна здатність між OD; T_{Ln} – пропускна здатність у зв’язку n.

Рисунок 4.10 показує оцінку пропускної здатності між такими парами хостів: h1-h5 (root S1), h3-h8 (root S2) і h2-h4 (root S2). Таблиця 4.1 надає правила для перевірки, які порти використовуються кожною парою OD. Наприклад, наступні посилання з’єднують h3 з h8, використовуючи S2 як корінь: S4:eth1, S4:eth4/S2:eth2, s2:eth4/S6:eth4 і S6:eth2. Пара OD h1-h5 має два конкретних стани трафіку, визначені в моменти 0 і 25 с. Немає трафіку, який почався між h3-h8, а це означає, що посилання, які їх з’єднують, використовуються для оцінки пропускної здатності. У цьому випадку для оцінки пропускної спроможності використовуються такі потоки трафіку: h1-h7, h3-h7 і h4-h8. Така ж ситуація стосується і пари OD h2-h4. У цьому випадку трафік між h1-h7, h2-h6 і h4-h8 генерує пропускну здатність для цієї пари OD.

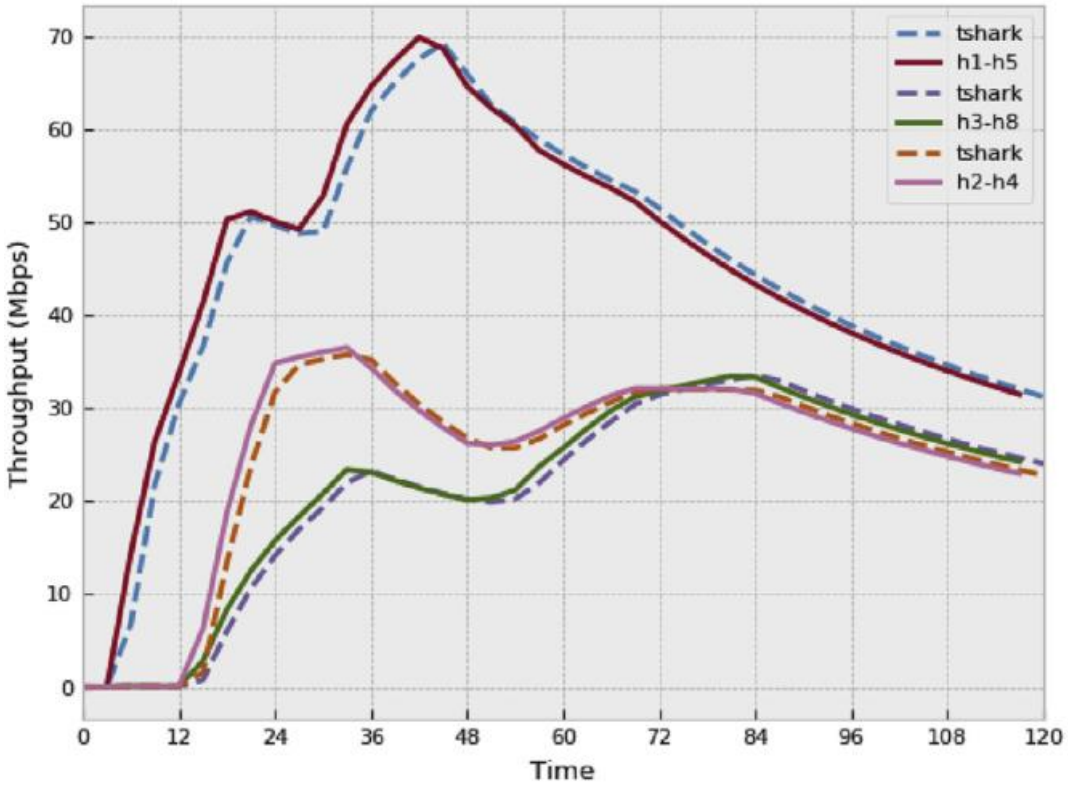

Рисунок 4.10 – Пропускна здатність між пунктами відправлення-призначення

ВИСНОВКИ

У цій роботі представлено новий підхід до вимірювання трафіку в мережах SDN. Було запропоновано детальний метод моніторингу великих даних для збору та формування статистики трафіку за допомогою значень лічильників. Для підтвердження ефективності цього методу виконана реалізація з використанням спеціальних інструментів Big Data. Також досліджується використання методів аналізу великих даних для аналізу в реальному часі великої кількості збережених історичних даних.

Результати показали, як запропонований метод використовував моніторинг TE, надаючи уявлення про те, як мережевий трафік впливає на пристрої, маршрути, що з’єднують будь-яку пару хостів OD, і зв’язки між будь-якою парою пристроїв. Крім того, оскільки запропонований метод зберігає всю згенеровану статистику в базі даних NoSQL, стає можливим використовувати аналітику великих даних для аналізу мережевого трафіку. Результати аналізу великих даних можуть допомогти при розв’язанні задач TE, таких як балансування навантаження комутатора та відмовостійкість площини даних.

Це дослідження може бути розвинуте у напрямку зменшення інтервалу між показаннями лічильника з 3 с до 1 с, зберігаючи при цьому незмінні результати. Це досить складно, оскільки запропонований метод використовує потоки для надсилання запитів до контролера SDN. Зменшуючи часовий інтервал між показаннями лічильника, збільшиться кількість запитів, що надсилаються потоками до контролера за одиницю часу.

За темою кваліфікаційної роботи опубліковано тези доповіді в рамках дев’ятої міжнародної науково-технічної конференції «Проблеми інформатизації» [24].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Y. Lee and Y. Lee. Toward Scalable Internet Traffic Measurement and Analysis with Hadoop. SIGCOMM Comput. Commun. Rev., 43(1):5-13, 2013.

2. C. Orsini, A. King, D. Giordano, V. Giotsas, and A. Dainotti. BGPStream: A Software Framework for Live and Historical BGP Data Analysis. Proc. of the IMC, pages 429-444, 2016.

3. S. Wassermann, P. Casas, T. Cuvelier, and B. Donnet. NETPerfTrace: Predicting Internet Path Dynamics and Performance with Machine Learning. Proc. of the Big-DAMA, pages 31-36, 2017.

4. A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras. A Survey of Clustering Algorithms for Big Data: Taxonomy and Empirical Analysis. IEEE Trans. Emerg. Topics Comput., 2(3):267-279, 2014.

5. C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos. Big Data Analytics: A Survey. J. Big Data, 2(1):21, 2015.

6. Y. Zhang, T. Cao, S. Li, X. Tian, L. Yuan, H. Jia, and A. V. Vasilakos. Parallel Processing Systems for Big Data: A Survey. Proc. IEEE, 104(11):2114-2136, 2016.

7. A. Callado, C. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. Fernandes, and D. Sadok. A Survey on Internet Traffic Identification. Commun. Surveys Tuts., 11(3):37-52, 2009.

8. T. T. Nguyen and G. Armitage. A Survey of Techniques for Internet Traffic Classification Using Machine Learning. Commun. Surveys Tuts., 10(4):56-76, 2008.

9. M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network Anomaly Detection: Methods, Systems and Tools. Commun. Surveys Tuts., 16(1):303-336, 2014.

10. R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto,

and A. Pras. Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. Commun. Surveys Tuts., 16(4):2037-2064, 2014.

11. S. Valenti, D. Rossi, A. Dainotti, A. Pescape, A. Finamore, and M. Mellia. Reviewing Traffic Classification. In Data Traffic Monitoring and Analysis – From Measurement, Classification, and Anomaly Detection to Quality of Experience. Springer, Heidelberg, 1 edition, 2013.

12. H. Hu, Y. Wen, T.-S. Chua, and X. Li. Toward Scalable Systems for Big Data Analytics: A Technology Tutorial. IEEE Access, 2:652-687, 2014.

13. D. Laney. 3d Data Management: Controlling Data Volume, Velocity, and Variety. Technical report, META Group, 2001.

14. J. Manyika and others. Big Data: The Next Frontier for Innovation, Competition, and Productivity, 2011.

15. F. Bajaber, R. Elshawi, O. Batarfi, A. Altalhi, A. Barnawi, and S. Sakr. Big Data 2.0 Processing Systems: Taxonomy and Open Challenges. J. Grid Comput., 14(3):379-405, 2016.

16. R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo. A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. J. Internet Serv. Appl., 9(16), 2018.

17. T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. Proc. of the SIGCOMM, pages 229240, 2005.

18. OpenFlow Switch Specification 1.5.0, 2014 [Online] Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.

19. F. Volpato, M.B. Castro, M.A.R. Dantas, 2019. OFQuality: quality of service management module for software-defined networking. Int. J. Grid Comput. Util. – IJGUC 10 (2), 187-198.

20. I.F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou. Research challenges for traffic engineering in software defined networks. IEEE Network 30 (3), 52-58, Jun.2016.

21. I.F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou. A roadmap for traffic engineering in SDN-OpenFlow networks. Comput. Network. 71, 1-30, 2014.

22. Z. Shu, J. Wan, J. Lin, S. Wang, D. Li, S. Rho, C. Yang. Traffic engineering in software-defined networking: measurement and management. IEEE Access 4, 3246-3256, 2016.

23. N.L.M. van Adrichem, C. Doerr, F.A. Kuipers. OpenNetMon: network monitoring in OpenFlow software-defined networks. In: 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-8, 2014.

24. Гнип А.К., Торба А.А. Метод моніторингу трафіку з використанням технологій Big Data для мереж мобільного звязку // Проблеми інформатизації. Тези доповідей дев’ятої міжнародної науково-технічної конференції, 18-19 листопада 2021 р., Черкаси: ЧДТУ, 2021. – С. 28.