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)

Методи маршрутизації для забезпечення якості мережного сервісу

(тема)

Виконав:

студент II курсу, групи СПм-22-6

Колісник Є.Б.
(прізвище, ініціали)

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

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

Освітня програма Системне програмування
(повна назва освітньої програми)

Керівник: проф. Можаєв О.О.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 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-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Колісніку Євгену Борисовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи маршрутизації для забезпечення якості мережного сервісу

затверджена наказом по університету від " 01 " квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

Топологія трикутник

Multipath TCP

Багатопотокова маршрутизація

4. Перелік питань, що потрібно опрацювати у роботі

Методи багатопоточної маршрутизації

Порівняльний аналіз методів багатопоточної маршрутизації

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 18 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	Отримання завдання та аналіз літератури	01.04.2024 – 06.04.2024	
2	Огляд існуючих рішень та методів	07.04.2024 – 12.04.2024	
3	Розробка моделі	13.04.2024 – 18.04.2024	
4	Вибір програмних засобів	19.04.2024 – 25.04.2024	
5	Програмна реалізація	26.04.2024 – 02.05.2024	
6	Аналіз отриманих результатів	03.05.2024 – 16.05.2024	
7	Оформлення записки	17.05.2024 – 14.06.2024	
8	Представлення роботи в ЕК	15.06.2024	

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Можаєв О.О..
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 74 с., 17 рис., 1 табл., 2 дод., 13 джерел.

МАРШРУТИЗАЦІЯ, АНАЛІЗ ЕФЕКТИВНОСТІ, ПОРІВНЯННЯ, КЛАСИФІКАТОР.

Метою кваліфікаційної роботи є дослідження методів маршрутизації для забезпечення якості трафіка.

У ході виконання кваліфікаційної роботи досліджено досліджено динамічний багатопотоковий протокол для транспортних сполучень, який дозволяє підвищити ефективність використання ресурсів мережі не менше, ніж у 95% випадків. Проведено порівняльнтй аналіз ефективності статичного та динамічного методів багатопоточної маршрутизації транспортних з'єднань у ПКМ мережах.

ABSTRACT

Master’s thesis: 74 pages, 17 figures, 1 tables, 2 appendices, 13 sources.

CLASSIFIER, CLASSIFIERROUTING, PERFORMANCE ANALYSIS, COMPARISON.

The purpose of the qualification work is to research routing methods to ensure traffic quality.

In the course of the qualification work, a dynamic multi-threaded protocol for transport connections, which allows to increase the efficiency of the use of network resources in at least 95% of cases, was investigated. A comparative analysis of the effectiveness of static and dynamic methods of multi-flow routing of transport connections in PCM networks was carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ЦОД – центр обробки даних

ПКМ – програмно-конфігурована мережа

ВСТУП

Вимоги додатків до швидкості передачі зростають постійно[1]. Так, згідно з прогнозом компанії Cisco Systems, обсяг відеотрафіку в найближчому майбутньому перевищить поточний обсяг відеотрафіку в десятки разів. Таке підвищення вимог пов'язане, наприклад, з поширенням високоякісних камер спостереження, телевізорів з роздільною здатністю 8K, а також з появою віртуальної реальності додатків на основі потокової передачі зображення.

Також слід зазначити тенденцію збільшення кількості центрів обробки даних (ЦОД). Зростання числа ЦОД пов'язано із змінами в організації додатків: відбувається перехід від клієнт-серверної архітектури, в якій ускладнюється підтримка програми через монолітну серверну частину, до мікросервісної архітектури, елементи якої можуть бути розподілені як за ресурсами ЦОДу, так і через мережу з ЦОД. Крім того, частина обчислень може виконуватися на периферії, дозволяючи знижувати мережні затримки на передачу інформації, а також стали потрібні функції архівації та дзеркалювання даних; банки та Інтернет-магазини розміщують у ЦОД сервера та системи зберігання даних клієнта. Перелічені зміни в організації додатків призводять не тільки до збільшення числа ЦОД та його клієнтів, а й до збільшення трафіку між ЦОД та клієнтами. Наведена теза підтверджується дослідженням TeleGeography, яке показало, що у 2017 році частка трафіку між ЦОД на найпопулярнішому маршруті через Атлантичний океан досягла 75% від загального трафіку, а у 2024 році вона має перевищити 93%.

Сказане пояснює актуальність завдання підвищення якості мережевого сервісу. Наведені вище приклади демонструють тренд збільшення обсягу трафіку в мережі, тому в цій роботі основна увага приділена задоволенню вимог додатків до швидкості транспортного сполучення, у зв'язку з чим, у дисертації термін якість сервісу трактується як швидкість транспортного

сполучення. Під швидкістю транспортного з'єднання (англ. throughput) розуміється обсяг переданих даних у одиницю часу, який містить як корисне навантаження, а й усі службові дані: поля заголовків пакетів, перенаправлення втрачених пакетів тощо. Під завданням забезпечення якості сервісу тут розумітимемо надання такого транспортного сполучення, швидкість якого не нижча за вимоги додатка.

Можна виділити два класи методів забезпечення якості сервісу: екстенсивний та інтенсивний. Методи балансування мережного трафіку дозволяють перерозподілити навантаження між альтернативними маршрутами в транспортній мережі, топологію якої свідомо вносять, щоб між будь-якою парою точок було кілька маршрутів на випадок непередбаченої відмови обладнання, тим самим підвищуючи стійкість до відмови мережі. Якщо уявити топологію мережі як графа, то надмірність у мережі означатиме багатореберну чи багатовершинну зв'язність графа. Методи мультиплексування припускають об'єднання кількох потоків даних в один з метою збільшення ефективності передачі. Приклад мультиплексування є підхід, званий мережевим кодуванням [2]. У мережевому кодуванні передбачається, що проміжні пристрої мережі можуть займатися не тільки завданням передачі даних, а й проводити операції над пакетами. Так, у лінійному мережевому кодуванні кілька пакетів можуть бути об'єднані в один за допомогою бінарної операції "Виключаюче АБО". А для змінених пакетів, що дійшли до одержувача, складається система рівнянь, за якою можна відновити вихідні пакети. Оскільки замість двох окремих послідовних передач пакетів відбувається лише одна передача, то звільняються ресурси передачі інших пакетів, що призводить до збільшення швидкості потоків.

Методи стиснення даних, що передаються дозволяють збільшити швидкість передачі корисного навантаження за рахунок зменшення обсягу даних, що фактично передаються. Застосування методів стиснення даних, що передаються, стало можливим завдяки збільшеним обчислювальним

можливостям мережевих пристроїв, що дозволяє проводити додаткові операції над даними без відчутних затримок. Сучасні методи стиснення дозволяють динаміці визначати найкращі параметри стиснення [3] та проводити необхідні операції прозоро для рівня додатків [4].

Методи управління швидкістю передачі необхідні для запобігання перевантаження у мережі – ситуації, коли швидкість надходження пакетів перевищує швидкість відправлення пакетів. Під час перевантаження надлишок пакетів може бути збережено тимчасово в буфері на пристрої, однак після його переповнення зайві пакети будуть скинуті та повинні бути відправлені повторно. Дослідження [5] показало, що постійні повторні передачі пакетів, що виникають через перевантаження в мережі, призводять до істотного зниження швидкості з'єднання (майже в 1000 разів), тому необхідно контролювати кількість пакетів, що одночасно знаходяться в мережі для кожного з'єднання. Для цього було розроблено алгоритми управління перевантаженням. Однак зниження кількості пакетів, що відправляються, може призвести до недовикористання доступної пропускної спроможності в мережі, тому необхідно розробляти алгоритми управління перевантаженням, придатні для певних умов в мережі (пропускна здатність і затримка в мережі, можливості мережного обладнання, розподіл навантаження в мережі і т.д.). В даний час продовжується активний розвиток алгоритмів управління навантаженням, наприклад, для алгоритму BBR [6] від Google вносяться зміни до ядра операційної системи Linux.

Методи сегментування дозволяють розбити одне транспортне з'єднання кілька послідовних з'єднань за допомогою додаткових серверів на маршруті, званих проксі.

1 МЕТОДИ БАГАТОПОТОЧНОЇ МАРШРУТИЗАЦІЇ

Як було зазначено у вступі, при розробці багатопотокового протоколу постають завдання розподілу транспортних сегментів даних між спорідненими підпотоками, розробки алгоритму управління навантаженням у споріднених підпотоках, регулювання кількості підпотоків та побудови маршрутів для кожного із споріднених підпотоків. Розглянемо докладніше завдання регулювання кількості споріднених підтоків.

Можна виділити два класи методів на вирішення зазначеної проблеми: статичний і динамічний. У статичних методах кількість споріднених підтоків для транспортного сполучення встановлюється у кожному випадку заздалегідь і не змінюється з часом. Кількість споріднених підтоків може залежати від налаштувань користувача або кількості мережних інтерфейсів на пристрої відправника та одержувача (у цьому випадку підтікання відкривається для кожної пари інтерфейсів).

У динамічних методах кількість споріднених підтоків на транспортному сполученні залежить від вимог додатку до якості сервісу та стану мережного оточення. Коли ресурсів маршрутів поточної безлічі споріднених підпотоків не вистачає для забезпечення необхідної якості сервісу, відбувається відкриття додаткового підпотоку. Наприклад, на рисунку 1.1 показаний випадок, коли між відправником та одержувачем необхідно забезпечити передачу даних зі швидкістю 10 Мб/с. Спочатку було створено лише один підпотік, маршрут якого виділено жовтим кольором у верхній частині топології мережі. Причому можемо вважати, що зі старту з'єднання пропускна спроможність жовтого маршруту була більше 10 Мб/с, тобто. одного підпотоку вистачало задля забезпечення необхідної якості сервісу. Однак настає момент, коли пропускна спроможність жовтого маршруту знижується до 8 Мб/с. Наприклад, через появу нових транспортних потоків, маршрут яких частково чи повністю збігається з

жовтим маршрутом. Так як пропускної спроможності маршруту одного підпотоку не вистачає для забезпечення необхідної якості сервісу, відповідно до динамічного принципу багатопоточної маршрутизації відбудеться відкриття додаткового нового підпотоку. Маршрут нового підпотоку зображений блакитною лінією, його пропускна спроможність дорівнює 8 Мбіт/с, та сумарна пропускна спроможність маршрутів двох підтоків дозволить задовольнити вимоги до якості сервісу.

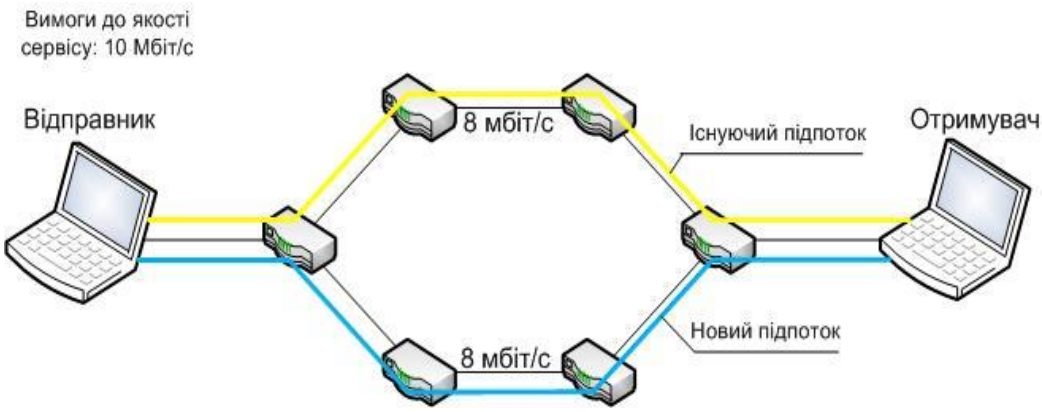

Рисунок 1.1 – Приклад роботи динамічного багатопоточного методу маршрутизації

У разі, коли вимоги користувача можуть бути задоволені меншою кількістю споріднених підтоків, зайві підтоки будуть завершені, тим самим звільняючи ресурси мережі для обслуговування інших потоків. Наприклад, на рисунку 1.1 другий підпотік з блакитним маршрутом може бути закритий, якщо пропускна здатність жовтого маршруту знову перевищить 10 Мб/с.

Виникає питання, який з методів багатопоточної маршрутизації – статичний чи динамічний, буде краще в тій чи іншій ситуації. Незважаючи на простоту реалізації статичного методу, можна навести приклад, коли його використання призводить до неефективного використання ресурсів мережі. Розглянемо приклад рисунку 1.2. Припустимо, що з кожним маршрутизатором перебуває користувач мережі і йде передача даних між

кожною парою користувачів. Також припускатимемо, що пропускна здатність всіх ліній зв'язку в представленій топології дорівнює 1. Тоді сумарна швидкість, яку можна отримати, використовуючи класичні однопотокові протоколи, дорівнює 6 (по 1 в кожну сторону на кожній лінії зв'язку.

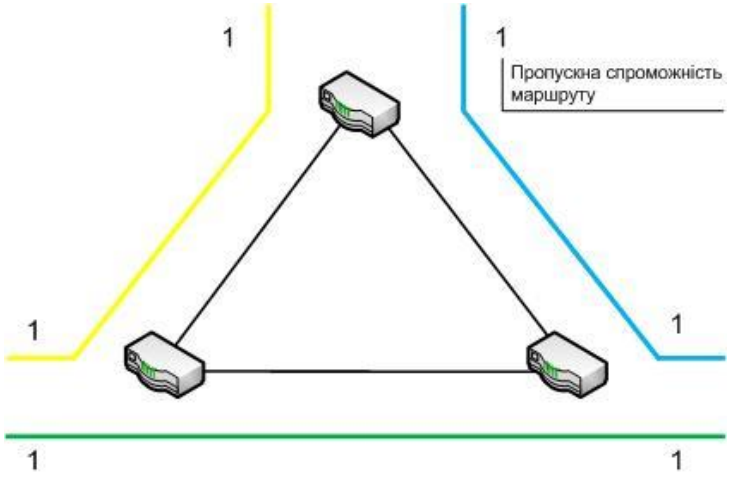

Рисунок 1.2 – Приклад топології «трикутник», коли методи багатопоточної маршрутизації не використовуються

Можна помітити, що між кожною парою користувачів існує два маршрути, що не перетинаються, тому в статичному методі багатопоточної маршрутизації логічно використовувати два підпотоки для кожного багатопоточного з'єднання. Приклад організації такої передачі зображений на рисунку 1.3, де маршрут другого підтоку позначений пунктирною лінією. Щоб оцінити швидкість багатопоточних з'єднань, зробимо додаткове припущення, що пропускна спроможність кожної лінії зв'язку буде ділитися порівну між підтоками, маршрути яких проходять через цю лінію зв'язку (таке припущення можна зробити, оскільки маршрути споріднених підтоків не мають спільних вузьких місць). Так як пропускна здатність кожної лінії зв'язку дорівнювала 1, а для аналізованої ситуації через одну і ту ж лінію зв'язку проходять 3 підпотоку, то швидкість кожного підпотоку

дорівнюватиме $\frac{1}{3}$. Тоді швидкість багатопоточного з'єднання буде дорівнювати $\frac{2}{3}$, а сумарна швидкість потоків – $\frac{2}{3} \times 6 = 4$. Отримуємо, що швидкість у разі використання статичного методу багатопоточної маршрутизації буде менше, ніж швидкість у разі без використання багатопоточних протоколів. Якщо ми будемо використовувати динамічний метод багатопоточної маршрутизації, то подібної ситуації можна буде уникнути, оскільки другий підтік не буде створюватися через непотрібність або закриється, оскільки не буде збільшення швидкості потокового з'єднання.

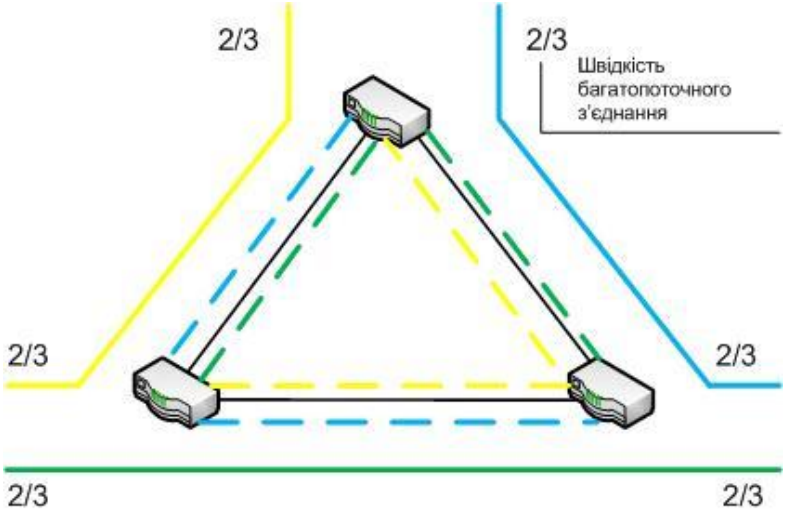

Рисунок 1.3 – Приклад топології «трикутник», коли використовується статичний метод багатопоточної маршрутизації

Таким чином, стає актуальним завдання порівняння ефективності статичного та динамічного методів багатопоточної маршрутизації та визначення умов їх ефективного застосування.

1.1 Аналіз робіт з ефективності багатопоточної маршрутизації

У роботах, спрямованих на аналіз ефективності багатопоточної маршрутизації, в більшості випадків як основний критерій досліджують

приріст швидкості транспортного сполучення, при статичному багатопоточному методі у певному середовищі передачі даних або за певного навантаження. Розглядається збільшення продуктивності статичного методу під час використання різних алгоритмів планування сегментів між підпотоками. Автори статті [7] досліджують ефективність статичного методу для автомобілів, підключених до мережі. Ці автомобілі можуть координувати свою роботу між собою з міськими службами дорожнього руху, а також виконувати допоміжні обчислення на інших мережних пристроях. Робота присвячена дослідженню продуктивності статичного методу багатопоточної маршрутизації в бездротових децентралізованих мережах, що самоорганізуються. У статті [8] розглядається вплив різних алгоритмів управління навантаженням у разі статичного багатопотокового методу. Автором цієї роботи було проведено дослідження масового використання статичного методу багатопоточної маршрутизації в рамках однієї автономної системи. Дослідження показало, що застосування статичного багатопоточного транспортного сполучення має перевагу перед однопотоковим з'єднанням протоколу TCP тільки за умови завантаження мережі, меншої 37% [8]. Під завантаженням мережі розуміється середнє значення завантаження ліній зв'язку, де завантаження лінії зв'язку розраховувалася як відношення сумарної швидкості TCP потоків, що проходять цю лінію зв'язку, до її пропускної спроможності.

З робіт із аналізу ефективності динамічного методу многопоточной маршрутизації можна назвати лише роботу [9], у якій розглядається ефективність динамічного методу для бездротових мереж.

Для проведення порівняльного аналізу статичного та динамічного методів багатопоточної маршрутизації необхідно вибрати протоколи, на яких буде досліджено ефективність роботи методів. Вибір проводився за такими критеріями:

- клас методу. Багатопотоковий транспортний протокол може відповідати статичному або динамічному методу;

- підтримка протоколу. Для отримання актуальних достовірних результатів необхідно, щоб реалізація багатопоточного протоколу була відкритою та доступною, а також мала діючу підтримку, щоб відповідати поточним вимогам мереж;

- опорний транспортний протокол. Багато протоколів використовують такі поширені транспортні протоколи, як TCP, UDP, для того щоб спростити впровадження цих протоколів в існуючу інфраструктуру мережі;

- можливість використання довільного алгоритму управління навантаженням. Так як розвиток інших методів забезпечення якості сервісу не стоїть на місці, то важливо, щоб багатопотоковий протокол міг використовувати сучасні алгоритми управління навантаженням у своїй роботі.

Аналіз подібних робіт показав, що найбільш популярним багатопоточним протоколом, що підтримує статичний спосіб, є Multipath TCP (MPTCP) протокол [10]. В обох випадках багатопоточний протокол буде складніше інтегрувати в існуючу мережу інфраструктури. З останніх робіт можна також відзначити багатопотоковий протокол MP-QUIC [8], який використовує UDP як опорний транспортний протокол. Однак протокол MP-QUIC заснований на тих же принципах роботи, що і протокол MPTCP, тому протокол MPTCP був обраний як основний представник статичного протоколу багатопоточного для транспортних з'єднань для проведення порівняльного аналізу ефективності статичного і динамічного методів багатопоточної маршрутизації. Крім цього, протокол MPTCP має доступну відкриту реалізацію, що підтримується, на відміну від протоколу MP-QUIC.

Динамічний алгоритм з роботи [8] називатимемо BNDEEI-MPTCP (Bandwidth-Need Driven Energy Efficiency Improvement of MPTCP). Побудована модель роботи методу багатопоточної маршрутизації для BNDEEI-MPTCP як один з критеріїв відкриття підпотоку використовує енергоефективність каналу, що мало застосовується для провідних мереж Інтернет-провайдерів. Крім того, BNDEEI-MPTCP спирається на

припущенні, що використовується алгоритм управління навантаженням, який визначає перевантаження по втраті пакета (loss-based алгоритми). Це не дозволить використовувати такі сучасні алгоритми керування навантаженням як BBR [7] у багатопотоковому транспортному протоколі. Тому як представник динамічного методу в порівняльному аналізі в роботі використано протокол Flow De-Multiplexing Protocol (FDMP), запропонований Смілянським Р. Л. та розроблений та досліджений автором цієї роботи. Результати проведеного огляду з найбільш значних багатопоточних протоколів підсумовано в таблиці 1.1

Таблиця 1.1 – Порівняння багатопотокових транспортних протоколів

Протокол	Клас	Підримка	Опорний транспортний протокол	Алгоритм керування перенавантаженням
MPTCP	Статичний	Так	TCP	Будь-який
SCTP-CMT	Статичний	Ні	SCTP	Будь-який
MPQUIC	Статичний	Ні	UDP	Будь-який
BNDEEI-PTCP	Динамічний	Ні	TCP	Loss-based

Головний висновок зі сказаного вище, полягає в тому, що відсутні роботи, присвячені порівнянню ефективності статичного та динамічного методів багатопоточної маршрутизації, а для проведення такого порівняння було обрано протоколи MPTCP та FDMP відповідно. У наступних розділах наведено більш детальний опис протоколів MPTCP і FDMP, який буде необхідний для розуміння пристрою експериментального стенду для порівняльного аналізу.

1.2 MPTCP

Multipath TCP (MPTCP) – це модифікація протоколу TCP, що підтримує статичний метод багатопоточної маршрутизації.

Розробники протоколу MPTCP наводять принципи, якими керуються для спрощення інтеграції свого протоколу до сучасних мереж [3]:

- сегменти даних, що передаються протоколом, повинні відповідати специфікації TCP протоколу: структура заголовка сегмента даних, його максимальний розмір, а також процедура встановлення та розриву транспортного з'єднання, щоб знизити ризики несумісності з комутаційним обладнанням;

- протокол повинен залежати від припущень відсутності фрагментації сегмента даних чи складу полів розширення заголовка сегмента (опціональних полів);

- протокол повинен використовувати існуючий інтерфейс протоколу TCP для роботи з додатками;

- транспортний агент повинен мати можливість автоматично вибирати, за яким протоколом – TCP або MPTCP, необхідно працювати. Така можливість необхідна у разі неможливої подальшої роботи за протоколом MPTCP.

Протокол MPTCP досягає поставлених цілей за рахунок використання протоколу TCP з механізмом синхронізації управління вікном перевантаження для кожного зі своїх підтоків та передачі службової інформації в опціональній частині транспортного заголовка пакетів. Опишемо основну схему функціонування протоколу MPTCP.

Встановлення з'єднання.

Установка з'єднання відбувається процедурою триразового рукостискання: обміном сегментами із прапорами SYN, SYN/ACK, ACK, які перебувають у транспортному заголовку сегмента. Кожен зазначений сегмент містить функцію MP_CAPABLE транспортного заголовка TCP. Ця опція вказує на підтримку роботи протоколу Multipath TCP і прагнення використовувати цей протокол на даному з'єднанні.

У разі відсутності опціонального поля в заголовку сегмента в будь-якому з пакетів триразового рукостискання установка з'єднання та подальша

його робота буде йти згідно TCP протоколу. Опція MP_CAPABLE містить 64-бітний ключ, який згодом використовуватиметься для визначення споріднених підтоків. У сегменті з прапором SYN знаходиться ключ відправника для цього з'єднання, у сегментах з прапорами SYN/ACK ключ одержувача для цього з'єднання, а в сегменті з прапором ACK – ключ відправника, за яким слідує ключ одержувача.

Запуск нового підпотоку.

Новий споріднений підтік запускається обміном сегментів з прапорами SYN, SYN/ACK, ACK, як і в процедурі трикратного рукостискання для встановлення з'єднання протоколу TCP. Кожен зазначений сегмент включається опція MP_JOIN. Опція MP_JOIN містить криптографічні хеші від ключів, якими обмінялися відправник та одержувач у опції MP_CAPABLE, які ідентифікують, до якого багатопоточного з'єднання підключається підтік.

У випадках, коли в сегментах з прапорами SYN, SYN/ACK, ACK у мережі була видалена опція MP_JOIN або неправильно завершилася автентифікація, надсилається сегмент з прапором RST для скасування встановлення з'єднання для нового підтоку.

Функції агента MPTCP у ході життя транспортного сполучення.

Підтримка біоактивного відображення між порядковими номерами сегментів, що використовуються на підпотоках, та початковими номерами, що показують порядок, у якому формувалися сегменти на відправнику при надходженні даних від програми. Це відображення потрібне для відновлення правильного порядку даних, що прийшли до одержувача.

Керування потоком з використанням єдиного буфера для споріднених підпотоків.

Підтвердження сегментів на рівні підпотоку та на рівні багатопоточного з'єднання.

Вся службова інформація, необхідна реалізації описаної вище функціональності, передається між транспортними агентами в опціональної частини транспортного заголовка протоколу TCP.

Завершення підключення.

ССегмент із прапором FIN у протоколі MPTCP має значення завершення для підтоку. Щоб завершити передачу на рівні з'єднання, опціональної частини транспортного заголовка передається прапор DATA_FIN. Після того, як буде отримано підтвердження на цей сегмент (спочатку мають бути підтверджені всі дані), транспортні агенти повинні надіслати сегмент із прапором FIN на всі підтоки.

Варіанти налаштування агента MPTCP, які може задати користувач:

Default – агент на відправнику працює як звичайний агент протоколу TCP і не ініціює відкриття нових підтоків.

Fullmesh – у кожного відправника і у кожного одержувача є свій пул IP адрес, агент відкриває підпотоки за принципом «кожний з кожним»: з кожної IP адреси відправника відкривається підтік на кожну IP адресу одержувача. Наприклад, якщо пул IP адрес кожної містить 3 адреси, то всього у нас буде 9 підтоків. Усі підтоки створюються під час відкриття з'єднання.

Ndiffports – агент відкриває N потоків між IP адресою одержувачем та IP адресою відправником, запитуючи нові TCP порти у операційної системи. Параметр N налаштовується засобами операційної системи, всі підпотоки створюються під час відкриття багатопотокового з'єднання.

Протокол MPTCP надає можливість використовувати заздалегідь задану кількість маршрутів між одержувачем та відправником, що може підвищити якість сервісу. Проте протокол MPTCP не дозволяє динамічно ні відкривати новий підпотік, ні закривати підпотоки за потреби. Також протокол MPTCP не відповідає за побудову маршрутів, через що побудовані маршрути в мережі (згідно з діючею в мережі політикою маршрутизації) можуть перетинатися і збігатися.

1.3 FDMP

У цьому розділі представлений протокол FDMP - один з варіантів динамічного методу багатопоточної маршрутизації, згідно з яким транспортний агент відправника відкриває або закриває споріднені підтоки в залежності від поточної завантаженості мережі та вимог якості сервісу. Протокол FDMP є багатопоточним протоколом, що реалізує схожу функціональність з протоколом MPTCP, але в якому алгоритм, який відповідає за регулювання кількості споріднених підтоків, працює інакше. Далі докладніше розглянемо протокол FDMP, вказуючи на його відмінності від MPTCP.

Умова відкриття підпотоку.

На перший погляд, транспортний агент FDMP протоколу повинен відкрити новий підтік, якщо швидкість багатопотокового з'єднання (визначувана як сума швидкостей споріднених підтоків) менше заданої нижньої межі якості сервісу для швидкості передачі. Проте це завжди так. Розглянемо приклад:

Нехай програма надсилає по мережі дані, введені користувачем із клавіатури. Через обсяг даних, що передаються по мережі, швидкість потоку буде невелика і, можливо, менше заданої межі якості сервісу. Але відкривати новий підтік для збільшення швидкості не має сенсу, оскільки сама програма відправляє дані на TCP рівень мережного стека операційної системи з меншою швидкістю. З цього прикладу випливає, що відповідь на питання – відкрити новий підпотік чи ні, залежить від швидкості передачі даних між додатком та транспортним рівнем мережного стеку протоколів операційної системи.

Розглянемо ще один приклад: нехай йде стабільна передача даних по мережі зі швидкістю більшою, ніж вимагає програма. У якийсь момент часу в мережі відбувається навантаження і швидкість з'єднання падає. На короткий момент часу швидкість стає нижчою за межу якості сервісу і FDMP-агент

відкриває новий підпотік. Але потім перевантаження закінчилося, і швидкість знову задовольняє якість сервісу, тому новий FDMP-подпоток був даремно відкритий. Отже, під час вирішення питання про відкриття нового потоку необхідно використовувати дані моніторингу з'єднання, а чи не просто результат одноразового виміру.

Таким чином, для визначення умови відкриття підтоку необхідно:

- знайти спосіб детектувати завантаженість наданих каналів;
- запропонувати спосіб вимірювання поточної пропускної спроможності багатопотокового з'єднання;

Розрахунок швидкості багатопоточного з'єднання.

Визначимо середню швидкість FDMP-з'єднання наступним чином:

$$V_{FDMP} = \frac{\widetilde{V(t_2)} - \widetilde{V(t_1)}}{t_2 - t_1}, \quad (1.1)$$

де $\widetilde{V(t)}$ – обсяг даних, які були передані та підтверджені з моменту початку з'єднання до часу t. Отримувана величина є середньою швидкістю передачі даних у мережі в період часу з t_1 до t_2.

Для розрахунку середньої швидкості за визначеною вище формулою транспортний агент FDMP зберігає інформацію щодо відправлених сегментів: обсяг даних, відправлених з моменту встановлення з'єднання, та час підтвердження про доставку сегмента одержувачу. Інформація зберігається в кільцевому буфері qos_buff, кожен із елементів якого містить такі поля:

- t_{rcv} – час, коли настало підтвердження на відправлений сегмент;
- $\tilde{V}$ – обсяг даних, які були передані додатком та підтверджені з початку встановлення з'єднання до часу t_{rcv}.

При цьому в транспортному агенті FDMP інформація записується не по кожному сегменту, а відразу по групі сегментів розмір якої є параметром транспортного агента. При такому підході зібрані дані відображають середню швидкість за тривалий проміжок часу, не займаючи надмірну кількість пам'яті операційної системи. Альтернативою було вважати не сегменти, а кількість відправлених байтів або робити записи через рівні часові інтервали, проте при відправленні сегментів невеликого розміру в першому випадку ми швидше зможемо дізнатися про зміну швидкості передачі, ніж у двох інших випадках.

Також у транспортного агента є кілька вказівників на елементи цього буфера:

- app_head – вказує на останній доданий у буфер елемент (інформація по сегменту, який був відправлений в мережевий стек додатком останнім, і, можливо, ще не підтверджений). Позначатимемо відповідний елемент буфера верхнім індексом ah, наприклад, запис $\widetilde{V^{ah}}$ позначає поле $\tilde{V}$ елемента буфера qos_buff, який посилається покажчик app_head;

- rcv_head – вказує на елемент нашого буфера, що містить останній підтверджений сегмент. Позначатимемо відповідний елемент буфера верхнім індексом rh;

- $tail$ – вказує на елемент, який був доданий раніше за всіх у буфер (оскільки буфер кільцевий, то, починаючи з деякого часу, елементи буфера будуть перезаписуватися новими). Позначатимемо відповідний елемент буфера верхнім індексом $tail$.

Таким чином, уточнимо розрахунок середньої швидкості FDMP з'єднання за наявності буфера:

$$V_{FDMP} = \frac{\widetilde{V^{rh}} - \widetilde{V^{tail}}}{t_{rcv}^{rh} - t_{rcv}^{tail}}. \qquad (1.2)$$

Застосування формули (2) вважається неможливим, якщо кількість елементів буфера qos_buff менша за заданий поріг, що є параметром транспортного агента. Це обмеження було введено, щоб розраховувати середню швидкість FDMP з'єднання, а не миттєву.

Також може виникнути ситуація, коли в буфері будуть надто старі дані, які можуть вплинути на швидкість у середньому. Наприклад, з'єднання простоювало кілька годин через відсутність даних, а потім передача даних відновилася. Середня швидкість, що розраховується за формулою (2), буде дуже малою, тому що в кільцевому буфері qos_buff містяться дані двогодинної давності. Тому в транспортному агенті FDMP при розрахунку швидкості згідно з формулою (2) використовується відсікання за часом: покажчик tail зрушується на елемент, час отримання якого t_{rcv} було не більше, ніж T^{max}секунд тому, де T^{max} – параметр транспортного агента.

Визначення вузького місця

Раніше було зазначено, що можуть виникати ситуації, коли програма відправляє на транспортний рівень дані повільніше, ніж швидкість транспортного рівня. І тут відкривати новий родинний підпотік не потрібно, т.к. вузьким місцем є програма, а не лінія зв'язку в мережі. Щоб відрізнити таку ситуацію, розглянемо докладніше роботу мережного стека операційної системи Linux. Програма надсилає дані в мережу, використовуючи системні виклики send, sendto, sendmsg, write під час роботи з сокетом. Всі перелічені системні виклики зрештою закінчуються викликом функції сокету sendmsg, яка надсилає дані на наступний рівень мережного стека (транспортний). У разі протоколу TCP (і його розширення FDMP) ця функція призводить до виклику функції tcp_sndmsg.

Функція tcp_sndmsg отримані дані додає у двозв'язний список sk_buff (рисунок 4), кожен елемент якого входить окремий TCP сегмент [5].

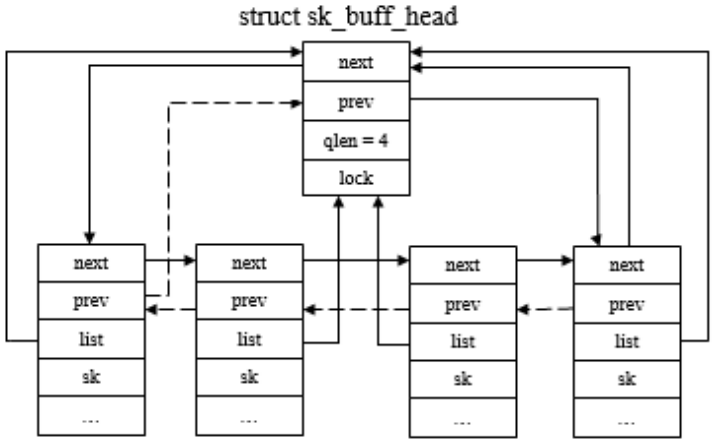

Рисунок1.4 – Двозв'язковий список sk_buff

Після підтвердження сегмента елемент, який відповідає за цей сегмент, видаляється зі списку.

Необхідно розрізняти наступні два випадки:

- оскільки програма запитує необхідну йому пропускну здатність один раз при відкритті TCP з'єднання, то може виникнути випадок, коли програма відправляє дані зі швидкістю меншою, ніж було затребувано. У цьому випадку транспортний агент FDMP відправляє дані, що надходять, і чекає надходження наступних сегментів. У результаті $V_{FDMP} < V_{QoS}$ (V_{QoS} – вимоги, що пред’являються до якості сервісу з'єднання), і новий споріднений підпотік відкривати не потрібно, так як швидкість надходження даних з рівня придатності;

- додаток повністю заповнює список sk_buff, розмір якого обмежений розміром буфера пам'яті, що виділяється для цього списку при ініціалізації сокету, і блокується доти, доки не звільниться місце в списку. Сегменти зі списку видаляються, коли надходить підтвердження про їх доставку одержувачу. Через війну одержувач додаватиме дані у перелік зі швидкістю, з якою транспортний агент FDMP відправляє дані, тобто. $V_{FDMP} < V_{QoS}$, але новий споріднений підтік потрібно відкрити для збільшення швидкості FDMP з'єднання.

Зазначені випадки невиразні у формулах (1)-(2), тому для остаточного визначення ситуації, коли необхідно відкрити новий споріднений підпотік, потрібно враховувати дані моніторингу буфера sk_buff.

Як варіант, можна було б при повному заповненні списку sk_buff сигналізувати про те, що буфер повний і за умови, що швидкість FDMP з'єднання менша за вимогу якості сервісу, відкривати новий підпотік. Однак передача даних з рівня програми на транспортний рівень може бути нерівномірною. Розглянемо для прикладу передачу потокового відео, в якій кодек MPEG4 [5] для стиснення та розпакування деяких відео-фреймів використовує як вже сформовані відео-фрейми, так і наступні відео-фрейми, на формування яких необхідно чекати. В результаті передача має характерні сплески активності у певні моменти часу, поза якими передача даних не ведеться (рисунок 1.5). Під час передачі даних список sk_buff може заповнитись повністю, проте під час затишшя буфер звільняється, і канал у цей час порожній. В даному випадку відкриття нового підтоку не буде потрібно, оскільки новий підток буде здебільшого не діяти, а з відправкою накопичених даних у буфері впораються й інші підпотоки під час бездіяльності.

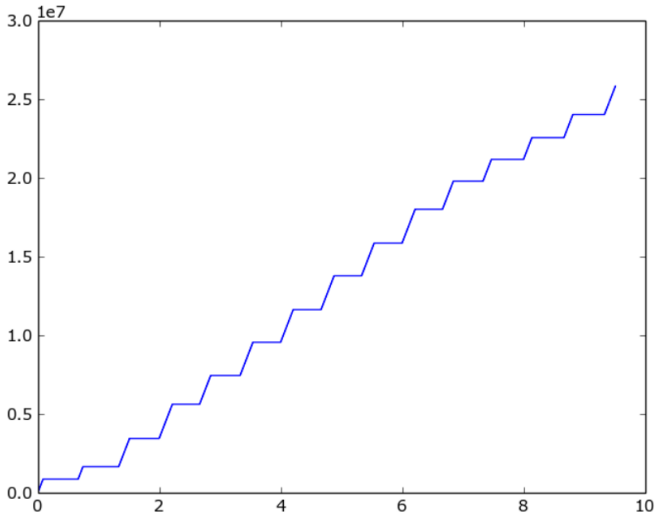

Рисунок 1.5 – Графік залежності кількості переданих потокових бітів відео від часу (сек)

Тому транспортний агент FDMP оцінює середню заповненість буфера, і у випадку, якщо в середньому список sk_buff заповнений на K% (назвемо цю умову IS_BUFF_FULL) і швидкість FDMP з'єднання менша за вимогу якості сервісу, потрібно відкрити новий споріднений підпотік.

Отже, умову відкриття підтоку можна записати у такому вигляді:

$$IS\_BUFF\_FULL \;\&\&\; (V_{FDMP} < V_{QoS}). \qquad (1.3)$$

Модель роботи транспортного агента FDMP

Тепер розглянемо пропоновану в цій роботі модель роботи транспортного агента FDMP, згідно з якою відбувається зміна числа споріднених підтоків. Її ідея схожа з ідеєю алгоритмів управління навантаженнями в мережі.

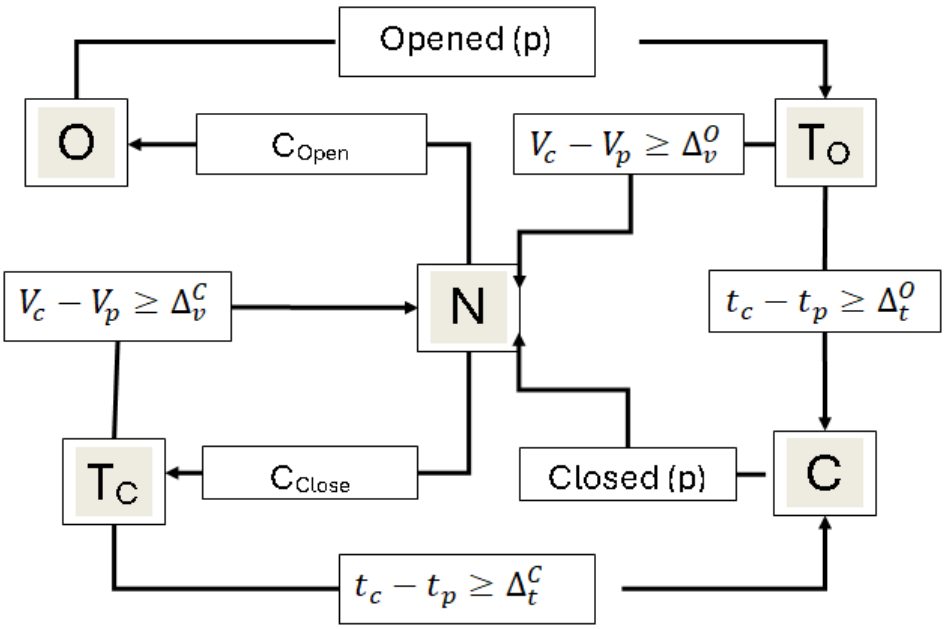

Рисунок 1.6 – Діаграма станів моделі роботи транспортного агента

У цих алгоритмах розмір вікна перевантаження cwnd змінюється в залежності від ситуації в мережі: при неповній завантаженості каналу розмір вікна збільшується, а при повному завантаженні зменшується. Аналогічно

наведеному прикладу, FDMP протокол динамічно змінює кількість споріднених підтоків: якщо швидкість з'єднання не задовольняє вимогам якості сервісу, то відкривається новий підток, якщо швидкість з'єднання перевищує видані обмеження, то ми намагаємося закрити зайві підтоки. Закриття підтоків необхідно для звільнення ресурсів операційної системи, що надмірно використовуються (наприклад, пам'ять, що виділяється для підтоку), а також ресурсів мережі (наприклад, правила в таблиці комутації). Цю модель можна подати у вигляді діаграми станів кінцевого автомата (рисунок 1.6).

Розглянемо представлені на рисунку 1.6 станів:

- N – нейтральний стан, у ньому ми не відкриваємо і не закриваємо підпотоки, а лише перевіряємо вимоги якості сервісу з періодом tN. Якщо умову відкриття підпотоку (3) неможливо перевірити через невелику кількість елементу в буфері qos_buff, ми залишаємося в нейтральному стані. Також може вийти, що умова відкриття підпотоку (3) виконано, але ми не можемо відкрити новий підпотік через обмеження кількості підпотоків, тоді ми залишаємося в нейтральному стані;

- O – стан відкриття нового підтоку. У цей стан можна перейти з нейтрального при виконанні умови (3) відкриття нового підтоку;

- TO – тест на відкриття нового підпотоку. У цьому стані кожні tO секунд перевіряється зміна швидкості з'єднання з новим додатковим підтоком порівняно з колишньою швидкістю без цього підтоку. У разі збільшення швидкості ми переходимо в нейтральний стан, інакше даний підпотік не приносить жодної користі, лише розтрачуючи ресурси операційної системи і ресурси мережі, тому ми закриваємо підпотік, що відкрився, в стані C;

- C – стан закриття підтоку. Перериваємо з'єднання на даному підпотоці, надсилаючи сегменти з прапором RST;

- TC – тест на закриття підтоку. У цей стан ми потрапляємо з нейтрального у випадку, якщо пропускна спроможність багатопотокового

з'єднання більша за вимогу якості сервісу та кількість підтоків більше одного. У даному стані ми вибираємо випадковим чином підтік, через який перестаємо надсилати дані (у поточній реалізації використовується round-robin). Кожні t_C секунд ми перевіряємо умову (3) і у разі її виконання переходимо у стан відновлення R, оскільки менша кількість підтоків не здатна задовольнити вимоги якості сервісу на даний момент. Якщо ж протягом T_c^{max} секунд умова (3) не виконується, то ми переходимо у стан закриття, де закриємо цей підпотік. T_c^{max} є параметром транспортного агента;

- R – стан відновлення. На час R^{max} не відбувається жодних перевірок, оскільки на значення середньої швидкості впливало припинення відправлення даних на один із підтоків при тесті на закриття. R^{max} є параметром транспортного агента. Потім відбувається перехід у нейтральний стан. T_c^{max}

1.4 Багатопотокова маршрутизація до ПКМ

Розглянуті в попередніх розділах багатопотокові протоколи працюють усередині транспортних агентів операційних систем, які не мають жодної інформації про структуру мережі. Тому завдання вибору та встановлення маршруту є завданням самої автономної системи мережі. Однією з умов для збільшення швидкості багатопоточного з'єднання є відсутність перетинів між маршрутами споріднених підтоків. Ця умова пов'язана з тим, що на місці перетину маршрутів споріднених підтоків може утворитися вузьке місце в мережі – канал із найменшою залишковою пропускною спроможністю. Тоді споріднені підпотоки не збільшуватимуть швидкість багатопоточного з'єднання, а поділятимуть ресурси однієї і тієї ж лінії зв'язку, хоча цими ресурсами міг мати один підпотік.

Однак варто зазначити, що в мережах Інтернет-провайдерів, існують топології, в яких знайдеться як мінімум одна пара вузлів, що не мають маршрутів, що не перетинаються між собою. Цей факт може бути пов'язаний

з неможливістю фізичної прокладки запасного маршруту між вузлами, що розглядаються. У цьому випадку Інтернет-провайдер може використовувати лінії з більшою пропускною здатністю, щоб впоратися зі збільшеним навантаженням, яке може надходити на цей маршрут із кількох місць. Для такого випадку, коли в мережі фізично відсутні маршрути, що не перетинаються, ми будемо розглядати завдання побудови маршрутів, що мають мінімальну кількість перетинів.

У традиційній мережі у разі кількох маршрутів проводиться балансування транспортних потоків між цими маршрутами. Визначення маршруту, яким вирушить транспортний потік, найчастіше відбувається на підставі хеш-функції, взятої від значень полів заголовків канального, мережного та іноді транспортних рівнів. Навіть якщо для розрахунку хеш-функції використовується інформація про транспортні порти із заголовка транспортного рівня, які відрізнятимуться для споріднених підтоків, ніхто не може гарантувати, що хеш-функція не видасть однакових значень для споріднених підтоків. А якщо інформація з транспортного заголовка не використовується, то споріднені підтоки матимуть однакові маршрути.

Крім завдання розподілу родинних підтоків між маршрутами виникає завдання побудови цих маршрутів. У традиційної мережі це завдання розв'язати складніше через децентралізоване управління, тобто. відсутність поточного стану мережі. Тому в традиційних мережах для підтримки багатопоточної маршрутизації необхідно додавати додаткову функціональність маршрутизації багатопоточних з'єднань в мережеве обладнання, що ускладнить розгортання багатопоточної маршрутизації.

У програмно-конфігурованих мережах [6] використовується централізоване управління, яке виноситься в окремий логічний пристрій, який далі називатимемо контролером ПКМ. Контролер ПКМ знає поточний стан мережі та може керувати установкою побудовою маршрутів. Розглянемо підходи до вирішення цього завдання.

В роботах [5,7], присвячених дослідженню багатопоточної маршрутизації в ПКМ, використовують для побудови маршрутів жадібний алгоритм, який далі будемо називати Greedy Shortest Path First (GSPF). На кожній ітерації алгоритм GSPF будує найкоротший маршрут нового спорідненого підтоку. Наприкінці кожної ітерації до ребер маршруту, отриманого алгоритмом GSPF, додається штраф. Штраф підбирають так, щоб маршрути з перетином були менш кращими, ніж маршрути без перетинів.

Перевагами алгоритму GSPF є його простота для пошуку нового маршруту для спорідненого підтоку на запит. Проте алгоритм GSPF який завжди призводить до оптимального рішення, тобто. не завжди знаходить непересічні маршрути, коли вони є. Наприклад, розглянемо топологію, зображену на рисунку 1.7. Нехай необхідно передати дані, за допомогою динамічного багатопоточного протоколу. Припустимо, що для цього використовуються два споріднені підпотоки між вузлами U і A. Тоді алгоритм GSPF для першого підпотоку побудує найкоротший маршрут (вважатимемо, що метрикою відстані є кількість мережевих вузлів на маршруті), позначений жовтим кольором. Але не можна підібрати жодного маршруту для другого спорідненого підтоку, без перетину з маршрутом першого підтоку. Однак саме безліч маршрутів, що не перетинаються, існує зображено на рисунку 1.8 (жовтим і блакитним кольором відповідно).

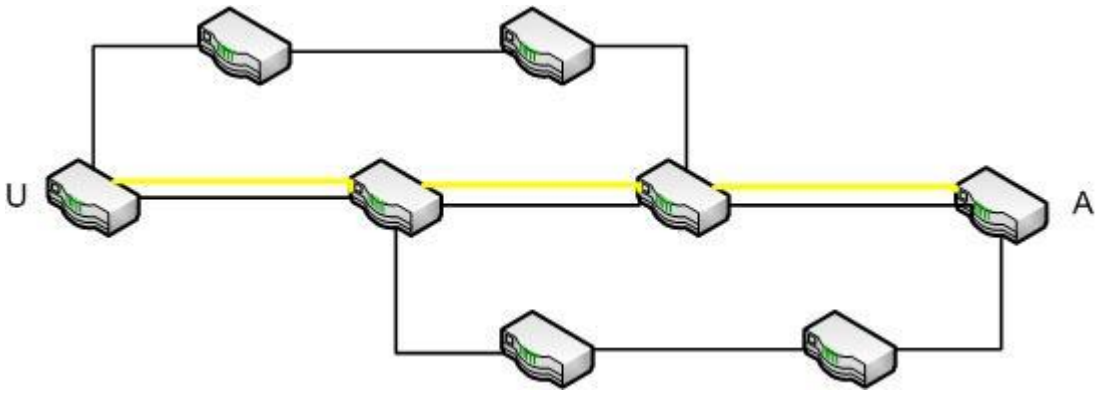

Рисунок 1.7 – Приклад неоптимальної роботи алгоритму GSPF

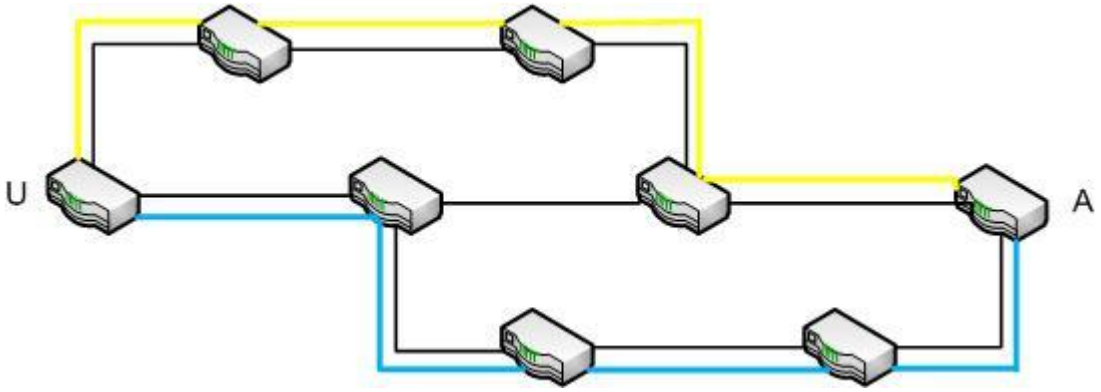

Рисунок 1.8 – Приклад маршрутів, що не перетинаються, не знайдених алгоритмом GSPF

Таким чином, алгоритм GSPF не підходить для пошуку маршрутів, що не перетинаються. З цією метою можна використовувати алгоритм побудови надлишкових дерев (Maximum Redundant Tree – MRT) [11]. Декілька побудованих дерев MRT дозволяють відправляти пакети не тільки по дереву найкоротших відстаней, а й балансувати транспортні потоки по MRT-деревах. Якщо розглядати маршрути від однієї точки до кореня в різних MRT деревах, то можна отримати безліч маршрутів з мінімальною кількістю перетинів. Однак алгоритм, що використовується в мережах [11] дозволяє будувати тільки 2 дерева, тобто. можна отримати лише 2 маршрути з мінімальним перетином. Слід зазначити, що складність алгоритму побудови двох дерев лінійно залежить від кількості вершин в топології. Існують алгоритми побудови трьох MRT дерев [12] та чотирьох MRT дерев [13] з квадратичною та кубічною складністю відповідно, проте ці алгоритми працюють для спеціальних видів графів (наприклад, трьох реберно-зв'язкових [12]). Робот з побудови п'яти та більше MRT дерев знайдено не було.

У цій роботі пропонується алгоритм побудови маршрутів, що підходить для будь-яких видів зв'язкових графів. На відміну від алгоритму GSPF, запропонований алгоритм знаходить оптимальне рішення за критерієм мінімізації штрафу за кількість перетинів. Функція штрафу в залежності від

кількості маршрутів, що проходять через те саме ребро графа мережі, змінюється, оскільки ситуація, коли всі маршрути мають перетин по одному ребру, з точки зору відмовостійкості, гірша за ситуацію, коли маршрути мають різні попарні перетину. Тому штраф для одного ребра, на якому є перетин, визначатиметься як $|V|^z$, де z – кількість маршрутів, що проходять через це ребро. Якщо в топології існують маршрути без перетинів, то мінімальний штраф дорівнюватиме нулю і результатом роботи алгоритму повинні бути маршрути без перетину. Далі ми також називатимемо маршрути з мінімальним штрафом маршрутами з мінімальним перетином, оскільки додаткові перетину збільшуватимуть функцію штрафу. У запропонованому алгоритмі завдання побудови маршрутів зводиться до пошуку максимального потоку мінімальної вартості (Min-Cost Max- Flow), тому далі називатимемо цей алгоритм MCMF.

У алгоритмі MCMF спочатку перетворять вихідний граф. Кожне ребро вихідного графа замінюється на k паралельних ребер, де k є вхідним параметром алгоритму. Для кожного вихідного ребра графа паралельні ребра, що вийшли, нумеруються довільним способом. Для кожної множини паралельних ребер вартість i-го ребра ($1 \leq i \leq k$) встановлюється рівною $|V|^{i-1}$, де під $|V|$ розуміється число ребер у вихідному графі. Це необхідно, щоб забезпечити якнайменшу кількість перетинів. Для графів із великою кількістю ребер можна зменшити вартість, якщо відомий діаметр графа. Якщо маршрути родинних підпотоків матимуть перетин у вихідному графі, то у перетвореному графі маршрути цих родинних підтоків будуть проходити через різні паралельні ребра. Для кожного нового паралельного ребра вартість зростає, оскільки більша кількість споріднених підтоків, маршрути яких проходять через те саме ребро, може призвести до більшої конкуренції за ресурси відповідної лінії зв'язку.

Для всіх ребер у перетвореному графі ємність встановлюється в 1. Нарешті, до перетвореного графа додається додаткова вершина – сток, що

з'єднується з вершиною призначення у графі ребром із ємністю k. Вартість ребра, інцидентного стоку, можна призначити довільним чином.

Фінальним кроком алгоритму MCMF є відображення маршрутів у перетвореному графі маршрути у вихідному графі. Хоча таке відображення може бути не єдиним, кожен набір маршрутів, що вийшов, матиме мінімальний перетин. Доказ цього твердження наведено нижче.

Для графа, зображеного на рисунку 1.7, перетворений граф відповідно до роботи алгоритму MCMF при k = 2 зображений на рисунку 1.9. У дужках для кожного ребра вказано його ємність та вартість відповідно/

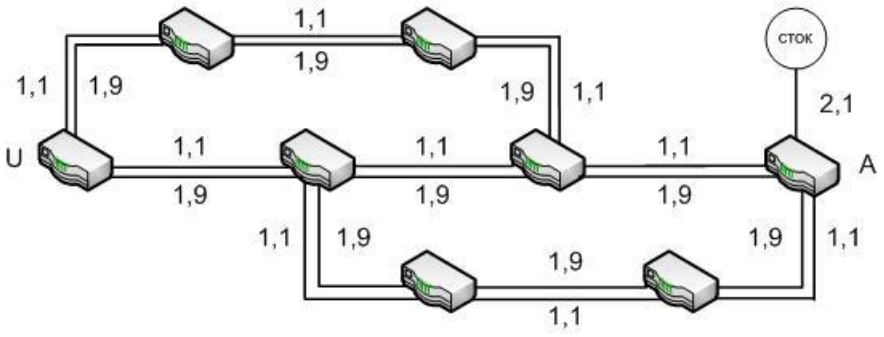

Рисунок 1.9 – Приклад перетвореного графа алгоритмі MCMF

За допомогою запропонованого алгоритму MCMF можна знайти маршрутів у графі мережі, що мають мінімальну кількість перетинів. Однак для застосування алгоритму MCMF спільно з динамічним методом багатопоточної маршрутизації необхідно додатково перевірити, що залишкова пропускна спроможність на маршрутах дозволить задовольнити вимоги до якості сервісу.

Окремим питанням є вибір k – яку кількість маршрутів необхідно будувати за допомогою алгоритму MCMF. Це питання безпосередньо залежить від кількох факторів. По-перше, від надмірності, яку може надати сама топологія. Під терміном ступінь демультиплексування графа мережі розумітимемо арифметичне середнє значення по всіх можливих парах

вершин графа максимальної кількості маршрутів, що не перетинаються, між парою вершин. На рисунку 1.10 показані результати дослідження топологій Інтернет-провайдера з бібліотеки Topology Zoo [14], з якого випливає, що ступінь демультиплексування для більшості топологій не перевищує трьох. Тому як k для поточних топологій Інтернет-провайдерів достатньо будувати по 3 маршрути. У проведеному дослідженні ступінь демультиплексування вважався за допомогою алгоритму MCMF, який шукає маршрути з мінімальним перетином. Максимальна кількість маршрутів без перетинів шукалася послідовним застосуванням алгоритму MCMF зі збільшенням k до першого перетину. При застосуванні аналогічного підходу з алгоритмом GSPF в 91 випадку з 259 ступінь демультиплексування виявлялася меншою, максимальна різниця могла становити 0,12.

По-друге, кількість маршрутів залежить від запитуваного додатком якості сервісу та залишкових ресурсів мережі. Діяльність пропонується будувати кількість маршрутів, що визначається ступенем демультиплексування, але видавати многопоточному з'єднанню таку кількість маршрутів, що дозволить задовольнити запитувані вимоги, тобто. кількість маршрутів, що видається, може виявитися меншою, ніж побудована безліч маршрутів. Якщо збудованого безлічі маршрутів не вистачає для забезпечення необхідної якості сервісу, то відмова в обслуговуванні багатопоточного з'єднання.

Також важливим питанням є встановлення правил ПКМ для реалізації побудованих маршрутів багатопотокового з'єднання. Існують дві стратегії роботи контролера ПКМ: реактивна та проактивна. У реактивній стратегії контролер будуватиме маршрут і встановлюватиме правила щодо настання деякої події, наприклад, появи першого пакета нового багатопоточного з'єднання в мережі. Для цього контролер ПКМ налаштовує підконтрольні йому комутатори для перехоплення всіх пакетів TCP рукостискань і відбирає пакети, які відносяться до багатопоточного з'єднання. У термінах протоколу OpenFlow, що розглядається в роботі як протокол взаємодії між контролером

ПКМ і підконтрольними комутаторами, контролер встановлює на комутатори низькопріоритетні правила для відправлення PacketIn повідомлень на контролер для кожного TCP пакета.

Рисунок 1.10 – Співвідношення кількості топологій з різним ступенем демультиплексування

При надходженні PacketIn повідомлення контролер аналізує опціональну частину транспортного заголовка та шукає опції, що стосуються багатопоточного протоколу MPTCP або FDMP. Якщо відповідні опції присутні, то контролер сприйматиме цей TCP потік як підпотік багатопоточного з'єднання.

Щоб визначати, до якого багатопоточного з'єднання відноситься підпоток, що розглядається, контролер зберігає інформацію про кожен підпоток в SIB (Subflow Information Base). При надходженні першого підтоку багатопотокового з'єднання в мережу контролер витягує ключ, що передається в опції MP_CAPABLE. Якщо пакет відповідає ініціації з'єднання, тобто. пакет з прапором SYN, то створюється новий запис в SIB і ми зберігаємо ключ відправника. Інакше запис вже створено у SIB і ми

зберігаємо ключ одержувача. На основі двох ключів контролер може обчислити хеш, який передаватиметься в наступних споріднених підтоках. Виявлення розрахованого хешу в пакетах інших підтоків означатиме належність невідомого до того моменту підтоку до багатопотокового з'єднання, інформацію про яке ми тільки зберегли в SIB.

За допомогою SIB контролер може визначати споріднені підтоки та прокладати для них маршрути з найменшою кількістю перетинів. Наступні пакети після процедури трикратного рукостискання повинні йти згідно з правилами, встановленими для відповідних маршрутів. Кожен підпотік визначається правилом OpenFlow за парою IP адрес та парою транспортних портів. Щоб підтримувати передачу пакетів в обидві сторони, контролеру необхідно встановити правил удвічі більше довжини маршруту.

Реактивна стратегія має такі недоліки:

- безліч мережевих додатків таких як засоби моніторингу, розподілені бази даних, ініціюють тисячі нових транспортних з'єднань в секунду. Кількість правил, згенерованих реактивною стратегією, займатиме практично всю пам'ять навіть флагманських комутаторів ПКМ, таких як NoviSwitch 2122, який підтримує до 1 мільйона шаблонних і до 6 мільйонів з точною відповідністю записів у своїй таблиці. Так як зазвичай використовуються дешевші комутатори, то для реактивної стратегії не вистачатиме записів у комутаторах ПКМ;

- окрім генерації надто великої кількості правил, реактивна стратегія генерує їх надто інтенсивно. Комутатори ПКМ зазвичай мають менш продуктивний процесор для роботи з контуром управління, ніж із контуром даних, який може в кращому випадку обробити тисячі оновлень таблиці за секунду. Контролери ПКМ мають великі обчислювальні можливості і можуть обробляти кілька мільйонів надходжень PacketIn в секунду [3]. Однак нетривіальна обробка пакетів багатопоточного з'єднання, можливо, зменшить це число на кілька порядків. Якщо швидкість, з якою програми відкривають

або закривають свої транспортні з'єднання, перевершує можливості апаратури, мережа не зможе обробляти їх.

Реактивна стратегія вносить значну затримку у процедуру триразового рукостискання. При отриманні першого пакета нового транспортного потоку комутатор ПКм генерує PacketIn повідомлення на контролер. Контролер аналізує PacketIn, що надійшов, розраховує маршрути і посилає FlowMod повідомлення для встановлення маршруту. Насправді сумарний час перерахованих дій перевищує 50 мілісекунд, що може відповідати тривалості деяких транспортних з'єднань і неприйнятно для додатків реального часу.

Хоча ці недоліки не роблять реактивну стратегію непрактичною, її не можна застосовувати в мережах Інтернет-провайдера безпосередньо.

На противагу реактивній стратегії проактивна стратегія передбачає попереднє завантаження правил пересилання пакетів на етапі ініціалізації контролера ПКМ. Отже, комутатори ПКМ мають обробляти пакети повністю самостійно, без втручання контролера ПКМ. Зокрема, комутатори ПКМ мають розподіляти споріднені підтоки різними маршрутами, що мають мінімальну кількість перетинів. Самі маршрути можуть бути побудовані заздалегідь і відповідні правила можуть бути встановлені заздалегідь, однак пакети MPTCP або FDMP не несуть жодних додаткових ознак, які можуть допомогти у визначенні приналежності тому чи іншому багатопоточному з'єднанню. Можна використовувати хеш функції для балансування за побудованими маршрутами, проте, як було зазначено раніше, відсутня гарантія того, що родинні підпотоки вирушать різними маршрутами.

Одним із можливих підходів для запобігання збігу маршрутів споріднених підтоків може стати відстеження багатопоточних з'єднань у комутаторах ПКМ так само, як це зроблено у реактивній стратегії в контролері ПКМ. Однак переважним буде підхід, коли транспортні агенти допомагають комутаторам визначати споріднені підпотоки. Наприклад, пакети споріднених підтоків можуть містити різне значення DSCP мітки в заголовку IP. Здійснювати порівняння за міткою DSCP повинні вміти всі

комутатори ПКМ, що підтримують роботу за протоколом OpenFlow версії не нижче 1.0, та реалізовано більшістю сучасних комутаторів ПКМ. Так як DSCP маркування є поширеною процедурою, її буде простіше реалізувати, ніж аналіз опцій транспортного заголовка. Нарешті, незважаючи на факт, що DSCP поле складається тільки з 6 біт, більшість із 64 значень ніколи не використовується на практиці – зазвичай підтримується не більше 14 різних класів трафіку, рекомендованих стандартом RFC [63]. Таким чином, в DSCP поле достатньо місця, щоб закодувати принаймні 3 різних підтоку для з'єднання будь-якого класу трафіку.

Більш радикальна, але готова до реалізації ідея, полягає у забезпеченні кожного хоста безліччю з k мережевих адрес, у яких закодовано номер спорідненого підтоку на деяких зафіксованих позиціях. Хоча ця ідея може бути непридатною для обмеженого простору адрес IPv4, вона стає здійсненною для IPv6 адрес, що дозволяють використовувати 64 біта для ідентифікації хоста. Сам хост може визначити, використовувати всі адреси одному мережному інтерфейсі чи розподілити їх між кількома мережевими інтерфейсами. Якщо транспортний агент багатопотокового протоколу налаштувати таким чином, щоб перший споріднений підпотік відправлявся від і-го адреси, то ідентифікація підтоків всередині мережі стає очевидною. У цьому випадку комутатори ПКМ повинні перенаправляти пакети і-го спорідненого підтоку на і-ий маршрут, прокладений заздалегідь.

Хотя проактивная стратегия устраняет, сглаживает, смягчает недостатки реактивной стратегии, она тоже обладает несколькими недостатками:

Для реалізації проактивної стратегії підтримки багатопоточної маршрутизації необхідні зміни або в комутаторах ПКМ, або на хостах, що ускладнює поширення багатопоточних протоколів.

Проактивна стратегія набагато гірше зможе підтримувати динамічний метод багатопоточної маршрутизації, оскільки на контролері ПКМ заздалегідь невідоме навантаження в мережі, а комутатор ПКМ не має

бачення стану всієї мережі, тому визначити, які маршрути треба використовувати для забезпечення необхідної якості сервісу.

Проактивна стратегія не вирішує проблему з кількістю правил для здійснення багатопоточної маршрутизації, яка була описана для реактивної стратегії.

В роботі пропонується використовувати змішану стратегію для реалізації багатопоточної маршрутизації у ПКМ. Контролер ПКМ заздалегідь будує маршрути з мінімальною кількістю перетинів між кожною парою точок у мережі. Кількість маршрутів визначається зі ступеня демультиплексування мережі. Побудувавши маршрути заздалегідь, ми можемо зменшити затримку на встановлення з'єднання, тому що не потрібно перераховувати їх щоразу при надходженні нового підтоку. Крім того, ми можемо заздалегідь встановити правила для кожного з маршрутів за винятком точки входу та точки виходу, тим самим скоротивши час, що витрачається на встановлення маршруту, залишивши встановлення лише двох правил. У динаміці контролер ПКМ має підтримувати стан цих маршрутів, де під станом розуміється залишкова пропускна спроможність маршруту. Контролер ПКМ може отримати стан маршруту, збираючи статистику про обсяг передається трафіку в одиницю часу (назвемо цю величину завантаженням лінії зв'язку) на кожній лінії зв'язку на маршруті і беручи мінімум з різниць пропускної спроможності та поточного завантаження. Тоді при появі нового багатопоточного з'єднання в мережі можна оцінити, чи вистачає ресурсів побудованих маршрутів для забезпечення потрібної якості сервісу для цього потоку класу чи ні. Якщо ресурсів маршрутів вистачає для задоволення вимог багатопоточного транспортного сполучення, що розглядається, то за цим транспортним з'єднанням резервуються необхідні ресурси.

Для того, щоб контролер ПКМ дізнався про вимоги багатопоточного транспортного сполучення, можна використовувати кілька підходів. Наприклад, можна зафіксувати класи додатків, де кожного класу будуть

запитуватися однакові вимоги до якості сервісу. Тоді клас програми можна закодувати в DSCP мітці, яку може розпізнати контролер ПКМ. Другим підходом може бути передача вимог додатка в опціональній частині транспортного заголовка, тоді контролер ПКМ зможе дізнатися про вимогу, оскільки все одно аналізує опціональну частину транспортного заголовка.

Ресурси, що займаються багатопоточним транспортним з'єднанням, повинні звільнятися після завершення з'єднання. Понад те, ресурси, займані одним підпотоком, мають звільнятися під час завершення цього підпотоку, оскільки динамічний метод многопоточной маршрутизації дозволяє закривати зайві підпотоки. Контролер ПКМ може дізнатися про завершення підпотоку за допомогою idle-timeout – таймера, який прив'язаний до певного правила в таблиці OpenFlow і веде свій відлік за відсутності пакетів, які відповідають цьому правилу. Якщо з'являється новий пакет мережі, який відповідає цьому правилу, то відлік починається спочатку. Після закінчення відведеного часу в idle-timeout комутатор ПКМ надішле службове повідомлення на контролер ПКМ із зазначенням, за яким правилом спрацював idle-timeout. Використовуючи цю інформацію, контролер ПКМ зможе зрозуміти, який із підтоків завершив свою роботу.

Так як для кожного багатопотокового транспортного з'єднання відбувається резервація ресурсів і за умови відсутності інших потоків в ПКМ (або під ці потоки виділено окремий резерв пропускної спроможності, який відомий заздалегідь), можна використовувати спрощену реалізацію з наступною моделлю. Зберігатимемо поточний стан кожної лінії зв'язку. У разі появи нового з'єднання MPTCP визначаємо резервовану пропускну спроможність для кожного маршруту споріднених підтоків цього з'єднання за допомогою алгоритму поступового заповнення [7], так як всі підпотоки MPTCP відкриваються відразу, маршрутах може бути нерівномірна залишкова пропускна здатність, і можуть бути перетину між маршрутами. У разі з'єднання FDMP визначаємо резерв пропускної спроможності, послідовно розглядаючи кожен маршрут і забираючи всю залишкову

пропускну спроможність маршруту, поки не наберемо необхідну для виконання вимоги потоку, що розглядається. Запропонована модель дозволяє не витрачати ресурси контуру управління на моніторинг завантаження ліній зв'язку.

Зараз розглянемо вирішення проблеми, пов'язаної з нестачею пам'яті комутаторів для правил, що здійснюють багатопоточну маршрутизацію. Для зменшення кількості правил не можна використовувати маршрутизацію на адресу точки призначення, оскільки маршрути з найменшою кількістю перетинів не є найкоротшими маршрутами, тому остання частина маршрутів, побудованих з різних точок відправлення до однієї точки призначення, може не збігатися. Тому багатопоточна маршрутизація навіть для одного підпотоку потребує більшої кількості правил, ніж для однопоточного з'єднання TCP.

Щоб зменшити кількість правил, необхідні багатопоточної маршрутизації, пропонується поділити безліч всіх маршрутів на групи. Будемо вважати, що всередині однієї групи не відбувається колізій - ситуації, коли два маршрути в групі проходять через одну і ту ж вершину v і частина маршрутів, що залишилася, від v до точки призначення не збігається. Тоді в рамках однієї групи можна проводити маршрутизацію на адресу точки призначення. Навіть якщо група складається лише з одного маршруту, то можна отримати перевагу, так як різні підпотоки, що проходять по тому самому маршруту, будуть оброблятися за одним і тим же правилом, а не за окремим набором. Ціною використання груп буде додаткове поле в заголовку пакета, щоб відрізняти пакети, які повинні йти маршрутами з різних груп. Хоча для цієї мети можна використовувати різні додаткові мітки (VLAN, MPLS тощо), пропонується розрізняти групи за допомогою MAC-адрес. Оскільки Інтернет-провайдер зазвичай забезпечує лише L3-зв'язок, то контролери ПКМ перезаписують MAC-адреси на граничних комутаторах ПКМ, щоб імітувати поведінку традиційних маршрутизаторів та діяти в рамках їхніх очікувань. У разі використання MAC-адрес ми не будемо вносити додаткову службову інформацію в заголовок пакета, тому цей метод

позбавлений додаткових витрат. Комутатор ПКМ на вході пакета в мережу повинен перезаписати MAC-адреси міткою, яка позначатиме номер групи. Комутатор ПКМ на виході пакета з мережі повинен замінити MAC-адресу джерела на свою власну та MAC-адресу призначення на адресу наступного маршрутизатора.

Можна оцінити, що максимальна кількість груп маршрутів не перевищуватиме $k * (n - 1)$, де k – кількість споріднених підтоків, а n – число комутаторів у мережі. Ця оцінка пов'язана з тим, що для однієї точки призначення можуть існувати по k маршрутів для кожного з (n – 1) комутаторів, що залишилися. Маршрути до різних точок призначення не впливають один на одного, тому їх можна поєднувати в одній групі. Однак сумарна кількість груп практично залежить від способу об'єднання маршрутів у групи.

Для розбиття маршрутів на групи, коли їх кількість буде мінімально, у роботі пропонується звести це завдання розбиття до завдання розмальовки графа. Розглянемо граф G з $k * (n - 1)$ вершиною, де кожна вершина відповідає одному з маршрутів до фіксованої точки призначення. Для кожної пари маршрутів, що мають колізію, з'єднаємо вершини руба. Тоді мінімальна кількість груп відповідатиме мінімальній кількості кольорів, необхідних щоб пофарбувати граф G таким чином, щоб сусідні вершини не мали одного кольору. Завдання знаходження розмальовки графа - добре відоме NP-повне завдання, і існує безліч підходів, щоб знайти оптимальне або наближене рішення. Слід зазначити, що складність розв'язання представленої завдання має нашкодити продуктивності мережі, оскільки це завдання має вирішуватися на етапі ініціалізації мережі.

У цій роботі було проведено оцінку кількості груп маршрутів для топологій з бібліотеки Topology Zoo. Результат оцінки наведено на рисунку 1.11, де зазначено залежність між числом груп і часткою топологій, що покриваються ними. Товста лінія відповідає групам маршрутів, побудованих відповідно до алгоритму MCMF, в той час як тонка лінія

відповідає групам, побудованим з алгоритмом GSPF. На рисунку наведено результати як для двох підтоків, так і для трьох підтоків. Для жодної з досліджених топологій не знадобилося понад 41 група, тоді як 80% випадків вимагають менш як 10 груп.

Хоча теоретична оцінка кількості необхідних груп досить велика, вимоги реальних мереж набагато нижчі від отриманої теоретичної оцінки. Оскільки кількість груп вбирається у бітову довжину MAC-адреси, ми можемо виділити окремий біт-індикатор кожної групи, тобто. для і-ої групи на всіх крім і-ої позиції біти встановлюються в нуль. Такий підхід дозволяє нам поєднувати однакові правила відправлення незалежно від номерів груп, які там задіяні, за допомогою бітових масок.

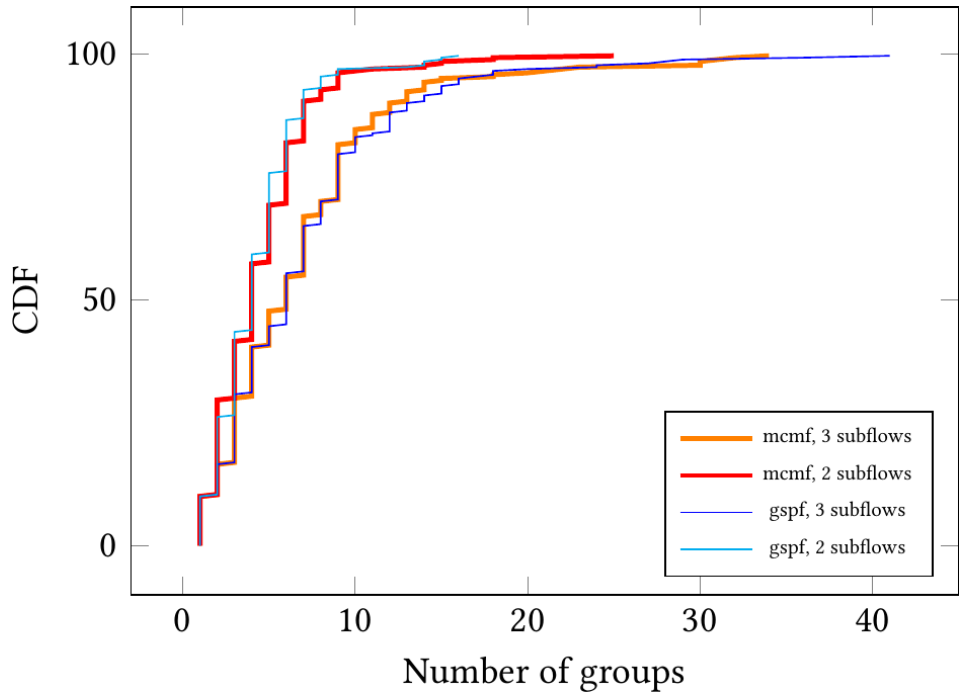

Рисунок 1.11 – Число груп, достатнє для обробки багатопотокових сполук між кожною парою вузлів для різних топологій

Також у роботі була проведена оцінка кількості правил OpenFlow, які виходять у разі реактивного підходу та у разі використання груп. У кожному експерименті передбачалося, що з кожним вузлом топології перебуває рівно

одна IP підмережа і всі IP підмережі можуть спілкуватися між собою. Хоча деякі префікси підмереж можуть бути об'єднані, підхід CIDR у дослідженні не використовувався, оскільки результат залежить від схеми адресації. Таким чином, запропонований підхід показує мінімальну кількість правил, необхідних реалізації багатопоточної маршрутизації з довільною схемою адресації.

Рисунок 1.12 показує кількість OpenFlow правил, необхідних реалізації багатопоточної маршрутизації реактивним підходом (наївний підхід) і запропонованим підходом з групами, складеними з маршрутів, побудованих алгоритмом MCMF. На рисунку 1.13 представлена оцінка мінімального розміру таблиці потоків протоколу OpenFlow, необхідна для встановлення правил з рисунка 1.12. Оскільки результати для GSPF алгоритму практично ідентичні з результатами алгоритму MCMF, вони були опущені.

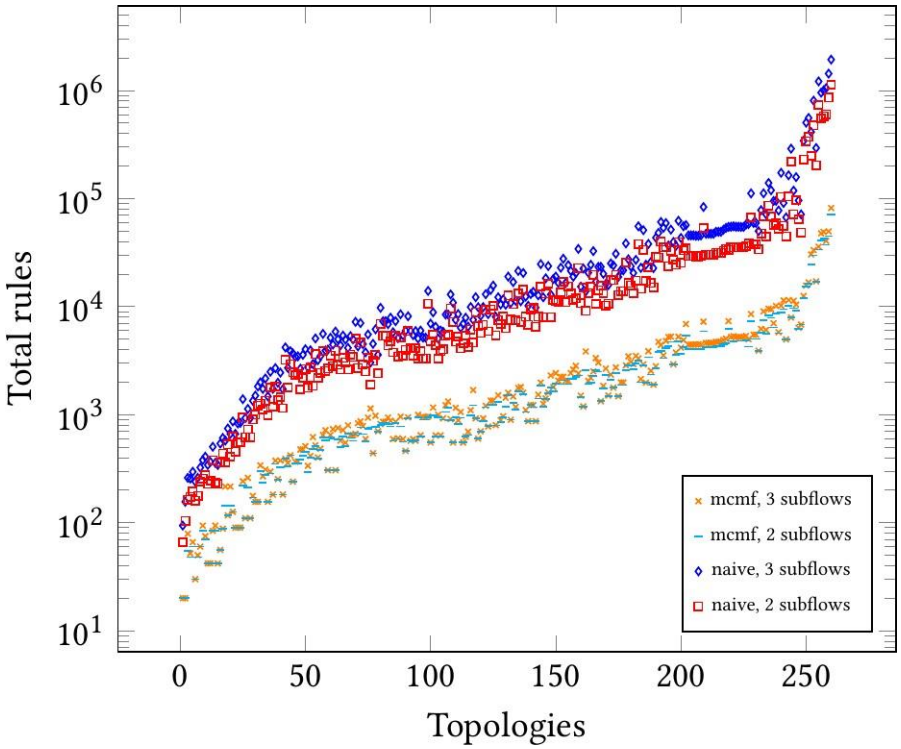

Рисунок 1.12 – Кількість OpenFlow правил, достатня для обробки багатопоточних з'єднань між кожною парою вузлів для різних топологій

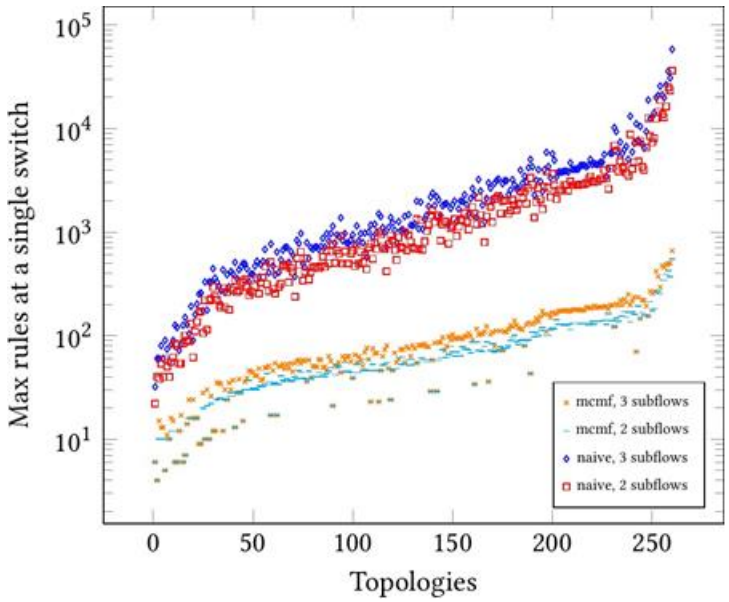

Рисунок 1.13 – Розмір таблиці потоків OpenFlow, достатній для обробки багатопоточних з'єднань між кожною парою вузлів для різних топологій

Згідно з отриманими результатами, наївний підхід до побудови правил вимагає значно більше правил, ніж запропонований підхід, заснований на групах маршрутів. Більш того, перевага запропонованого підходу значно зростатиме, якщо збільшуватиметься кількість потокових транспортних з'єднань між вузлами мережі. Як наслідок, наївний підхід вимагає великого розміру таблиць потоків OpenFlow, а запропонований підхід дозволить обійтися дешевшими комутаторами ПКМ.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ ЕФЕКТИВНОСТІ СТАТИЧНОГО ТА ДИНАМІЧНОГО МЕТОДІВ БАГАТОПОТОЧНОЇ МАРШРУТИЗАЦІЇ

У попередньому розділі були описані статичний та динамічний методи багатопотокової маршрутизації. Природне питання для наукового дослідження, який із цих методів краще застосовувати, та у яких випадках. Припускатимемо, що вимоги до швидкості передачі даних для всіх транспортних потоків у мережі Інтернет-провайдера відомі. Тоді неформально завдання вибору методу багатопоточної маршрутизації можна сформулювати наступним чином: вибрати такий метод багатопоточної маршрутизації, який дозволить задовольнити вимоги якомога більшої кількості транспортних потоків в мережі Інтернет-провайдера за один і той же інтервал часу.

2.1 Методика проведення порівняння статичного та динамічного методів багатопоточної маршрутизації

Як було зазначено вище, щодо порівняння було використано метод перевірки статистичних гіпотез, заснований на оцінці влучення випадкової величини – кількість потоків, вимоги яких за швидкістю задоволені, в заданий інтервал. Теорія статистичної перевірки гіпотез дозволяє для заданого довірчого інтервалу гіпотези оцінити кількість експериментів, які необхідно виконати, щоб забезпечити необхідний рівень достовірності гіпотези.

2.1.1 Опис вхідних та вихідних параметрів

Основними поняттями, що використовуються для визначення критеріїв порівняння та опису експериментів, описані нижче. Порівняльний аналіз

складався з проведення серії експериментів, кожен із яких характеризувався п'ятіркою

$$< N, F, T, M, sub_{max} >,$$

де: $N=< G, p >$ – транспортна мережа з топологією, представленою орієнтованим графом $G = (V, E)$, на якому задана функція розмітки $p: e \rightarrow (c(e), d(e), loss(e))$, де $e \in E, c(e) \in R^+, d(e) \in R^+$. Вершини графа – комутатори/маршрутизатори мережі, а дуги – лінії. ρ характеризує якість сервісу зв'язку ліній: пропускну здатність $c(e)$ лінії, що відповідає дузі e, затримку передачі $d(e)$ на цій лінії та ймовірність втрати пакету $loss(e)$.;

$F = \{f_i\}$ – безліч потоків даних, що передаються по мережі N, кожен з яких представлений четвіркою $< ip_i^{src}, ip_i^{dst}, tp_i^{src}, tp_i^{src} >$, де $ip_i^{src}, ip_i^{dst}, tp_i^{src}, tp_i^{src}$ – IP адреси та транспортні порти відправника та одержувача. Зрозуміло, що таке подання унікальне для кожного потоку в мережі – це фактично адреса на транспортному рівні відправника та одержувача;

$T = [0, t_{max}]$ – дискретний час спостереження за мережею з кроком δ;

$M =< \phi, \psi, \tau >$ – представлення властивостей потоків, де:

$\phi: f_i \rightarrow (v_i^s, v_i^t)$ – розподіл полюсів для кожного потоку, де під полюсом розуміється стік чи витік потоку. Кожен потік має свій виток і сток, з'єднані з мережею в точках $v_i^s \in V$ та $v_i^t \in V$, відповідно. Припускаємо, що лінії зв'язку між витоком і мережею, а також між мережею і стоком мають достатню пропускну здатність для будь-якого потоку і нульову затримку;

$\psi: f_i \rightarrow (y_i, u_i)$ – вимога якості потоків – Service Level Agreement (SLA) потоку. SLA потоку. SLA задає обсяг даних $y_i \in \mathbb{R}^+$, який необхідно

передати за час $u_i \in \mathbb{R}^+$. Аналогічно можна задати вимогу до швидкості потоку $q_i = \frac{y_i}{u_i}$;

$\tau: f_i \to t_i$ – розклад старту потоків, де $t_i \in T, t_i + u_i < t_{max}$;

sub_{max} – максимальна кількість споріднених підпотоків. Нагадаємо, що кожен потік передачі даних в наших експериментах складається з декількох підтоків, які називаються спорідненими. У разі протоколу MPTCP кожен потік fi розбитий рівно на $submax$ споріднених підтоків, а у разі протоколу FDMP кількість задіяних у передачі даних споріднених підтоків може змінюватись, і може бути як менше, так і $submax$.

Результатом кожного експерименту є наступне відображення:

$$\theta: f_i \to \left\{L_{ij}\left\{r_{ij}\right\}_t\right\}, t \in T,$$

де: $L_{ij} = (e_{ij}^1, e_{ij}^2, \dots, e_{ij}^n)$ – маршрут підпотоку j потоку f_i, представлений безліччю впорядкованих дуг, $e_{ij}^k \in E$, k – порядковий номер дуги у маршруті.

$\{r_{ij}\}_t$ – тимчасовий ряд, елементи $r_{ij}(t)$, якого становлять середню швидкість спорідненого підпотоку j потоку f_i на інтервалі $[t-\delta, t], t \in T, t \le t_{max}$.

2.1.2 Критерії порівняння

Ефективність статичного та динамічного методів будемо оцінювати за наступним критерієм – кількість потоків, вимоги SLA яких було задоволено. Введемо такі позначення.

Середньою швидкістю потоку f_i на інтервалі $[t-\delta, t]$ назвемо величину $r_i = \sum_j r_{ij}(t)$. Середньою швидкістю потоку f_i в експерименті або

просто середньою швидкістю потоку f_i називатимемо величину $r_i = \frac{\sum_T r_i(t)}{|\{t_k \in T : t_i \le t_k \le t_i + u_i\}|}$. Таким чином, середня швидкість потоку – це середнє арифметичне середніх швидкостей потоку на інтервалах часу, починаючи з моменту старту потоку і до його закінчення, тобто $t_i \le t_k \le t_i + u_i$. Вертикальними лініями $\|$ тут і далі позначаємо кількість елементів для множини, укладеної між лініями. Наприклад, у формулі середньої швидкості потоку $|\{t_k \in T : t_i \le t_k \le t_i + u_i\}|$ – позначає кількість дискретних моментів часу, укладених між часом старту та часом закінчення потоку.

Вартістю підпотоку j потоку f_i на інтервалі $[t - \delta, t]$ назвемо величину $c_{ij}(t) = r_{ij}(t) * |L_{ij}|$. Величина $c_{ij}(t)$ показує, яку частину пропускної спроможності ліній зв'язку займає підпотік j потоку f_i на розглянутому інтервалі. Чим більша довжина маршруту, тим більше ресурсів мережі витрачається на передачу цього підпотоку. Вартістю потоку f_i на интервале $[t - \delta, t]$ буде $c_i(t) = \sum_j c_{ij}(t)$. Вартістю потоку f_i на інтервалі $[t - \delta, t]$ буде $c_i(t) = \sum_j c_{ij}(t)$. Середньою вартістю потоку f_i в експерименті або просто середньою вартістю потоку f_i називатимемо величину $c_i = \sum_T \frac{c_i(t)}{|\{t_k \in T : c_i(t_k) > 0\}|}$.

У термінах введених позначень кількість потоків, котрим вимога якості сервісу виконано, можна описати так:

$$sf = |\{f_i : r_i \ge q_i\}|.$$

Також визначимо вартість потоків, SLA яких було задоволено:

$$c = \sum_{i : r_i \ge q_i} c_i .$$

Використання цього параметра, що визначається в результаті

експерименту, дозволяє розрізняти випадки, коли методи, що порівнюються, дають однакові значення за першим критерієм – кількість потоків з виконаним SLA, але вимагають різних витрат ресурсів. Наприклад, розглянемо топологію мережі «трикутник», зображену рисунку 1.3 на сторінці 11. Якщо як вимоги до швидкості багатопоточного з'єднання вибрати $\frac{2}{3}$, то і статичний метод, і динамічний метод багатопоточної маршрутизації дозволять задовольнити всі потоки. Однак у разі статичного методу знадобиться два споріднених підпотоки з вартістю $\frac{1}{3} * 1 + \frac{1}{3} * 2 = 1$ (маршрути споріднених підпотоків матимуть довжину 1 і 2 відповідно), а у разі динамічного методу – лише один підпотік, тому що одного маршруту буде достатньо для задоволення необхідної якості сервісу, а його вартість $\frac{2}{3} * 1 = \frac{2}{3}$. Загальна вартість використання статичного та динамічного методів дорівнюватиме 6 і 4 відповідно. У разі статичного методу буде зарезервовано всю пропускну спроможність мережі, а динамічний метод дозволить обслужити додаткові транспортні потоки.

2.2 Опис Стенду

Для цілей проведення імітаційних експериментів було розроблено та побудовано Стенд, що відповідає наступним вимогам, складеним на основі опису вхідних та вихідних параметрів експерименту, представлених у розділі 2.1.1:

- будувати імітаційну модель мережі N із заданими параметрами якості сервісу ліній зв'язку ρ;
- генерувати потоки даних F згідно з заданим розкладом τ із заданими параметрами SLA ψ ;
- направляти потоки даних до імітаційної моделі мережі згідно з заданим розподілом полюсів ϕ;

- керувати передачею потоків даних F за допомогою емуляторів багатопотокових транспортних агентів протоколів MPTCP і FDMP з параметром $submax$;

- будувати маршрути $\{L_{ij}\}$ з найменшим перетином для споріднених підпотоків;

- збирати дані про швидкість споріднених підпотоків $\{r_{ij}\}_t$ для дискретних моментів часу T.

Структуру розробленого Стенду показано на рисунку 2.1. Стенд складається з програмної і апаратної частини. Архітектура програмного забезпечення представлена на рисунку 2.2.

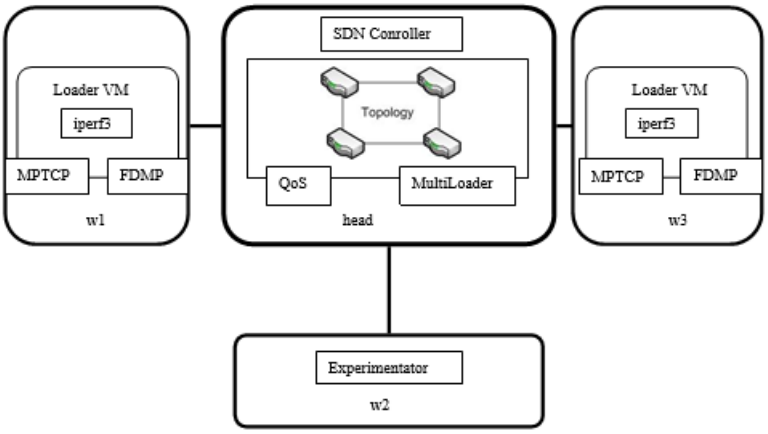

Рисунок 2.1 – Структура стенду

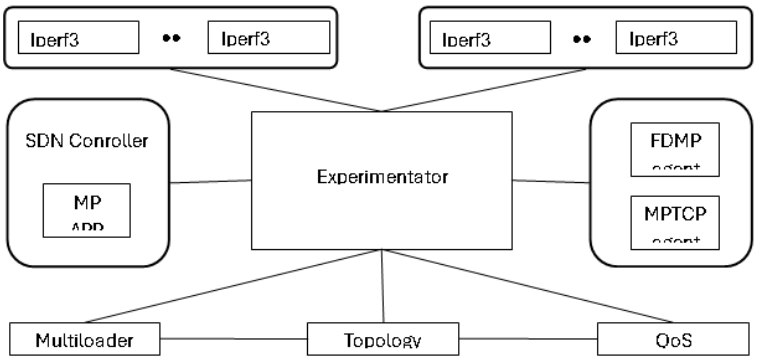

Рисунок 2.2 – Архітектура програмної частини експериментального стенду

Нижче описано призначення, основні функції та деталі реалізації кожної підсистеми Стенду, представленої на рисунку 2.2.

Topology – підсистема, що відповідає за імітаційне моделювання мережі N. Підсистема Topology надає інтерфейс для інших підсистем, який дозволяє розгорнути/очистити мережу, задану параметром N, а також отримати інформацію про модель розгорнутої в даний момент мережі, необхідну для її налаштування (наприклад, для налаштування якості сервісу ліній зв'язку).

QoS – підсистема, що відповідає за налаштування якості сервісу ліній зв'язку імітаційної моделі мережі N. Надає інтерфейс із відповідними параметрами іншим підсистемам.

Multiloader – підсистема, що відповідає за налаштування перенаправлення потоків даних від генераторів трафіку в імітаційну модель мережі та назад відповідно до заданого розподілу полюсів ϕ. Надає інтерфейс із відповідними параметрами іншим підсистемам.

Iperf3 – підсистема, що відповідає за генерацію навантаження потоків даних F згідно з заданим розкладом τ із заданими параметрами SLA ψ. Надає інтерфейс із відповідними параметрами іншим підсистемам.

FDMP агент – підсистема, що відповідає за передачу потоків даних згідно з протоколом FDMP, описаним у розділі 1.3. Підсистема надає інтерфейс для зміни максимальної кількості споріднених підтоків sub_{max}.

MPTCP агент – підсистема, що відповідає за передачу потоків даних згідно з протоколом MPTCP, описаним у розділі 1.2. Підсистема надає інтерфейс для зміни кількості споріднених підтоків sub_{max}.

MP App – підсистема, що відповідає за побудову маршрутів $\{L_{ij}\}$ споріднених підтоків відповідно до змішаної стратегії, описаної в розділі 1.4, та встановлення відповідних правил OpenFlow для цих маршрутів.

Experimentator – підсистема, що відповідає за запуск експериментів та збирання результатів. Інтерфейс запуску експерименту приймає як вхідні

параметри всю п'ятірку $< N, F, T, M, sub_{max} >$ і задіє всі інші підсистеми для проведення експерименту. При цьому проводиться опитування статистики правил OpenFlow ПКМ-комутаторів імітаційної моделі мережі та опитування статистики підсистеми Iperf3 у моменти часу, визначені параметром T.

Розроблений стенд, деталі реалізації якого описані у додатку Б, відповідає всім шістьом вимогам, перерахованим на початку розділу. Використання різних серверів для різних підсистем дозволить виключити гонку за процесорний час для завдань генерації потоків даних та імітаційного моделювання мережі Інтернет-провайдера.

2.3 Область допустимих значень параметрів експерименту

Перед стартом серії експериментів відбувається генерація значень параметрів $< N, F, T, M, sub_{max} >$ для кожного експерименту. Вважаємо, що допустимі значення параметрів мають рівномірний розподіл. Область допустимих значень параметрів була такою:

- топологія (G). Як топологія використовується одна з бібліотеки Topology Zoo;

- кількість потоків (F). Відповідно до вимог російського Інтернет-провайдера Ростелеком, кількість потоків в мережі може досягати 64000. Тому кількість потоків у експериментах змінювалася від 16000 до 64000;

- розподіл полюсів (ϕ). Кількість полюсів, для яких проводилося дослідження, змінювалося від 10% до 100% кількості вершин в мережі з кроком 10. Наприклад, 100% означає, що у кожної вершини в топології є свій полюс. Потоки рівномірно розподіляються між полюсами. Розташування полюсів у топології генерується випадковим чином і щоразу по-різному;

- кількість підпотоків (F). Основні значення, котрим проводилося дослідження: 2, 3. Використовувалися дані значення, оскільки ступінь демультиплексування більшість топологій Інтернет-провайдерів вбирається у 3, як було показано у розділі 1.4;

- розклад старту потоків (τ). Згідно з дослідженням [9] час надходження пакетів в автономну систему Інтернет-провайдера найкраще описується за допомогою розподілу Вейбулла. Тому в експериментальному дослідженні час старту потоку генерується за допомогою розподілу Вейбулла від 0 до деякої константи, яка повинна бути меншою за час закінчення експерименту.

- Класи потоків (ψ). Для всіх потоків однієї пари віртуальних машин можна встановити клас потоку. Кожен клас визначає необхідну якість сервісу: обсяг даних та час, за який ці дані мають бути передані. Виходячи з параметрів класу потоку, визначається вимога до пропускної спроможності транспортного сполучення. Класи потоків вибираються на основі класів відео (4K, 1440p, 1080p, 720p, 480p, 240p), представлених у джерелі [4]. За основу взято відео трафік, оскільки згідно з прогнозом Cisco [2] обсяг відеотрафіку значно зросте в найближчому майбутньому. Вимога часу передачі одного потоку даних вбирається у однієї хвилини;

- параметри середовища передачі (ρ). Пропускна здатність каналів, за якої проводилося дослідження – 1Гбіт/с. Обмеження 1 Гбіт/с пов'язане з обмеженнями експериментального стенду, де пропускна здатність між серверами не може перевищувати 40 Гбіт/с. Щоб вузьке місце утворювалося над лінії зв'язку між серверами, а віртуальної мережі, було обрано обмеження 1 Гбіт/с. Додатково була проведена серія експериментів, де пропускна здатність каналів збільшувалася разом з вимогами додатків (максимально в 10 разів), щоб перевірити застосовність отриманих результатів у мережах з більш швидкими каналами передачі даних;

- час закінчення експерименту (T). 3 хвилини. Так як час закінчення потоку не перевищує хвилини, то за 3 хвилини кілька разів зможуть змінитись транспортні потоки, що проходять через імітаційну модель мережі, що дозволить краще дослідити можливості динамічного методу багатопоточної маршрутизації.

- при старті експерименту підсистема Experimentator дає вказівку підсистемі Topology на побудову імітаційної моделі мережі, підсистемі QoS на налаштування параметрів якості ліній зв'язку, а підсистемі Multiloaer створення граничного комутатора і додаткових ліній зв'язку відповідно до заданого розподілу полюсів. Запускається контролер ПКМ Runos разом із додатком MP App, яке, згідно з змішаною стратегією, встановлювало правила OpenFlow, що реалізують передачу за побудованими згідно з алгоритмом MCMF або GSPF маршрутами в топології між полюсами. Кількість правил скорочено за допомогою угруповання маршрутів, описаного в розділі 1.1.3. Підсистема Experimentator запускає генератори трафіку iperf3 згідно з обраною кількістю потоків, розкладом потоків та вимогою SLA потоків. Статистика збирається від iperf3 по кожному потоку та від граничного комутатора по кожному підпотоку щомиті. Після закінчення експерименту стенд готується до наступного експерименту.

2.4 Результати експериментального порівняння

Для об'єктивного вибору оцінки кількості експериментів, які необхідно було провести, щоб можна було б зробити науково обґрунтований висновок порівняльного аналізу, було застосовано метод перевірки статистичних гіпотез про числове значення ймовірності події, описаний у додатку А.

Спочатку перевірялася гіпотеза перевагу динамічного методу многопоточной маршрутизації над статичним методом. Висувалася припущення, що динамічний метод не гірший за статичний з ймовірністю не менше 95%. Для перевірки цієї гіпотези згідно з методом, описаним у додатку А, та значенням довірчого інтервалу $p_0 = 0.95$, потрібно не менше 422 експериментів. Усього було проведено 1790 експериментів, що достатньо для обґрунтованого ухвалення рішення про здійсненність гіпотези. З 1790 проведених експериментів 1729 (96,59%) експериментів показали кількість задоволених потоків при динамічному методі FDMP більше або дорівнює

кількості задоволених потоків при статичному методі. Для прийняття висунутої гіпотези з рівнем значущості $\alpha = 0.01$ необхідно, щоб значення критерію $\varphi = \frac{(\hat{p}-p_0)}{\sqrt{\frac{p_0*(1-p_0)}{n}}}$ із додатку А було більше $x^{\text{кр}}_{\text{лев},\alpha} = -2.34$

Для проведеного експериментального дослідження $\varphi \approx 3.4$, тому гіпотеза приймається.

Оскільки кількість експериментів досить велика, то докладні результати кожного експерименту про швидкість потоків і займаних маршрутів зайняли б занадто багато місця у тексті. Тому далі будуть наведені лише агреговані результати з усього експериментального дослідження. Однак, Стенд дозволяє оцінити ефективність методів багатопоточної маршрутизації і для окремих мереж Інтернет-провайдерів.

З 1729 випадків, де динамічний метод був не гірший статичного, у 289 випадках (16,7%) досягалася рівність за кількістю успіхів (рисунок 18). Причому вартість потоків, визначена в пункті 2.1.2, у 102 випадках менша у динамічного методу, у 88 випадках менша у статичного методу, а в інших випадках однакова. Варто зазначити, що 80 випадків із 289 рівності кількості успіхів припадають на топології з коефіцієнтом демультиплексування 1, тобто. з повною відсутністю альтернативних маршрутів. Більшість інших випадків рівності припадає на топології з невеликим ступенем демультиплексування та невеликою часткою полюсів. Випадковий розподіл полюсів у топології може призвести до ситуації, коли відсутні альтернативні маршрути для топологій зі ступенем демультиплексування меншим 2

Аналіз експериментів, в яких статичний метод виявився кращим, показав, що основна область значень вхідних параметрів, при яких статичний метод має перевагу: топології з кількістю вузлів менше 37 і часткою полюсів, менше або дорівнює 40% від кількості вузлів у топології. випадків динамічний метод працюватиме не гірше, а в більшості випадків краще статичного методу багатопоточної маршрутизації.

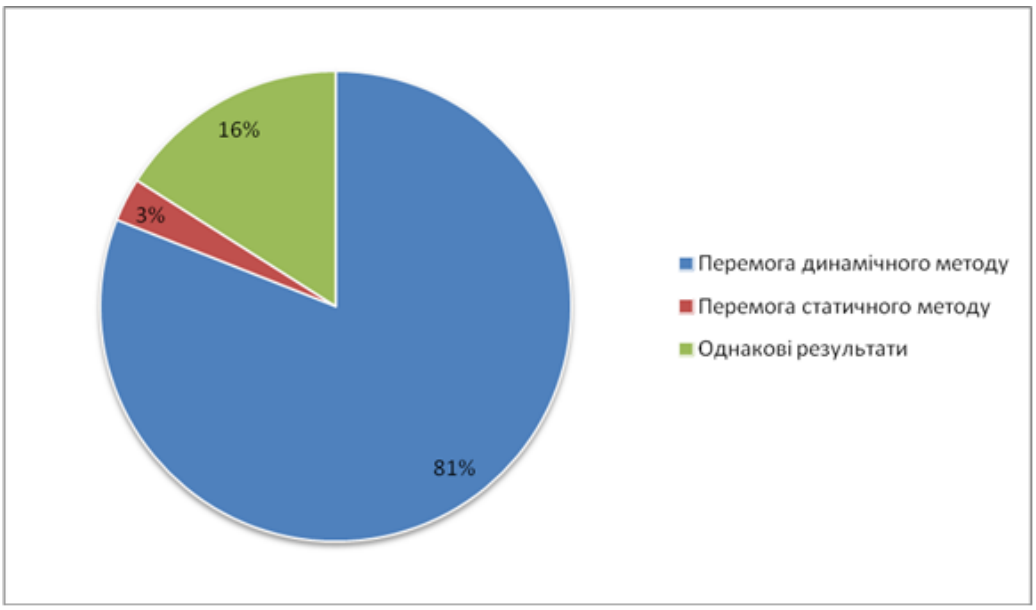

Рисунок 2.3 – Порівняння ефективності статичного та динамічного методів багатопоточної маршрутизації

Можна зробити припущення, що динамічний метод більше обслуговує потоки з класу трафіку з невеликою вимогою до якості сервісу, проте аналіз результатів показав, що для кожного класу трафіку не менше ніж 69% від кількості експериментів спостерігається приріст у числі задоволених потоків при використанні динамічного методу (рисунок 2.4). Виникає питання, наскільки наведені результати будуть актуальними у майбутньому, коли зростуть пропускні здібності каналів та вимоги додатків.

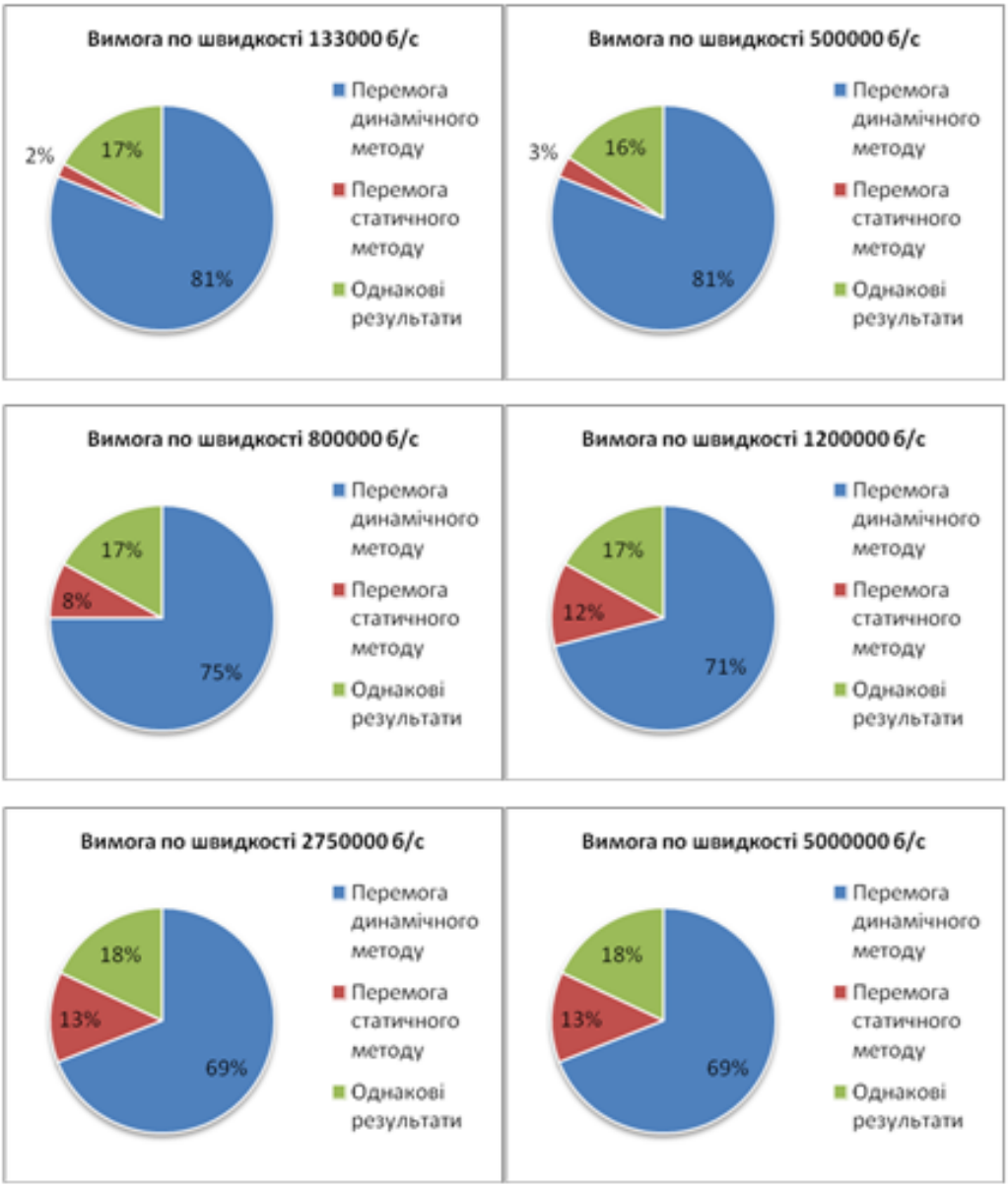

Рисунок 2.4 – Порівняння ефективності методів багатопоточної маршрутизації для різних вимог щодо швидкості

Додаткове експериментальне дослідження, в якому пропускні здібності каналів і вимоги додатків були збільшені в однакове число разів, показало, що ефективність динамічного методу багатопоточної маршрутизації залишається такою ж.

ВИСНОВКИ

Існуючі роботи з аналізу ефективності багатопоточної маршрутизації переважно аналізують лише ефективність статичного методу, а аналіз динамічного методу не проводився для мереж із фіксованим зв'язком Інтернет-провайдерів. Для проведення порівняльного аналізу обрані протоколи MPTCP та FDMP як представники статичного та динамічного методів. Хоча вибрані протоколи надають TCP інтерфейс для рівня програми, запропонована методика та результати порівняльного аналізу статичного та динамічного методів можуть стати в нагоді і для нових протоколів, заснованих на UDP протоколі, наприклад, QUIC. Так, для QUIC вже розроблена модифікація MP-QUIC, що підтримує багатопоточну маршрутизацію аналогічно MPTCP, тобто. зі статичним способом.

Для реалізації багатопоточної маршрутизації в ПКМ пропонується використовувати комбінацію реактивної та проактивної стратегій, а маршрути споріднених підтоків будувати згідно із запропонованим алгоритмом MCMF.

Перелічені підходи були використані в експериментальному стенді для порівняльного аналізу. Порівняльний аналіз статичного та динамічного методів багатопоточної маршрутизації показав перевагу динамічного методу, коли кількість задоволених потоків не менша, ніж для статичного методу, з ймовірністю не менше 95%, при цьому статичний метод добре працює тільки на невеликих топологіях з кількістю вузлів менше 36 і невеликою часткою полюсів

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дяченко В. О., Колісник Є. Б., Ляшова А. О., Можаєв О. О. Методи маршрутизації для забезпечення якості мережного сервісу//Системи управління, навігації та зв'язку.Т3.2024, стр. 67-71

2. Locations of Data Centers. [HTML] (https://www.datacenters.com/locations).

3. Salah T. et al. The evolution of distributed systems towards microservices architecture //2016 11th International Conference for Internet Technology and Secured Transactions (ICITST). – IEEE, 2016. – С. 318-325.

4. Zhang L. et al. Online electricity cost saving algorithms for co-location data centers //IEEE Journal on Selected Areas in Communications. – 2015. – Т. 33. – №. 12.– С. 2906-2919.

5. Telecommunications market research that's data-driven, TeleGeography. [HTML] (https://www.telegeography.com/).

6. Sinha S., Kandula S., Katabi D. Harnessing TCP's burstiness with flowlet switching //Proc. 3rd ACM Workshop on Hot Topics in Networks (Hotnets-III). – 2004.

7. Irawati I. D., Hadiyoso S., Hariyani Y. S. Link aggregation control protocol on software defined network //International Journal of Electrical and Computer Engineering. – 2017. – Т. 7. – №. 5. – С. 2706.

8. Chiesa M., Kindler G., Schapira M. Traffic engineering with equal-cost-multipath: An algorithmic perspective //IEEE/ACM Transactions on Networking. – 2016. – Т. 25. – №. 2. – С. 779-792.

9. Awduche D. et al. Requirements for traffic engineering over MPLS. – 1999. –№. rfc2702.

10. Enyedi G., Rétvári G. Finding multiple maximally redundant trees in linear time //Periodica Polytechnica Electrical Engineering (Archives). – 2010. – Т. 54. – №. 1-2. – С. 29-40.

11. Zehavi A., Itai A. Three tree-paths //Journal of Graph Theory. – 1989. – T. 13.– №. 2. – C. 175-188.

12. Curran S., Lee O., Yu X. Finding four independent trees //SIAM Journal onComputing. – 2006. – T. 35. – №. 5. – C. 1023-1058

13. The Internet Topology Zoo [HTML] (http://topology-zoo.org/).