Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Моделі та методи управління потоками даних в корпоративних комп'ютерних мережах

(тема)

Виконав:
студент II курсу, групи КСМм-19-1
Процик О.В.
(прізвище, ініціали)

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

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

Освітня програма
Комп'ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Янковський О.А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Комп'ютерні системи та мережі
(повна назва)

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

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Процику Олексію Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Моделі та методи управління потоками даних в корпоративних комп'ютерних мережах

затверджена наказом по університету від " 30 " жовтня 2020 р. № 1487 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 14 грудня 2020 р.

3. Вхідні дані до роботи 1) моделі та методи для керування мережевими інформаційними потоками; 2) сучасні вимоги до мережних показників; 3) перелік використаних програмних та апаратних засобів: ОС Windows 10, OpNet 14, NS-2.

4. Перелік питань, що потрібно опрацювати в роботі
1) аналіз сучасного стану проблеми
2) огляд технологій керування інформаційними потоками
3) модель управління мережним трафіком
4) вибір програмних та апаратних засобів реалізації
5) проведення експериментальних досліджень
6) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) слайди презентації - 19 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	Аналіз стану проблеми та сучасних методів її	03.11.20–06.11.20	
	вирішення		
2	Огляд технологій керування інформаційними	07.11.20–16.11.20	
	потоками		
3	Розробка моделі управління мережним	17.11.20–20.11.20	
	трафіком		
4	Вибір програмних та апаратних засобів	21.11.20–30.11.20	
	реалізації		
5	Тестування метода управління мережним	01.12.20–02.12.20	
	трафіком		
6	Оформлення пояснювальної записки	03.12.20–08.12.20	
7	Оформлення графічних матеріалів	09.12.20–10.12.20	

Дата видачі завдання 2 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Янковський О.А.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 97 с., 43 рис., 1 дод., 29 джерел.

AQM, АЛГОРИТМ, ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНІ СИСТЕМИ, МЕТРИКА, ПЕРЕВАНТАЖЕННЯ, ТЕЛЕКОМУНІКАЦІЙНІ МЕРЕЖІ, ТОПОЛОГІЯ.

Мета атестаційної роботи – розробка моделей та методів управління інформаційними потоками в сучасних IP-мережах.

Проведено огляд теоретичних аспектів функціонування протоколів TCP та IP, виконано аналіз існуючих інструментальних засобів щодо оцінки ефективності різноманатних мережевих протоколів, досліджено методи для підвищення пропускної здатності комп’ютерних мереж.

ABSTRACT

Master’s thesis: 97 pages, 43 figures, 1 appendices, 29 sources.

AQM, ALGORITHM, INFORMATION AND COMMUNICATION SYSTEMS, METRICS, OVERLOADING, TELECOMMUNICATIONS NETWORKS, TOPOLOGY.

The purpose of certification work is to develop models and methods of information flow management in modern IP networks.

A review of the theoretical aspects of the operation of TCP and IP protocols, an analysis of existing tools for evaluating the effectiveness of multinational network protocols, investigated methods for increasing the bandwidth of computer networks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AQM – активне управління чергою (англ., Active Queue Management)

ARED – адаптивне раннє випадкове виявлення (англ., Adaptive Random Early Detection)

AS – автономна система (англ., Autonomous System)

AVQ – адаптивна віртуальна черга (англ., Adaptive Virtual Queue)

DRED – динамічне раннє випадкове виявлення (англ., Adaptive Random Early Detection)

ECN – явне повідомлення про перевантаження (англ., Explicit Congestion Notification)

FRED – потоковий RED (англ., Flow RED)

Gbps – гигабіт за секунду (англ., Gigabit per second)

GRED – пологий RED (англ., Gentle RED)

ICMP – протокол міжмережевих керуючих повідомлень (англ.,Internet Control Message Protocol)

IP – Інтернет-протокол (англ., Internet Protocol)

LAN – локальна мережа (англ., Local Area Network)

Mbps – мегабіт за секунду (англ., Megabit per second

PQM – пасивне управління чергою (англ., Passive Queue Management)

SRED – стабілізований RED (англ., Stabilized RED)

WAN – глобальна мережа (англ., Wide Area Network)

ВСТУП

У зв'язку з величезним зростанням Інтернет-трафіку і появою нових його типів, а також для задоволення запитів на надання все більш різноманітного діапазону мережевих послуг, контроль за перевантаженнями, які виникають, став одним з найбільш важливих завдань в існуючих комп'ютерних мережах. Контроль мережевих перевантажень, що дозволяє різним типам Інтернет-трафіку задовольняти заданим обмеженням якості обслуговування (QoS), стає дуже важливим. Більша частина мережевих додатків вимагає прогнозування перевантаження, що насувається, ще до того, як воно виникло.

Перевантаження виникає в мережевих маршрутизаторах, коли кількість вхідних пакетів перевищує доступні мережеві ресурси, такі як буферний простір і виділення смуги пропускання. Це може привести до зниження продуктивності мережі щодо середньої затримки в черзі пакетів, швидкості втрати пакетів і пропускної здатності.

Традиційний метод уникнення перевантажень складається в управлінні довжиною черги маршрутизатора. Він полягає тільки в установці максимальної довжини для кожної черги, зазвичай рівний ємності буфера і має на увазі прийом пакетів до тих пір, поки черга не заповниться. Розміщення новоприбулих пакетів блокується до тих пір, поки в черзі не визволиться якесь місце в результаті відправки деяких пакетів. Цей метод відомий як «tail drop», оскільки пакет, який прибув найостаннішим (тобто той, який знаходиться в кінці черги), відкидається, коли черга заповнюється. Цей метод використовувався в Інтернеті протягом багатьох років, але у нього є два важливих недоліки: «Lock-Out» і «Full Queues» («блокування» і «переповнення черги»).

Для того, щоб вирішити ці проблеми, були запропоновані і реалізовані деякі механізми активного управління чергою (AQM) для управління

довжиною черги, зменшення наскрізної затримки, зменшення ймовірності втрати пакетів і запобігання явищ блокування. Контроль перевантаження може бути досягнутий за рахунок використання певних схем управління буфером. До даних схемами управління буфером можна віднести Random Early Detection (RED) [1], Random Early Marking (REM) [2], схему на основі віртуальної черги, в якій віртуальна чергу є адаптивної [3], а також механізм пропорційного інтегрального регулятора [4] та ін.

Проте ці схеми чутливі до налаштувань параметрів і мають слабкі сторони в виявленні і контролі перевантаження в динамічно мінливих мережевих ситуаціях. Ще одним недоліком алгоритмів AQM є те, що вони застосовуються тільки до марковських моделей, які розглядаються як моделі трафіку, що залежать від ближнього радіусу дії (Short Range Dependent – SRD). Однак вимірювання трафіку в мережах зв'язку показують, що мережевий трафік може проявляти як самоподібні, так і залежні від дальнього радіусу дії (Long Range Dependent – LRD) властивості. Отже, важливо розробляти нові алгоритми не тільки для управління перевантаженням, а й для того, щоб мати можливість прогнозувати початок перевантаження в мережі.

1 ЗАДАЧА АТЕСТАЦІЙНОЇ РОБОТИ

У наші дні в Інтернеті широко розгорнуто ряд нових додатків для потокової передачі аудіо/відео, передачі голосу по IP тощо. Ці програми підтримуються протоколом дейтаграм користувача (UDP), а не TCP. У UDP відсутнє наскрізне управління потоком, а також механізм управління перевантаженням. У UDP джерела самостійно регулюють швидкість передачі і не беруть до уваги перевантаження мережі. Через зростаючу тенденцію розгортання UDP і його нездатність керувати перевантаженням ці додатки призводять до двох основних проблем для Інтернету: перевантаження і несправедливого розподілу смуги пропускання.

Метою даної магістерської атестаційної роботи є розробка деяких нових методів контролю перевантаження для комп'ютерних мереж, які використовують різні характеристики трафіку.

В рамках даної магістерської атестаційної роботи необхідно:

- провести огляд сучасних методів контролю перевантажень в комп'ютерних мережевих каналах;
- провести аналіз факторів, що впливають на продуктивність мереж;
- провести огляд різних методів управління трафіком комп'ютерних мереж;
- розробити метод управління чергами маршрутизаторів відповідно до поточного мережевого стану;
- провести імітаційне моделювання теоретичних викладок;
- провести аналіз отриманих результатів.

2 ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

2.1 Перевантаження в мережі Інтернет

Інтернет – це велика комп'ютерна мережа, яка обмінюється даними між мільйонами користувачів за допомогою комутації пакетів на основі протоколу TCP/IP. Перевантаження зазвичай виникає в мережі з комутацією пакетів, коли кількість переданих пакетів досягає пропускної здатності мережі. Більш того, мережа вважається перевантаженою, коли кількість пакетів, які потребують передачі, перевищує кількість пакетів, якими мережа може управляти [5].

Перевантаження може викликати погіршення продуктивності мережі, наприклад, тривалу наскрізну затримку, високу частку втрачених пакетів і можливий колапс перевантаження через постійну повторну передачу втрачених пакетів по протоколу TCP [6]. Мережевий колапс, пов'язаний з перевантаженням був вперше виявлений Джоном Нагелем в 1984 році [7], і цей колапс спонукав дослідників зосередитися на контролі перевантаження і механізмах управління буфером. Швидке розширення різноманітності використовуваних додатків в сучасних мережах значно ускладнює роботу механізмів контролю перевантаження TCP і запобігання перевантажень [8].

2.2 Причини виникнення перевантажень

Сотні мільйонів користувачів підключені до мережі Інтернет для зберігання і передачі інформації, такої як повідомлення електронної пошти та веб-сторінки. Користувачі або кінцеві системи пов'язані один з одним різними лініями зв'язку, коаксіальними кабелями, крученою парою, оптичними волокнами і різноманітними системами бездротового з'єднання. У більшості випадків ці кінцеві системи побічно пов'язані одна з однією через

проміжні пристрої зв'язку, звані маршрутизаторами. Маршрутизатор отримує інформацію від користувачів і направляє її в потрібне місце в мережі. Інформація, що відправляється, може пройти через кілька маршрутизаторів, перш ніж вона досягне місця призначення. Кожен з мережевих маршрутизаторів має обмежений буферний простір, в якому він може розмістити інформацію перед її відправкою на інший маршрутизатор або в кінцевий пункт призначення.

Буферний простір необхідний для запобігання втрати пакетів, коли дані надходять зі швидкістю, що перевищує максимальну швидкість обробки маршрутизатора. Дані, які не можуть бути оброблені і переадресовані безпосередньо, зберігаються в буфері маршрутизатора до тих пір, поки не буде доступний процесор маршрутизатора [9]. Через обмеження простору буферів в мережі і різкого збільшення трафіку, що передається, черги маршрутизаторів швидко переповнюються, що призводить до відкидання багатьох пакетів. На рисунку 2.1 показано переповнення буфера маршрутизатора, яке призводить до перевантаження.

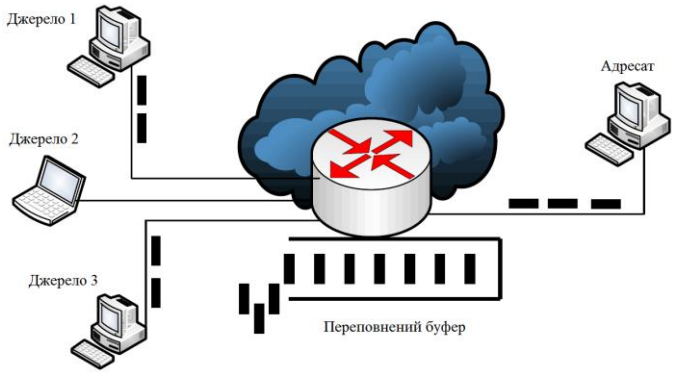

Рисунок 2.1 – Переповнення буфера в проміжному пристрої, що приводить до перевантаження

Балансування доступної смуги пропускання – ще одна проблема, яка може привести до перевантаження мережевих каналів вхідного і вихідного трафіку. Іншими словами, канали з низькою пропускною здатністю або

великі відмінності швидкостей передачі між різними з'єднаннями можуть викликати серйозне перевантаження. Незважаючи на те, що протягом багатьох років було запропоновано безліч варіантів розв'язання проблеми, перевантаження як і раніше залишається серйозною проблемою в сучасних мережах. Багато із запропонованих механізмів забезпечують лише часткове вирішення, маючи на увазі, що потенційні вузькі місця або перевантаження в цілому перемістяться в інші точки в мережі, які необхідно розглядати окремо [10].

2.3 Перевантаження і продуктивність мережі

Сучасні телекомунікаційні та комп'ютерні мережі, включаючи Інтернет, проектуються таким чином, щоб мінімізувати втрати інформації і забезпечувати високу якість обслуговування. У мережевих буферах виникають сплески кількості пакетів, коли пакети приходять з короткими інтервалами, і в результаті швидкість маршрутизатора буде менше, ніж це потрібно мережі [11]. У такій ситуації пакети відкидаються і генеруються повідомлення про перевантаження.

В іншій ситуації, коли отримання пакетів не підтверджуються вчасно, повідомляється про перевантаження, яке в цьому випадку є результатом затримки постановки в чергу. Перевантаження впливає на продуктивність мережі через неповне використання каналів і тривалих затримок, а в перевантаженій мережі черги в проміжних пристроях, таких як маршрутизатори і комутатори, різко збільшуються, і вхідні пакети починають відкидатися. Затримка і пропускна здатність – два найбільш важливих показника якості обслуговування в мережі, і обидва вони залежать від перевантаження мережі. Коли в буфер маршрутизатора надходить інтенсивний трафік, довжина черг збільшується, і відповідно довгі черги збільшують затримку передачі пакетів. З іншого боку, втрата пакетів в перевантажених мережах означає різке зниження пропускної здатності.

Пакети, які були відкинуті або втрачені через виникле перевантаження, повинні бути повторно передані, і може виникнути багато повторних передач, так як, джерело інформації буде продовжувати повторно передавати один і той же пакет, поки пункт призначення не відправить підтвердження успішного отримання пакету. Такий розвиток подій часто викликає існування безліч небажаних копій одного і того ж пакету в мережі. Це явище вважається основним недоліком використовуваних сьогодні механізмів управління передачею інформації. Дублювання пакетів призводить до неправильного використання мережевої смуги пропускання [12].

Забезпечення якості обслуговування (QoS) є дуже важливою проблемою для сучасних користувачів мережі Інтернет в зв'язку з розвитком мультимедійних додатків, таких як передача голосу по IP і потокове відео. У таких додатках потрібна підтримка QoS, особливо щодо затримки передачі.

На рисунку 2.2 показано, як навантаження трафіку впливає на затримку і пропускну здатність мережі. Крім того, перевантаження може викликати серйозне зниження продуктивності, особливо якщо мережа продовжує залишатися перевантаженою, що призводить до ситуації колапсу перевантаження.

2.4 Методи контролю перевантаження

У зв'язку з швидким поширенням мережевих технологій, контроль перевантаження став важливою проблемою, особливо з різким збільшенням обсягів переданого трафіку реального часу, тому різні методи контролю перевантаження були розроблені з моменту появи явища перевантаження в Інтернеті.

Доступні рішення по управлінню перевантаженням поділяються на дві основні категорії: розімкнуті і замкнуті, в залежності від мети рішення по управлінню трафіком. Якщо метод управління є превентивним методом, іншими словами, що дозволяє уникнути перевантаження, в цьому випадку

управління перевантаженням називається розімкненим циклом. З іншого боку, механізми замкнутого циклу є реактивними методами і працюють з перевантаженням після того, як воно стался, або на його ранніх стадіях.

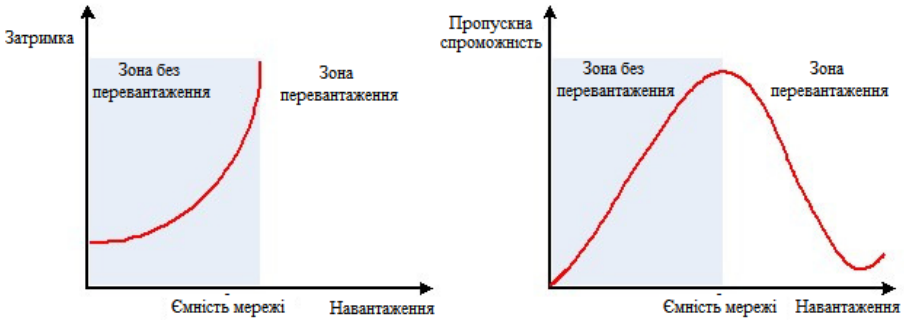

Рисунок 2.2 – Параметри затримки і пропускної здатності в залежності від навантаження

На рисунку 2.3 показана класифікація методів контролю перевантаження.

Розімкнутий цикл – це метод, спрямований на запобігання виникненню перевантаження без реалізації зворотного зв'язку. Такий метод боротьби з перевантаженням вирішує проблему за рахунок правильних політик проектування на різних рівнях OSI, таких як транспортний, мережевий і канальний. Ці політики проектування включають політику повторної передачі, політику підтвердження і політику відкидання пакетів.

Розімкнутий цикл не відстежує стан мережі, але вимагає знання і врахування мережевих ресурсів. Перед початком нового сеансу визначається маршрут передачі пекетів і вимоги до ресурсів в транзиті.

Цей тип запобігання перевантаження підходить для телефонних мереж, коли всі виклики мають майже фіксовані вимоги до смуги пропускання [1]. Однак в Інтернеті є різні додатки, поведінка яких відрізняється і не може бути передбачена заздалегідь. Таким чином, щоб запобігти перевантаженню мереж, необхідно додати зворотний зв'язок, який має на увазі, що більшість дій спрямовані на запобігання перевантаження.

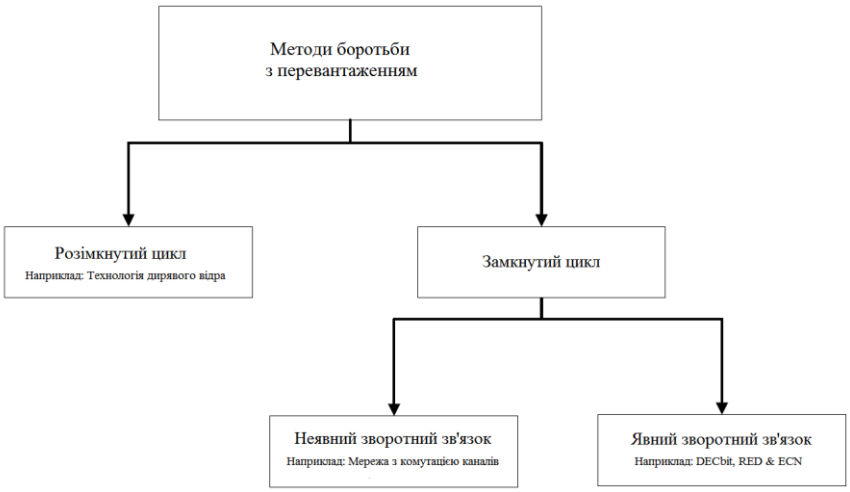

Рисунок 2.3 – Класифікація методів контролю перевантаження

Методи замкнутого циклу вирішують проблему перевантаження після її виникнення шляхом моніторингу мережі для виявлення можливих ситуацій перевантаження, що вимагає використання відповідного механізму зворотного зв'язку для відновлення після виникнення перевантаження. Доступні методи зворотного зв'язку діляться на дві різні категорії: з неявним зворотним зв'язком і з явним зворотним зв'язком.

Неявний зворотний зв'язок забезпечує наскрізне управління перевантаженням за рахунок застосування механізмів зворотного зв'язку між джерелом і одержувачем.

Неявний зворотний зв'язок намагається підтримувати високу пропускну здатність і низьку затримку пакетів даних за рахунок часу RTT і дублювання підтверджень для втрачених пакетів.

Неявний або непрямий зворотний зв'язок означає, що джерела збільшують свої швидкості до тих пір, поки буфери маршрутизатора не заповнені, але цей тип зворотного зв'язку не забезпечує досить високу ефективність у мережах, оскільки спочатку виникає перевантаження, а тільки потім реакція на нього [13].

Такий метод просто реалізувати, але він неефективний з точки зору продуктивності мережі. Хорошим прикладом неявного зворотного зв'язку є

наскрізне управління перевантаженням TCP.

Через недоліки неявного зворотного зв'язку виникла потреба в більш ефективних механізмах зворотного зв'язку, які могли б передавати точну інформацію про перевантаження джерел шляхом відкидання або маркування пакетів, щоб запобігти переповнення буферів в проміжних пристроях. Цей метод називається явним зворотним зв'язком.

На відміну від неявного зворотного зв'язку, коли кінцеві вузли неявно оцінюють перевантаження, проміжні пристрої з явним зворотним зв'язком, такі як маршрутизатори і комутатори, сигналізують джерелам про виникнення затримок у передачі пакетів. Для того, щоб змінити стан перевантаження надається точна інформація про перевантаження до того, як проміжні буфери маршрутизаторів будуть повністю заповнені, і, відповідно, відправники зможуть зменшити свої швидкості передачі раніше, і як наслідок, черги будуть рости значно повільніше [13].

Є багато методів замкнутого циклу. Розглянемо деякі з цих підходів.

Біт попередження: в цьому методі проміжні пристрої контролюють мережу на предмет перевантаження. Як тільки можливе перевантаження виявляється в мережі, до заголовку пакета додається біт попередження, для того, щоб попередити джерело про перевантаження. Джерело отримує біт попередження в пакеті підтвердження (ACK) від пункту призначення і відповідно знижує швидкість відправки. Джерело продовжує регулювати швидкість передачі до тих пір, поки маршрутизатор не перестане відправляти біт попередження, на цьому етапі джерело повернеться до нормальної швидкості передачі, поки не отримає інше попередження.

Блокування пакетів: цей механізм змушує джерело знижувати швидкість передачі в разі перевантаження, відправляючи пряме повідомлення з перевантаженого вузла. Це повідомлення може передавати більш докладну інформацію джерела, наприклад, сигналізувати джерела про необхідність знизити швидкість передачі даних, наприклад, на 50%, або вказувати допустиму швидкість передачі даних.

Коли джерело отримує блокуючий пакет, воно повинен знизити швидкість його відправки до місця призначення на необхідний відсоток. Після отримання пакета блокування джерело повинно ігнорувати будь-які інші блокуючі пакети, які надходять від того ж вузла призначення протягом фіксованого інтервалу часу, а потім джерело знову починає прослуховування протягом іншого інтервалу часу.

Якщо джерело отримує блокуючий пакет протягом цього нового інтервалу часу, воно повинен додатково знизити швидкість відправки, оскільки перевантаження все ще присутнє. Якщо протягом періоду прослуховування блокуючі пакети не надходять, джерело знову починає збільшувати швидкість потоку.

На рисунку 2.4 зображений метод роботи блокуючих пакетів.

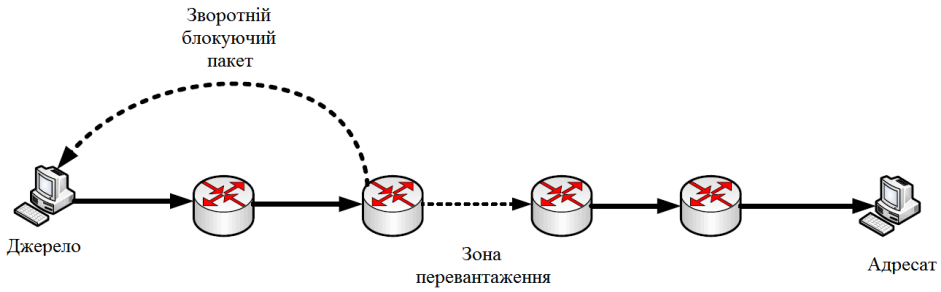

Рисунок 2.4 – Метод блокуючого пакета

У блокуючих пакетів є пріоритет швидкого зворотного відправлення джерел для зниження їх швидкості відправки, що дозволяє швидко відновити роботу мережі після перевантаження. Крім того, цей зворотний зв'язок надходить від маршрутизаторів з вузьких місць, а це означає, що дана інформація про виникле перевантаження дуже точна.

З іншого боку, механізм блокування пакетів має недоліки, оскільки маршрутизаторам доводиться генерувати пакети зворотного зв'язку, які вимагають виділення обчислювальних ресурсів, в той час як самі маршрутизатори перевантажені.

Більш того, блокуючі пакети виключають адресата в контурі зворотного зв'язку, що впливає на продуктивність мережі, наприклад на оцінку RTT.

Пакет з перевантаженням по кроках: в цій схемі кожен маршрутизатор на наскрізному шляху надає зворотний зв'язок про перевантаження безпосередньо попередньому маршрутизатору, як показано на рисунку 2.5. Попередній вузол потім змінює свою швидкість відправки в залежності від отриманої інформації.

У традиційному блокувальному пакеті інформація зворотного зв'язку повинна проходити довгий шлях від віддаленого пункту призначення до відповідного джерела, тоді як в пакетному методі будь-який вузол на шляху трафіку повинен знизити свою швидкість передачі після отримання пакету блокування, таким чином пакетний метод перевантаження по переходу забезпечує швидку реакцію на перевантаження.

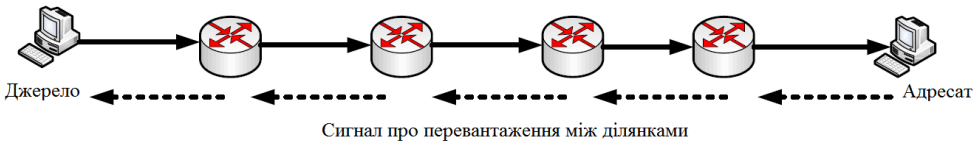

Рисунок 2.5 – Управління перевантаженням по кроках

Механізм зворотного зв'язку з попереднім вузлом залежить від розміру черги в перевантаженому вузлі. Коли довжина черги досягає порогового значення, попередні вузли повідомляються про перевантаження і автоматично знижують свою швидкість передачі, щоб зменшити перевантаження. Такі механізми реагують на перевантаження швидше, ніж наскрізні схеми, що призводить до кращої продуктивності мережі в порівнянні з наскрізними схемами.

Проте, покрокові схеми вимагають значного обсягу обчислювальних ресурсів на кожному проміжному пристрої на шляху трафіку, що спричиняє

значні втрати в маршрутизаторах. Крім того, у покрокової схеми є проблема масштабованості через необхідність управління станом кожного потоку в кожному проміжному пристрої [14].

За минулі роки в літературі було представлено багато методів контролю перевантаження, спрямованих на зменшення перевантаження і підвищення мережевої продуктивності. Активне управління чергою – важлива концепція, яка використовується для контролю перевантаження в мережах.

AQM іноді використовує один біт явного повідомлення про перевантаження (ECN) в заголовку пакета, щоб повідомити кінцевим користувачам або кінцевим вузлам про перевантаження, яке виникає в проміжних пристроях, таких як маршрутизатори або шлюзи.

2.5 Контроль перевантаження в протоколі TCP

Протокол управління передачею (TCP) – це протокол, орієнтований на з'єднання, це означає, що з'єднання має бути встановлено між двома кінцевими користувачами (відправником і отримувачем) до початку передачі [12]. Алгоритми управління перевантаженням TCP, реалізовані в кінцевих системах, використовують втрату пакетів як показник перевантаження мережі. Цей алгоритм був розроблений для управління розміром черги буфера, так як затримка в чергах збільшує наскрізну затримку і впливає на продуктивність різних мережевих додатків [15].

Як згадувалося вище, TCP має наскрізний зворотний зв'язок, і в TCP реалізована функція контролю перевантаження. TCP – це протокол, який протягом багатьох років використовувався для уникнення перевантажень в комунікаційних мережах, таких як Інтернет. TCP – важливий елемент набору протоколів Інтернету і забезпечує надійну передачу даних між користувачами мережі по ненадійному каналу зв'язку на основі IP. Протокол TCP є основою управління мережевими перевантаженнями.

Механізм управління перевантаженням в TCP дотримується принципу «збереження пакетів», згідно з яким пакети не повинні пропускатися в мережі зі швидкістю, більшою, ніж вони залишають мережу в стані рівноваги [8]. Таким чином, механізми контролю перевантаження в TCP обмежують кількість пакетів для кінцевого користувача, що знаходяться всередині мережі в будь-який момент часу. Оскільки в будь-який момент складно дізнатися, які пакети знаходяться всередині мережі, а які вже ні, алгоритми контролю перевантаження TCP припускають, що всі непідтверджені пакети все ще знаходяться всередині мережі, а всі пакети, підтверджені адресатом, вже покинули мережу.

Використовуючи вищенаведене припущення, максимальну кількість непідтверджених пакетів для кожного відправника буде обмежено алгоритмами контролю перевантаження TCP. Ця максимальна кількість пакетів відома як вікно перевантаження TCP (cwnd), вона змінюється з часом і залежить від стану мережі [16]. Контроль перевантаження TCP розділений на чотири алгоритми; повільний старт, запобігання перевантаження, швидкої повторної передачі і швидкого відновлення.

Повільний старт контролює розмір cwnd, в той час як кінцева система намагається знайти стан рівноваги мережі. Після того, як крайова система досягла стану рівноваги мережі, алгоритм запобігання перевантаження використовується для управління розміром cwnd. Два алгоритму, швидкого відновлення і швидкої повторної передачі, були запропоновані для відновлення після декількох повторних передач пакетів під час таймаутів.

Алгоритм повільного старту TCP використовується кінцевими користувачами для управління швидкістю передачі. При повільному старті відправник збільшує швидкість передачі, експоненціально збільшуючи cwnd.

Cwnd запускається відправником з невеликої кількості пакетів (зазвичай одного або двох) після встановлення TCP-з'єднання. Потім відправник чекає необхідного підтвердження (ACK) від пункту призначення, що підтверджує успішне отримання пакета.

Після цього вікно перевантаження збільшується до двох пакетів, і відправник очікує одержання двох ACK, потім вікно перевантаження збільшується до чотирьох пакетів і так далі до тих пір, поки між відправником і отримувачем не виникне тривалого періоду відсутності підтверджень.

Після того, як у відправника закінчиться час очікування отримання відповідного ACK або відправник отримає дублікати ACK, що вказує відправнику на можливість перевантаження, він зменшить своє вікно перевантаження наполовину.

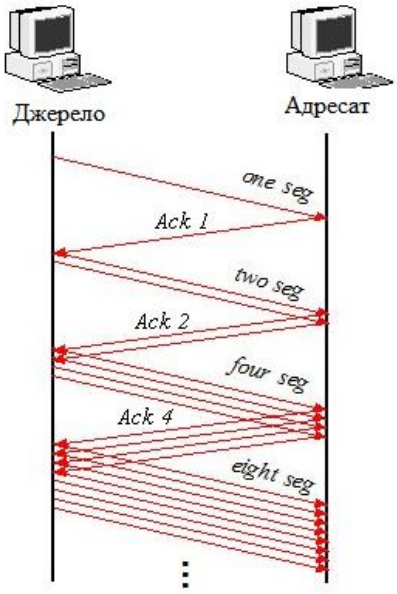

Рисунок 2.6 – Робота алгоритму повільного старту

Якщо одержувач підтверджує кожен пакет даних пакетом ACK, відправник подвоює своє вікно перевантаження протягом часу кругового обігу (RTT).

Насправді повільний старт зовсім не повільний, незважаючи на свою назву. Таким чином, коли немає втрати пакетів, мережа не відчуває перевантаження. Відправник може збільшувати своє перевантажувальне вікно з експоненційною швидкістю.

Припускаючи, що перший пакет був успішно підтверджений, перевантажувальне вікно збільшується до двох сегментів. Після успішної передачі цих двох сегментів і підтверджень вікно збільшується до чотирьох сегментів, а потім до восьми і шістнадцяти і так далі. Коли відправник наближається до межі пропускної здатності мережі або до розміру перевантажувального вікна, оголошеному одержувачем, виникає перевантаження. Коли це відбувається, TCP виходить з фази повільного старту і переходить в режим запобігання перевантаження. На рисунку 2.7 показаний режим повільного старту.

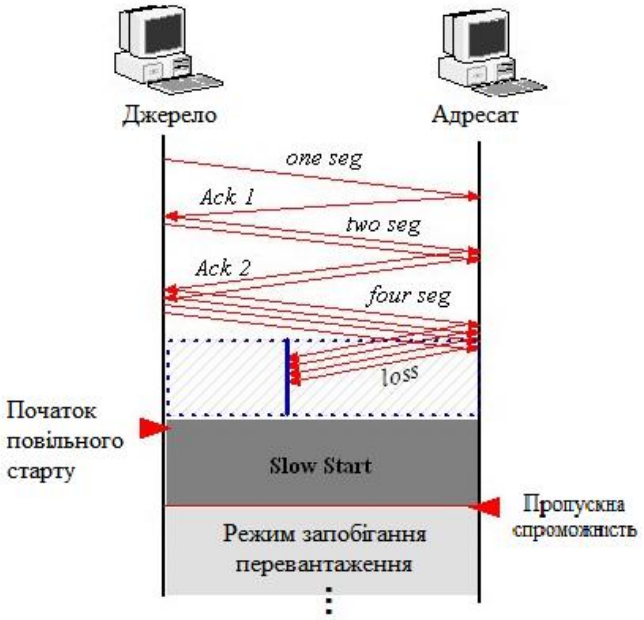

Рисунок 2.7 – Робота алгоритму повільного старту в разі виникнення перевантаження

У режимі запобігання перевантаження, на відміну від повільного старту, відправник пакетів збільшує свій cwnd більш плавно. У якийсь момент під час повільного старту мережа повинна втрачати пакети з якої-небудь причини, але, швидше за все через перевантаження. Якщо втрата пакетів виявлена через потрійні дублюючі ACK, то відправник знає, що принаймні деякі з пакетів прибули до одержувача.

При цьому відправник залишається в фазі запобігання перевантаження, але зменшує свій cwnd наполовину від поточного розміру, принаймі, до двох сегментів. З іншого боку, якщо втрата пакета виявлена по таймауту, то відправник знає, що велика кількість його пакетів було втрачено, оскільки не було отримано жодних ACK і, таким чином, в мережі існує серйозне перевантаження.

У цьому випадку відправник зменшує свій cwnd до одного пакета і перемикається в режим повільного старту, щоб знову знайти рівноважний стан мережі.

Іншими словами, алгоритм повільного старту використовується в поєднанні з алгоритмом запобігання перевантаження, для того, щоб відновити потік даних і одночасно, щоб він не сповільнювався і не залишався повільним.

Алгоритм Fast Retransmit – це третій алгоритм, який реалізований в TCP для боротьби з перевантаженнями. Якщо отриманий дубльований ACK, відправнику важко вирішити, чи викликано це втратою пакету або одержувач отримав пакет не по порядку через затримку.

Зазвичай, якщо одержувач може змінювати порядок сегментів, то незабаром одержувач відправить останнє підтвердження. Іншими словами, в разі несправності не повинно бути отримано більше одного або двох дубльованих ACK.

Однак, якщо отримано більше двох дубльованих ACK, це явна ознака того, що принаймні один сегмент був втрачений. якщо їх три або більше, відправник повторно передасть втрачений сегмент і навіть не чекатиме, поки закінчиться час таймера повторної передачі. Рисунок 2.8 показує повторну передачу сегментів.

Fast Recovery – четвертий алгоритм в механізмі перевантаження TCP, цей алгоритм реалізований разом з алгоритмом повторної передачі. Алгоритм повторної передачі виконується, коли отримані три дубльованих ACK, і, отже, це вказує на перевантаження в мережі, а значить необхідно зменшити

перевантажувальне вікно відправника. Однак ACK, що дублюються, можуть бути згенеровані тільки при отриманні сегмента, і це явна ознака того, що в мережі немає серйозного перевантаження і що втрати сегментів відбуваються рідко.

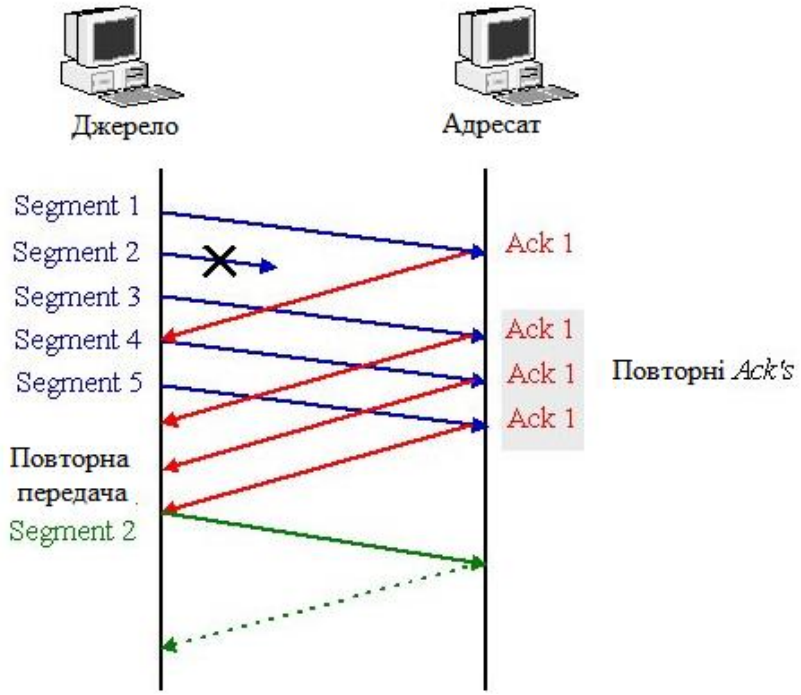

Рисунок 2.8 – Алгоритм швидкої повторної передачі

Fast Recovery – четвертий алгоритм в механізмі перевантаження TCP, цей алгоритм реалізований разом з алгоритмом повторної передачі. Алгоритм повторної передачі виконується, коли отримані три дубльованих ACK, і, отже, це вказує на перевантаження в мережі, а значить необхідно зменшити перевантажувальне вікно відправника. Однак ACK, що дублюються, можуть бути згенеровані тільки при отриманні сегмента, і це явна ознака того, що в мережі немає серйозного перевантаження і що втрати сегментів відбуваються рідко.

В цьому режимі джерело, замість того, щоб зменшити швидкість передачі до одного сегмента і повністю повернутися в режим повільного старту, переходить в режим запобігання перевантаження.

Єдина причина не використовувати повільний старт після швидкої повторної передачі полягає в тому, що між відправником і отримувачем все ще йде передача даних, оскільки ACK генеруються, коли одержувач отримує відповідні сегменти від відправника, і тому швидкість передачі не повинна різко знижуватися.

2.6 Активне управління чергою

Були запропоновані різні механізми управління чергою для управління перевантаженням трафіку і підтримки диференційованих вимог QoS. Традиційний підхід до управління чергою – встановити максимальний ліміт на обсяг даних, які можуть бути поміщені в буфер. Наприклад, в підході Tail Drop (TD) [17] дію по відкиданню пакетів не почнеться, поки не буде вичерпано простір черги. Коли черга заповнюється, TD починає негайно всі пакети, які надходять, відкидати, поки перевантаження не усунуто і в черзі не стане доступним деякий простір. TD як і раніше є корисним механізмом в IP-маршрутизаторах через його надійність і просту реалізацію. На жаль, TD часто викликає великі затримки пакетів, стрибкоподібне відкидання пакетів, погіршення стабільності системи і рівноправності смуги пропускання при наявності постійного перевантаження. «Lock-Out» і «Full Queues» є основними недоліками TD через відкидання пакетів тільки при виникненні перевантаження. Феномен «повних черг» викликає глобальну синхронізацію потоків і, як наслідок, великі затримки пакетів, а також низьку загальну пропускну здатність, тому, що черги TD завжди сповнені або близькі до заповнення протягом тривалих періодів часу. Отже, TD не підходить для інтерактивних мережевих програм через їх вимоги до низької наскрізної затримки.

З іншого боку, в результаті синхронізації феномен «блокування» дозволяє одному з'єднанню або декільком потокам монополізувати простір черги в деяких ситуаціях. Дві інші альтернативні дисципліни черги,

«випадкове відкидання при повному заповненні» і «відкидання фронту при повному заповненні», можуть вирішити проблему «блокування», але не можуть вирішити проблему «повних черг».

Щоб впоратися з обома проблемами і забезпечити низьку наскрізну затримку поряд з високою пропускною здатністю, рекомендується активне управління чергою (AQM) в маршрутизаторах.

Щоб уникнути ситуації, коли буфер зберігає статус «повний» протягом тривалого часу, схема AQM починає відкидати або маркувати пакети до того, як черга буде заповнена, для того, щоб повідомити джерела про початкові етапи перевантаження. При отриманні сигналу перевантаження (тобто маркування явного повідомлення про перевантаження (ECN) або відкинутих пакетів), згенерованого схемою AQM, джерела трафіку зменшують обсяг трафіку, що вводиться в мережу. Зберігаючи невелику середню довжину черги, AQM зменшує середню затримку постановки в чергу і зменшує кількість відкинутих пакетів, що призводить до збільшення коефіцієнта використання каналу зв'язку через поглинання більшої кількості пакетів і уникнення глобальної синхронізації.

Двома ключовими проблемами в механізмі AQM є питання коли і як відкидати/маркувати пакети, а також, що визначає початкову стадію перевантаження і, відповідно, відображає ступінь перевантаження. Обидва ці питання мають значний вплив на ключові показники продуктивності, включаючи коефіцієнт використання, пропускну здатність, середню затримку в черзі і ймовірність втрати пакетів.

2.6.1 AQM на основі черг

Випадкове раннє виявлення (RED), спочатку запропоноване в [18], є найбільш відомим алгоритмом AQM. RED в даний час широко застосовується в маршрутизаторах. RED відстежує експоненціально-зважене середнє значення (EWMA) довжини черги і випадковим чином відкидає

пакети з ймовірністю відкидання, яка лінійно збільшується із середньою довжиною черги між двома граничними значеннями.

Як перший механізм AQM, RED був розроблений для вирішення проблем, виявлених в роботі TD. Рівномірно випадкові ранні мітки/відкидання пакетів розподіляються в часі, що знижує ймовірність глобальної синхронізації джерел трафіку. Більш того, з RED середня довжина черги зменшується, що дозволяє маршрутизаторам пристосуватися до пік трафіку. Отже, механізм RED перевершує політику TD з точки зору пропускної здатності, затримки в чергах і справедливості обслуговування. Крім того, комбінація RED і явного повідомлення про перевантаження (ECN) може поліпшити продуктивність TCP в бездротових мережах.

Однак поряд з поглибленими дослідженнями і розглядом більшої кількості сценаріїв були виявлені деякі серйозні недоліки RED, наприклад, складність настройки його параметрів. Відомо, що конкретна установка параметрів RED застосовна тільки у вузькому діапазоні мережевих умов. В [19] було запропоновано ітераційний алгоритм для оптимізації двох порогів RED, але цей алгоритм неможливий через його тривалий час виконання до досягнення оптимальних значень. Таким чином, досить важко вибрати набір значень параметрів, щоб збалансувати компроміс між різними показниками продуктивності з різними сценаріями. Також відомо, що цей механізм управління потоком в кінцевому підсумку стає нестабільним в міру збільшення затримки RTT або збільшення пропускної здатності мережі.

Три ключові проблеми, пов'язані зі схемою RED AQM: установка параметрів, нечутливість до зміни навантаження вхідного трафіку і невідповідність між макроскопічною і мікроскопічною поведінкою динаміки довжини черги. На основі аналізу великих експериментів з агрегованим трафіком, що містить різні категорії потоків, можна продемонструвати недосконаліть RED через використання середньої довжини черги, особливо коли середнє її значення далеко від миттєвої довжини черги. Взаємодія між середньою довжиною черги і гострим краєм функції відкидання призводить

до деякої патології, такої як збільшення ймовірності відкидання потоків UDP і збільшення кількості послідовних втрат.

Щоб подолати цей недолік, Фолід і Фолл [20] запропонували використовувати функцію плавного спуску, навіть коли середня довжина черги перевищує максимальний поріг і назвали цей покращений алгоритм Gentle-RED (GRED). RED і GRED відносяться до AQM на основі черг за ступенем серйозності перевантаження. Також відомі безліч інших AQM на основі черг, такі як, стабілізований RED (SRED), Adaptive RED (ARED), Exponential-RED (E-RED) і т.д.

2.6.2 AQM на основі швидкості надходження пакетів

Друга класифікація AQM, названа на основі швидкості, використовує швидкість надходження пакетів для управління перевантаженням. Цілі алгоритмів AQM на основі швидкості (тобто BLUE [21], адаптивної віртуальної черги (AVQ) [22], стабілізованого AVQ (SAVQ) [23] і підходу на основі використання каналу (LUBA) [24]) полягають в зниженні невідповідності швидкості між постановкою в чергу і витягом з черги і, таким чином, призводить до незначного погіршення пакетів, малу затримку і високий ступінь використання каналу. Як типовий представник схем AQM на основі швидкості, BLUE, був першою спробою зв'язати втрату пакетів, а також використання каналу, а не довжину черги і повідомлення про перевантаження.

BLUE може ефективно відправляти назад повідомлення про перевантаження з правильною швидкістю, оскільки ймовірність, що використовується для маркування/відкидання пакетів при їх постановці в чергу, регулюється відповідно до подій втрати пакетів і бездіяльності каналу. Тобто, BLUE збільшує ймовірність того, що пакети, які приходять, постійно відкидаються через переповнення буфера. І навпаки, ймовірність зменшується, якщо черга стає порожньою.

BLUE має кращу продуктивність, ніж RED, з точки зору швидкості втрати пакетів і вимог до розміру буфера. З іншого боку, відомо, що BLUE погано працює при невеликому розмірі буфера через відсутність контролю середньої довжини черги і значні втрати можуть відбуватися під час вступу імпульсного трафіку.

2.6.3 Комбінація буфера і AQM на основі швидкості

Щоб досягти компромісу між стабільністю черг і швидкодією, метрики, використовувані в деяких схемах AQM (наприклад, випадкове експоненціальне маркування (REM) [25], стабілізований віртуальний буфер (SVB) [26] і RaQ [27]), засновані на комбінації параметрів буфера і швидкості.

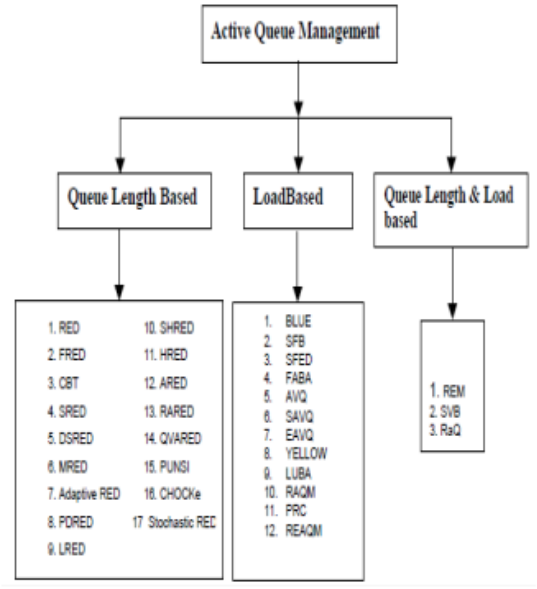

Рисунок 2.9 – Класифікація AQM

Наприклад, REM підтримує змінну, яка називається «Вартість», яка оновлюється на основі невідповідності швидкості (тобто різниці між вхідною швидкістю і пропускною спроможністю каналу) і невідповідності довжини

черги (тобто різниці між довжиною черги і цільовою довжиною) в якості міри перевантаження. «Вартість» збільшується, якщо зважена сума цих розбіжностей позитивна, і зменшується в іншому випадку. Зважена сума позитивна, коли або швидкість введення перевищує пропускну здатність каналу, або існує надлишкова довжина черги, яку необхідно очистити, і негативна в іншому випадку. Імовірність маркування змінюється експоненціально залежно від змінної «Вартість».

3 АКТИВНІ МЕТОДИ УПРАВЛІННЯ ЧЕРГОЮ (AQM)

3.1 Огляд AQM

В сучасних комп'ютерних мережах трафік обробляється максимально швидко, але немає ніяких гарантій щодо своєчасності або фактичної його доставки.

У сучасному Інтернеті є все зростаючий попит на якість обслуговування (QoS) і справедливість між інформаційними потоками, тому управління чергами маршрутизаторів є вельми важливим завданням.

Метою алгоритмів управління чергою є реалізація методу управління буфером в проміжних пристроях в спробі пересилати або обробляти дані в дорозі до місця призначення відповідно до вимог QoS. Алгоритми управління чергою діляться на дві категорії в залежності від методу ймовірності скидання, який реалізований в системі управління. Перша і найпростіша з двох категорій – це пасивне управління чергою (PQM), яке вважається традиційним алгоритмом управління чергою. Ця категорія відкидає пакети тільки при заповненні черги. Хорошим прикладом цього типу управління чергою є Drop Tail (DT).

Друга категорія – це активне управління чергою (AQM), і в цій категорії пакети відкидаються, коли буфер заповнений, і також скидаються з якоюсь імовірністю до того, як буфер заповниться. Прикладом AQM є випадкове раннє виявлення (RED).

3.2 Пасивне управління чергою

PQM – традиційний спосіб управління довжиною черги на маршрутизаторах; це як і раніше одна з популярних конфігурацій через простоту своєї реалізації.

У PQM максимальна довжина встановлюється для кожної черги на маршрутизаторі і враховуються всі пакети, які надходять, поки не буде досягнута максимальна довжина черги. Після переповнення буфера пакети відкидаються до тих пір, поки розмір черги знову не стане нижче максимального допустимого в результаті деяких відхилень.

PQM був популярним методом управління чергою протягом багатьох років через його простоту, хоча він призводить до деяких раніше зазначених серйозних недоліків, таким як проблема глобальної синхронізації і феномен блокування. Існує багато PQM, і найбільш відомим алгоритмом є Drop Tail (DT). Існують і інші алгоритми PQM, подібні Drop-Tail, які можуть застосовуватися при заповненні черги, такі як Drop Front і Random Drop on Full.

3.2.1 Drop Tail

Drop tail – традиційний спосіб управління довжиною черги на маршрутизаторах і найбільш часто використовуваний в мережевих шлюзах. В DT максимальна довжина встановлюється на маршрутизаторі для кожної черги, яка вміщує всі пакети, які надходять, поки не буде досягнута максимальна довжина черги. Після переповнення буфера пакети починають скидатися доти, поки розмір черги знову не стане нижче максимального.

Алгоритм DT реагує тільки тоді, коли буфер вже заповнений, що викликає високу частоту втрати пакетів, так як буфер може бути заповнений більшу частину часу, що також призводить до значних затримок.

Глобальна синхронізація відбувається, коли всі з'єднання одночасно знижують швидкість передачі, коли кілька пакетів відкидаються протягом тривалого періоду в результаті переповнення буфера. Коли з'єднання одночасно зменшують розмір вікна відправки, буфер буде порожній протягом певного інтервалу часу, що призводить до малого коефіцієнту використання каналу.

Механізм Drop Tail погано взаємодіє з інтерактивними мережевими додатками, тому що черги Drop Tail завжди заповнені або майже заповнені більшу частину часу, а пакети постійно відкидаються при досягненні ємності чергиі, що призводить до зниження продуктивності.

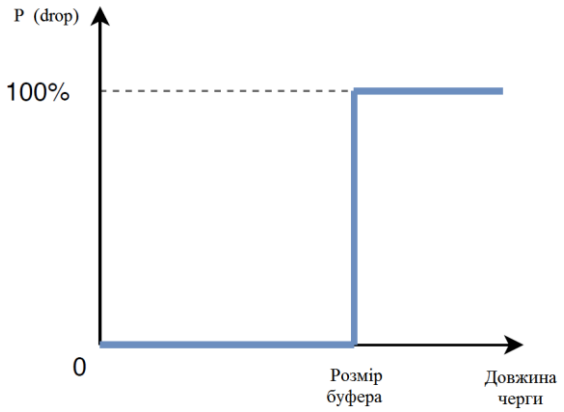

Рисунок 3.1 – Функція ймовірності відкидання пакетів в DT

Рисунок 3.2 – Використання смуги пропускання з DT

В даний час ступінь використання інтерактивних додатків через Інтернет, таких як голосовий та відеоконференцзв'язок, значно збільшилася. Ці додатки вимагають низької наскрізної затримки і мінімального джиттера. Коли буфер майже заповнений протягом тривалих періодів часу, затримка, звичайно, буде великою, що робить алгоритм DT таким, що не відповідає для цих додатків.

3.2.2 Drop Front

Алгоритм Drop Front аналогічний алгоритму Drop Tail, оскільки обидва механізму відкидають пакети, коли черга заповнена. Алгоритм Drop Front відкидає пакети з передньої частини черги при досягненні ємності буфера, в той час як Drop Tail відкидає пакети з хвоста черги.

Drop Front on full, відкидає пакет на початку черги, коли новий пакет прибуває в повну чергу. Цей алгоритм вирішує проблему блокування, і це єдина перевага Drop Front перед Drop Tail. З іншого боку, як і алгоритм Drop Tail, Drop Front страждає від тривалої затримки, глобальної синхронізації та інших проблем, викликаних переповненням черг.

3.2.3 Випадкове випадання RD

Алгоритм Random Drop відкидає пакети випадковим чином, коли буфер заповнений, в спробі звільнити місце для нових пакетів, що надходять. Алгоритм Random Drop представляє собою більш складний алгоритм порівняно з Drop Tail і Drop Front, так як цей алгоритм вимагає управління з'єднанням, оскільки ймовірність відкидання пакету з певного з'єднання пропорційна процентній частці цього з'єднання в буфері маршрутизатора. Як тільки пакет випадково відкидається, черга повинна зрушити пакети вперед, залишаючи місце в кінці черги для нового пакета, що прибуває, цей процес призводить до величезного обсягу обчислень. Зрештою, цей алгоритм може відкидати пакети, які деякий час вже перебували в черзі в очікуванні обслуговування, просто щоб звільнити місце для іншого нового пакета, що є очевидною тратою часу обробки.

Random Drop не має рівноправності в обслуговуванні пакетів, коли з'єднання, що не відповідають на перевантаження, зберігають високу швидкість відправки і монополізують чергу, залишаючи невелику частку смуги пропускання для повільних з'єднань.

3.3 Активне управління чергою (AQM)

Традиційне управління чергами страждає безліччю недоліків, які серйозно впливають на продуктивність мереж і в багатьох випадках не відповідають необхідному QoS. Активне управління чергою (AQM) було запропоновано для вирішення вищезазначених проблем, викликаних DT і іншими традиційними управліннями чергами.

Рішення проблеми повних черг, викликаної DT, полягає в тому, щоб відкидати пакети до того, як черга заповниться, і згодом відправники сповільнять свою швидкість передачі через це повідомлення про перевантаження, тим самим запобігаючи переповненню черги. Іншими словами, використовуючи механізми AQM, джерела заздалегідь інформуються про перевантаження і можуть відповідним чином реагувати. AQM – це проактивний підхід, який настійно рекомендується використовувати для проміжних пристроїв в Інтернеті.

Алгоритми AQM можна класифікувати наступним чином:

- зі стабілізацією черг маршрутизатора (ERD, RED, ARED, REM, PI, AVQ, SRED, Blue);
- з рівноправністю між потоками (FRED, CHOKe, Stabilized Random Early Drop (SRED), RED-PD;
- з управлінням «не відповідають» потоками (SFB);
- з високою пропускною здатністю (CoDel, PIE);
- з підвищеною продуктивністю для коротких з'єднань (FQ_CoDel).

Деякі алгоритми AQM можуть мати специфікації, що відносяться до різних категорій.

3.3.1 Стабілізація черг маршрутизатора

Основна мета AQM – подолати недоліки Drop Tail, які негативно позначаються на продуктивності мережі, а точніше замінити традиційне

управління чергами та запобігти переповнення черг. Крім цієї мети, багато алгоритми AQM також намагаються забезпечити QoS шляхом стабілізації черг маршрутизатора, щоб обмежити затримку постановки в чергу пакетів.

Стабілізація черг маршрутизаторів є величезною проблемою для алгоритмів AQM, оскільки досягнення стабілізації в ідеалі не повинно знижувати пропускну здатність каналу або збільшувати частоту втрати пакетів.

3.3.1.1 Раннє випадкове відкидання (ERD)

Алгоритм Early Random Drop (ERD) – один з алгоритмів AQM, який був розроблений для подолання збоїв DT. ERD використовує один поріг, і поведінка цього алгоритму залежать від фактичного розміру черги. ERD виявляє перевантаження, коли фактичний розмір черги досягає порогового значення, і ця інформація повертається до джерел шляхом відкидання всіх нових пакетів, що надходять, з фіксованою ймовірністю, поки фактичний розмір черги не впаде нижче порогового значення або рівня відкидання. Коли кількість пакетів в черзі досягає ємності черги, кожен новий пакет, що надходить, відкидається.

Алгоритм ERD забезпечує більш раннє повідомлення про перевантаження для користувачів і показує більш низьку ступінь глобальної синхронізації в порівнянні з традиційним управлінням чергами. ERD забезпечує справедливість серед користувачів мережі, постійно відстежуючи і виявляючи агресивні джерела.

Однак продуктивність ERD сильно залежить від відповідності порога відкидання і ймовірності відкидання до поточного розподілу мережевого трафіку.

Поліпшення алгоритму ERD для відповідності будь-якого мережевого трафіку може бути досягнуто шляхом реалізації динамічного регулювання порога відкидання і ймовірності відкидання пакету.

3.3.1.2 Random Early Detection (RED)

Випадкове раннє виявлення (RED) – це алгоритм, який привів до значного підвищення ефективності використання мережевих ресурсів. Хоча RED має серйозні проблеми з продуктивністю, через поліпшення продуктивності мережі RED рекомендований в якості стандартної схеми управління активною чергою в маршрутизаторах.

Випадкове раннє виявлення, судячи з назви, переслідує дві важливі мети. По-перше, RED намагається утримати чергу подалі від межі перевантаження, використовуючи деякі сигнали як індикатори перевантаження. По-друге, він відкидає пакети випадковим чином відповідно до функції ймовірності, яка збільшується зі збільшенням середнього розміру черги.

Така поведінка гарантує, що RED не налаштований проти будь-якого з'єднання або будь-якого переривчастого трафіку і не дає агресивним джерелам монополізувати чергу. Більш того, RED дає краще уявлення про перевантаження, оскільки відкидає пакети випадковим чином відповідно до середнього розміру черги, а не миттєвого розміру черги. Таким чином, якщо середній розмір черги великий, висока ймовірність постійного перевантаження.

Крім того, RED долає недоліки традиційного управління чергою, такі як усунення глобальної синхронізації, і вирішує проблеми з високою затримкою, контролюючи середню довжину черги.

Початковий RED класифікується як механізм раннього запобігання перевантаження для трафіку і розроблений для забезпечення сумісності з механізмом TCP для поліпшення можливостей TCP шляхом повідомлення про перевантаження на маршрутизаторі.

RED використовує переваги механізму TCP, відправляючи повідомлення випадковим джерел, щоб знизити швидкість відправки. Таким чином, RED уникає глобальної синхронізації всіх з'єднань одночасно,

випадковим чином відкидаючи пакети від випадкових з'єднань, щоб забезпечити справедливий розподіл смуги пропускання для запитів з'єднань.

Алгоритм RED включає чотири параметри для регулювання його роботи. Це такі параметри: мінімальний поріг (minth), максимальний поріг (maxth), максимальна ймовірність відкидання (maxp) і середня довжина черги (avg).

RED використовує довжину черги експоненціально зваженого ковзного середнього (EWMA) як індикатор перевантаження.

Розрахунок середнього виконується за допомогою фільтра нижніх частот з експоненціально зваженим ковзним середнім. На рисунку 3.3 показана залежність ймовірності відкидання/позначки пакета від розміру черги маршрутизатора.

Пакети відкидаються з лінійної ймовірністю, коли середня довжина черги знаходиться між двома граничними значеннями. Пакети відкидаються з ймовірністю, яка дорівнює одиниці, якщо середня довжина черги більше максимального порога. На рисунку 3.4 показана блок-схема алгоритму RED [19].

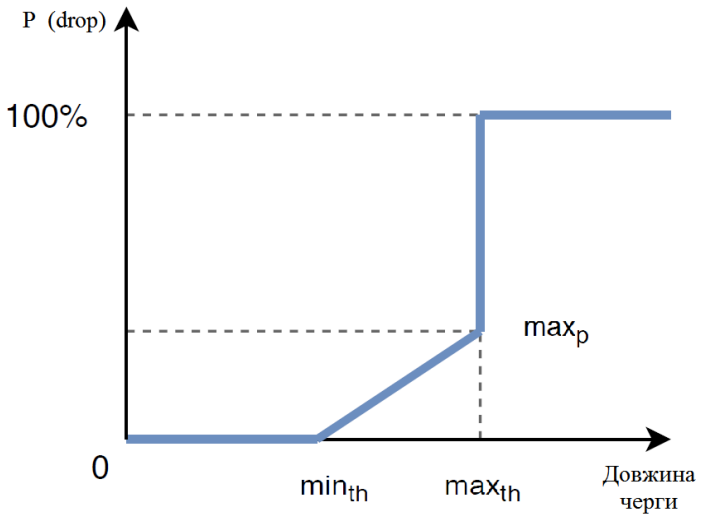

Рисунок 3.3 – Залежність ймовірності відкидання / позначки пакета від розміру черги

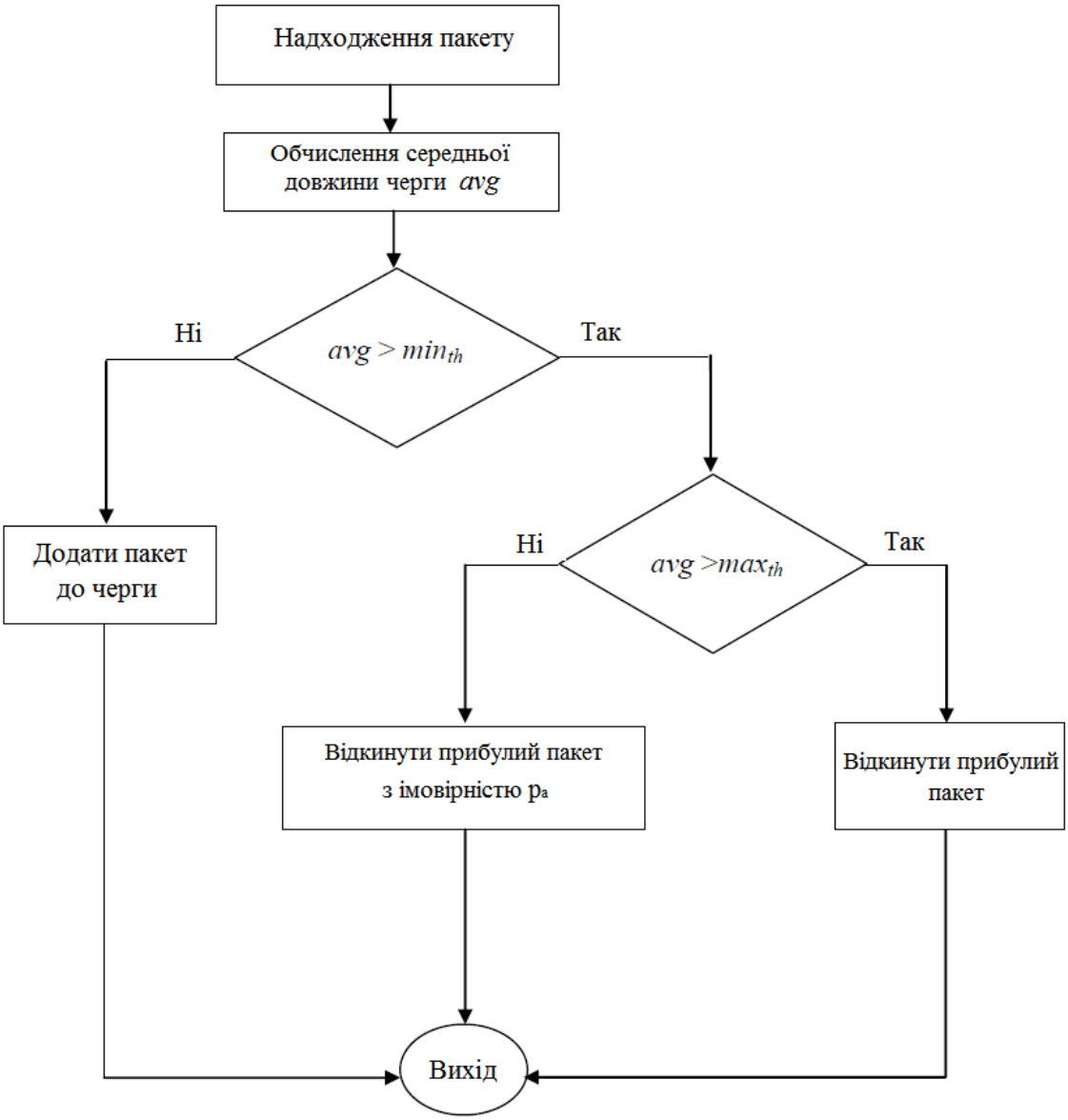

Рисунок 3.4 – Блок-схема алгоритму RED

Основними недоліками алгоритму RED є:

- відсутність значного покращення продуктивності чистого веб-трафіку та сумішей UDP;
- відсутність значного покращення продуктивності протоколу передачі файлів (FTP) та протоколу передачі гіпертексту (HTTP);
- компроміси між стабільністю та реакцією системи;
- труднощі з налаштуванням RED параметрів;
- несправедливість розподілу смуги пропускання.

Initialization:
$avg = 0$
$count = -1$
For each packet arrival
Calculate the new avg:
if the queue is nonempty
$avg = (1 - w_q)\, avg + w_q\, q$
else
$m = f(time - q\_time)$
$avg = (1 - w_q)^m\, avg$
if $min_{th} \le avg < max_{th}$
Increment $count$
Calculate probability p_a:
$p_b = max_p\, (avg - min_{th}) / (max_{th} - min_{th})$
$p_a = p_b / (1 - count \cdot p_b)$
with probability p_a:
mark the arriving packet
$count = 0$
else if $max_{th} < avg$
mark the arriving packet
$count = 0$
else $count = -1$
when queue becomes empty
$q\_time = time$

Saved Variables:
avg: average queue length
q_time: start of the queue idle time
$count$: packets since last marked packet

Fixed parameters:
w_q: queue weight
min_{th}: minimum threshold for queue
max_{th}: maximum threshold for queue
max_p: maximum value for p_b

Other:
p_a: current packet marking probability
q: current queue size
$time$: current time
$f(t)$: a linear function of the time t

Рисунок 3.5 – Псевдокод алгоритму RED

3.3.1.3 Адаптивний RED (ARED)

Однією зі слабких сторін RED є те, що середня довжина черги залежить від рівня перевантаження і налаштувань параметрів. Досягнення передбачуваних середніх затримок за допомогою RED вимагає постійної налаштування параметрів RED.

Крім того, RED не працює добре, коли середній розмір черги стає більше, ніж max_{th}, що призводить до значного зниження пропускної здатності і збільшення швидкості відкидання пакетів.

Adaptive RED зберігає базову структуру RED і регулює максимальну ймовірність скидання ma_{xp}, щоб середній розмір черги залишався між двома граничними значеннями min_{th} і max_{th}. Псевдокод Adaptive RED представлений на рисунку 3.6 [19].

For every *interval* seconds:

if (*avg* > *target* and $max_p \leq 0.5$)

increase max_p :

$max_p = max_p + \alpha$

else if (*avg* < *target* and $max_p \geq 0.01$)

decrease max_p :

$max_p = max_p \ \beta$

Variables:

avg: average queue length

Fixed parameters:

interval: time; 0.5 seconds

target: target for *avg* [min_{th} + 0.4 (max_{th} - min_{th}), min_{th} + 0.6 (max_{th} - min_{th})]

α: increment; min (0.01, max_p /4)

β: decrease factor; 0.9

Рисунок 3.6 – Псевдокод Adaptive RED

Таким чином, ARED знижує як частоту втрати пакетів, так і дисперсію затримки в черзі. Алгоритм ARED призначений для автоматичної установки w_q на основі швидкості з'єднання і адаптації max_p у відповідь на виміряну довжину черги.

Призначення адаптації max_p – зберегти середній розмір черги між min_{th} і max_{th} і зберегти середній розмір черги в межах цільового діапазону. max_p,

що було досягнуто з використанням політики адитивного збільшення і мультиплікативного зменшення (AIMD). Щоб уникнути зниження продуктивності алгоритму ARED, max_p слід обмежити в межах діапазону [0,01, 0,5].

3.3.1.4 Стабілізований RED (SRED)

Алгоритм SRED був запропонований для стабілізації роботи механізму RED. SRED упереджуючи відкидає пакети з ймовірністю, що залежить від навантаження, коли буфер переповнений.

SRED відкидає пакети в залежності від кількості активних потоків і миттєвої довжини черги замість обчислення середньої довжини черги. SRED допомагає стабілізувати рівень заповнення буфера, оцінюючи кількість активних з'єднань або потоків. Остаточна ймовірність відкидання пакету в SRED дається рівнянням (3.1).

$$P_{SRED} = \begin{cases} P(\text{q}) & \textit{для великої кількості акт. потоків} \\ \dfrac{P(\text{q})}{65536}(\textit{к-ть потоків})^2 & \textit{для малої кількості акт. потоків} \end{cases}. \quad (3.1)$$

Імовірність скидання пакета P(q) визначається рівнянням (3.2):

$$P(\text{q}) = \begin{cases} \max_p & \dfrac{B}{3} \le q < B \\ \dfrac{1}{4}\max_p & \dfrac{B}{6} \le q < \dfrac{B}{3} \\ 0 & 0 \le q < \dfrac{B}{6} \end{cases}, \quad (3.2)$$

де: max_p – це максимальна ймовірність скидання; B – ємність буфера, а q – миттєва довжина черги.

SRED долає проблеми масштабованості, пов'язані з деякими

алгоритмами AQM, тому що він не збирає і не аналізує інформацію про стан окремих потоків. У алгоритму SRED нормалізована пропускна здатність дуже низька навіть при невеликій кількості потоків трафіку.

3.3.1.5 Random Exponential Marking (REM)

Алгоритми управління потоком – це розподілені алгоритми для поділу мережевих ресурсів між конкуруючими джерелами. Вони часто складаються з двох підалгоритмів: алгоритму каналу, що виконується всередині мережі на маршрутизаторах, і алгоритму джерела, що виконується на граничних маршрутизаторах або хост-комп'ютерах.

Алгоритм REM складається з алгоритму зв'язку, який імовірно маркує пакети всередині мережі, і алгоритму джерела, який адаптує швидкість джерела до маркування, що спостерігається. Імовірність наскрізного маркування є експоненціальною, що дозволяє джерелу оцінити міру перевантаження шляху і таким чином, регулювати свою швидкість. Алгоритм REM не вимагає інформації про потік. На рисунку 3.7 показаний псевдокод алгоритму REM [25].

Алгоритм REM був запропонований для досягнення як високого коефіцієнта використання, так і незначних втрат і затримок. Його ключова ідея – використовувати змінну «вартість» в якості запобіжного перевантаження.

Вартісна змінна використовується для визначення ймовірності маркування. Він періодично оновлюється в залежності від невідповідності швидкості і невідповідності черзі. Невідповідність швидкості являє собою різницю між швидкістю приходу пакетів і пропускною спроможністю каналу, а невідповідність черзі представляє собою різницю між довжиною черги і цільовою довжиною черги.

Вартісна змінна збільшується, якщо зважена сума цих невідповідностей позитивна, і зменшується в іншому випадку. Зважена сума позитивна, коли

швидкість введення перевищує пропускну здатність каналу або коли є надмірне накопичення пакетів, яке необхідно прибрати, і негативна в іншому випадку.

Найбільш важлива відмінність між RED і REM полягає в тому, що REM відокремлює міру перевантаження від показника продуктивності, такого як довжина черги, затримка або втрата пакета.

Periodically

Update aggregate input rate:

$$in \leftarrow (1-\delta)\, in + (\delta)\, new\_in$$

Update marking probability m_l:

$$p_l \leftarrow \max \{p_l + \gamma(\alpha_l(buffer) + in - capacity), 0\}$$

$$m_l \leftarrow 1 - \emptyset^{-p_l}$$

End periodically

while buffer ≠0

Mark packet with probability m_l

End while

Saved variables:

in: aggregate input rate estimate

p_l: link price

m_l: current marking probability

Fixed parameters:

δ: weight in aggregate input rate estimation

γ: stepsize in price adjustment

α_l: weight of puffer in price adjustment

$\emptyset$: base in marking probability computation

Temporary variables:

new_in: current aggregate input rate

$buffer$: current buffer occupancy

$capacity$: current link capacity

Рисунок 3.7 – Псевдокод алгоритму REM

Ще одна відмінність між RED і REM полягає в тому, що RED має лінійну ймовірність маркування пакетів, в той час як REM має експонентну ймовірність маркування, як показано на рисунку 3.8.

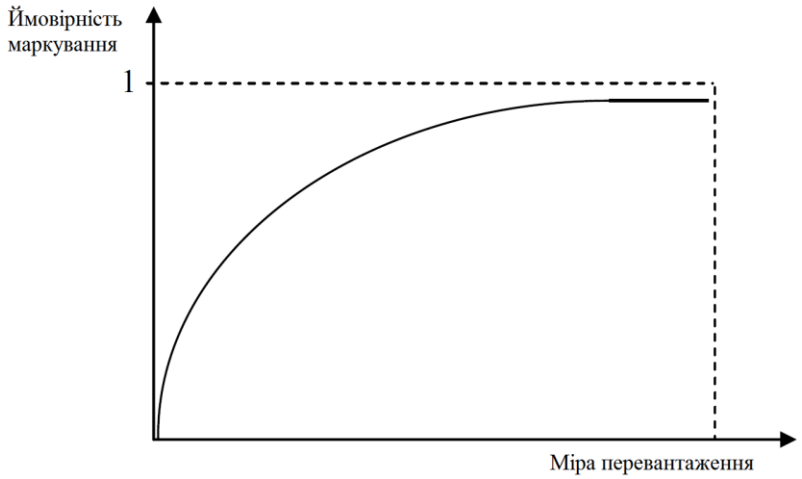

Рисунок 3.8 – Експоненціальна ймовірність маркування REM

REM дозволяє стабілізувати чергу маршрутизатора і зменшити втрату пакетів, але у нього є недолік – це складність настройки параметрів для забезпечення бажаної продуктивності.

3.3.1.6 BLUE

BLUE – це алгоритм AQM, який використовує втрату пакетів і коефіцієнт використання каналу для вимірювання перевантаження мережі і використовує ймовірність маркування або відкидання p_m. Якщо є переповнення буфера, BLUE збільшує p_m. Якщо черга стає порожньою, BLUE зменшує p_m. Це дозволяє BLUE визначити правильну швидкість, з якою він повинен відправляти повідомлення про перевантаження. BLUE використовує також два інших параметра, які контролюють p_m в часі.

Перший – freez_time, який визначає мінімальний інтервал часу між двома послідовними оновленнями p_m. Інші використовувані параметри – це d_1 і d_2. Вони визначають величину, на яку збільшується p_m при переповненні буфера або зменшується, коли черга порожня. Алгоритм BLUE показаний на рисунку 3.9. Відомо, що BLUE підтримує стабільність черг маршрутизатора і знижує рівень втрати пакетів [21].

```
Upon packet loss or buffer overflow:
if ( ( now - last_update) > freeze_time ) then
    pm= pm + d1
    last_update = now
Upon link idle:
if ( ( now – last_update) > freeze_time ) then
    pm= pm – d2
    last_update = now
```

Рисунок 3.9 – Псевдокод для алгоритму BLUE

3.3.2 Алгоритми з рівноправністю між потоками

Ці алгоритми відомі як алгоритми планування, і в міру обслуговування кожної черги пакети з цієї черги видаляються і плануються для передачі. Коли кількість потоків продовжує рости, важко підтримувати окремі черги пакетів і стан для кожного потоку. Сучасні алгоритми демонструють меншу складність за рахунок апроксимації справедливого розподілу смуги пропускання для потоків, але вони як і раніше вимагають певної кількості накладних витрат на реалізацію.

3.3.2.1 Flow Random Early Drop (FRED)

FRED був запропонований для вирішення проблеми рівноправності між TCP-з'єднаннями. FRED забезпечує вибіркове відкидання на основі врахування буфера для кожного активного потоку. У FRED коефіцієнт втрат розраховується з використанням середньої довжини черги для кожного потоку.

FRED зберігає свій стан тільки для потоків, для яких він має буферізовані пакети, а не для всіх потоків, що проходять через

маршрутизатор. Алгоритм FRED відрізняється від алгоритму RED тим, що виконує усереднення, як при прибутті, так і при відправці пакетів. FRED використовує функцію лінійного скидання, аналогічну RED. Таким чином, FRED можна розглядати як модифікацію RED, яка покращує справедливість, коли різні типи трафіку спільно використовують маршрутизатор. FRED забезпечує кращий захист для стрибкоподібних і низькошвидкісних потоків і більш ефективний при ізоляції потоків, які не відповідають.

3.3.2.2 CBT-RED

Буфери зі спільною пам'яттю забезпечують ефективне використання пам'яті і покращують продуктивність під час перевантаження, але у них є деякі технічні проблеми, такі як швидкість доступу та управління пам'яттю.

Алгоритм CBT-RED призначений для вирішення проблеми рівноправності, коли трафік TCP конкурує з трафіком UDP. Джерела UDP не відповідають на пакети, відкинуті RED, тому що трафік UDP не використовує схему запобігання перевантаження.

В результаті джерела UDP отримують більшу пропускну здатність, ніж джерела TCP. Це призводить до несправедливості між трафіком UDP і TCP. CBT-RED вирішує проблему рівноправності між трафіком TCP і UDP, встановлюючи пороги черги відповідно до типу трафіку і його пріоритету. Алгоритм встановлює поріг відкидання для трафіку UDP, який відрізняється від порога відкидання TCP. Це призводить до захисту TCP-трафіку від UDP-трафіку.

3.3.2.3 Збалансований BRED

BRED дозволяє вирішити проблему справедливого поділу смуги пропускання між адаптивними потоками (TCP) і неадаптивними потоками (UDP) на маршрутизаторах.

Алгоритм BRED превентивно відкидає пакети, штрафуючи неадаптивний трафік, який займає смугу більше, ніж його справедлива частка смуги пропускання. BRED регулює смугу пропускання потоку, виробляючи облік активних потоків буфера.

Рішення про відкиданні ґрунтується на зайнятості буфера потоку. BRED підтримує змінну q_len, яка є мірою кількості пакетів з потоку в буфері.

Буфер розділений на чотири сегмента, кожен з яких має різну ймовірність скидання. Рішення про відкиданні або прийомі пакету, що прибуває, ґрунтується на кількості пакетів з цього потоку, які вже існують в буфері. Якщо різні потоки мають різні розміри пакетів, алгоритм буде працювати в байтовому режимі, а не в пакетному. У порівнянні з іншими алгоритмами управління чергами, алгоритм забезпечує більш збалансований розподіл між різними потоками.

BRED дуже ефективний у забезпеченні справедливого поділу смуги пропускання між адаптивними і неадаптивними потоками. Алгоритм підтримує мінімальну інформацію про стан потоку і є масштабованим.

Хоча алгоритм може мінімізувати відмінності в смузі пропускання, одержуваної кожним потоком, він повинен підтримувати стани потоку, а це означає, що складність його реалізації пропорційна розміру буфера маршрутизатора.

3.3.2.4 Алгоритм CHOKe

В алгоритмі CHOKe, коли новий пакет прибуває в перевантажений маршрутизатор, обчислюється середньозважена черга, використовуючи фільтр нижніх частот, як це роблять RED і його варіанти. Алгоритм також має два порога min_{th} і max_{th}, аналогічні RED. Якщо середня черга менше min_{th}, пакети, що надходять, ставляться в чергу. Якщо середня черга перевищує max_{th}, CHOKe, як і RED, відкидає всі пакети, які надходять.

Коли середня черга знаходиться між min_{th} і max_{th}, CHOKe порівнює заголовок пакета, що надходить, з заголовком випадково обраного пакета в черзі.

Якщо обидва пакети належать одному потоку, обидва відкидаються. В іншому випадку новий пакет буде відкинутий з ймовірністю, яка залежить від рівня перевантаження. Ця ймовірність обчислюється точно так же, як в RED.

Алгоритм CHOKe – це простий алгоритм без збереження стану, який не вимагає якої-небудь спеціальної структури даних. Однак цей алгоритм навряд чи буде добре працювати, коли кількість потоків велика в порівнянні з буферним простором.

3.3.3 Алгоритм з керуванням потоками Stochastic Fair BLUE (SFB)

SFB забезпечує наближення рівноправності між потоками шляхом виявлення і обмеження швидкості потоків, які не відповідають та використовують велику частину пропускної здатності. SFB має бункери обліку L × N, основна мета бункерів обліку – підраховувати кількість пакетів різних потоків, що недавно надійшли на маршрутизатор.

В облікових осередках L представляє рівні, кожен рівень має N осередків. SFB має L незалежних хеш-функцій; кожна з цих хеш-функцій пов'язана з одним рівнем осередків обліку. L хеш-функції використовують IP-адреси і номери портів джерела і одержувача пакету для обчислення індексу осередку для цього пакета на кожному з L рівнів.

Коли новий пакет прибуває на маршрутизатор шлюзу, лічильники пакетів осередків, проіндексованих L хеш-функціями, збільшуються. Точно так же, коли пакет відправляється, лічильники пакетів осередків, індексованих L хеш-функціями, зменшуються.

Для кожного бункера підтримується і оновлюється ймовірність маркування або відкидання P, як в BLUE, в залежності від зайнятості черги

цього бункера. Якщо кількість пакетів в осередку перевищує певне значення (поріг), P для цього осередку збільшується. Якщо рівень чергиі осередку стає рівним нулю, P зменшується.

Потоки з високою пропускною спроможністю розпізнаються, коли ймовірність маркування або відкидання P стає рівною 1. Ці потоки з високою пропускною спроможністю потім змушені обмежувати свою швидкість. SFB дуже добре працює, коли кількість потоків з високою пропускною спроможністю невелика. З іншого боку, в разі занадто великої кількості потоків з високою пропускною здатністю, потоки з низькою пропускною здатністю, які хешіруются в ці бункери, неправильно ідентифікуються як такі, що використовують більшу частину пропускної здатності і караються.

3.3.4 Алгоритми з високою пропускною здатністю

3.3.4.1 CoDel

Рішення проблеми постійного заповнення буфера, bufferbloat, є основним завданням методів AQM.

Контрольована затримка (CoDel) [14] дозволяє керувати чергою без параметрів, забезпечуючи хороші результати в обробці затримок.

Алгоритм CoDel має такі властивості:

- не допускає зміни параметрів AQM операторами, користувачами або виконавцями;
- вміє розрізняти «хороші» і «погані» черги;
- контролює затримку пакета, не змінюючи затримки інших пакетів;
- адаптується до зміни швидкості з'єднання.
- має просту і ефективну реалізацію.

CoDel використовує час перебування пакета, як значення монітора. Воно визначається як час, протягом якого пакет знаходиться всередині буфера, іншими словами, це різниця між часом видалення з черги і часом

очікування пакета. Використання часу перебування пакета в якості значення монітора забезпечує кращу продуктивність, ніж використання довжини черги. Крім того, час перебування пакета використовується для розрізнення «хороших» і «поганих» черг, застосовуючи обробку відкидання/позначки тільки в «поганих» чергах.

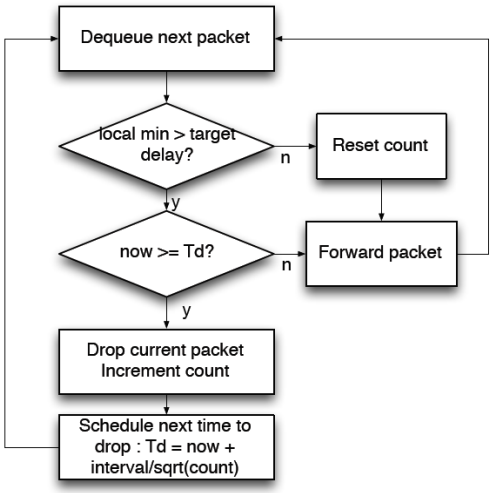

Рисунок 3.9 – Алгоритм CoDel

Для управління чергою CoDel використовує дві змінні: оцінку і мету. Оціночна функція використовує час перебування, для того, щоб встановити відмінності між «хорошою» і «поганою» чергою, яка зазвичай має максимальне RTT для всіх з'єднань. «Погані» черги – це ті черги, в яких CoDel запускає операцію відкидання або маркування, що дає іншим переваги щодо часу затримки.

З іншого боку, мета – мета задане значення, яке повідомляє CoDel, коли починати діяти. Якщо алгоритми починають передчасно відкидати пакети, то продуктивність знижується, що не є оптимальним втручанням в чергу пакетів.

Якщо коротко, алгоритм CoDel використовує час в черзі в якості параметра монітора. Коли мінімальний час перебування в черзі перевищує цільове значення всередині «поганої» черги, запускаються дії з видалення

або маркування пакетів. Цикл управління алгоритму буде намагатися перевести систему черг зі стану вище цільового в стан, в якому чергу знаходиться нижче цільового значення. Період відкидання пакетів буде змінюватися в залежності від того, скільки пакетів було раніше відкинуто, таким чином, корисна пропускна здатність максимізується.

CoDel – один з кращих методів AQM. Багато дослідників називають його використання кращим перед іншими рішеннями, однак у нього є деякі показники, які поки не оптимізовані.

3.3.4.2 Пропорційний інтегральний регулятор (PIE)

Налаштування RED є однією з основних проблем при його реалізації, з іншого боку, CoDel споживає надмірні ресурси обробки, що робить його дорогим і складним в експлуатації. Пропорційний інтегральний контролер Enhanced (PIE) об'єднує переваги обох: простоту реалізації, як в RED, і обробку затримки, як в CoDel.

Крім цього, PIE використовує випадкове відкидання/маркування пакетів, аналогічно RED, і виявлення перевантаження на основі затримки в черзі, як в CoDel (замість довжини черги). Крім того, управління затримкою засноване на тенденціях, збільшуючи або зменшуючи строгість контролю в залежності від рівня перевантаження. Основними перевагами PIE є:

- прямий контроль затримки замість довжини черги;
- досягнення високого ступеня використання каналу та низької затримки без втрати ефективності мережі;
- алгоритм простий і масштабований;
- алгоритм стабільно працює для різних мережевих топологій і ситуацій;
- параметри встановлюються автоматично.

Таким чином, PIE являє собою комбінацію RED і CoDel, що поєднує переваги і методи.

PIE складається з трьох компонентів: випадкового скидання, ймовірно періодичного скидання і оцінки швидкості виходу з черги. Імовірність відкидання і маркування застосовується, коли пакет потрапляє в маршрутизатор. Імовірність p розраховується на основі алгоритму PIE і застосовується до вхідного пакету. Ця ймовірність періодично оновлюється за такою процедурою: по-перше, обчислюється поточна затримка в черзі, по-друге, обчислюється ймовірність p і, нарешті, оновлюється попередня затримка.

Імовірність відкидання пакету розраховується відповідно до тенденції руху черги, з урахуванням тенденції зміни черги і затримки. Для цього використовуються параметри α і β, α визначає відхилення поточної затримки, а β є додатковим налаштуванням, що залежить від того, зростає або зменшується затримка.

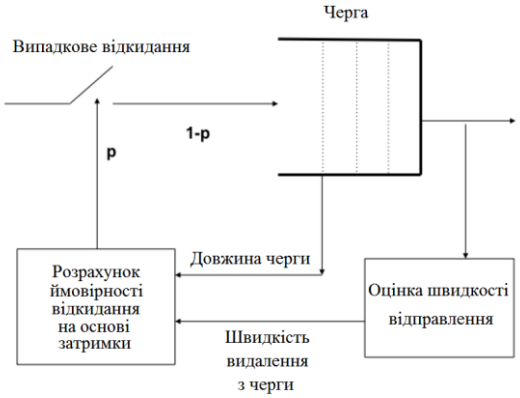

Рисунок 3.10 – Схема роботи PIE

Оцінка швидкості відправлення пакетів ще один момент для аналізу в PIE. Вона пов'язана з флуктуаціями доступності каналу і пропускної здатності. Швидкість відправлення розраховується, коли довжина черги перевищує порогове значення, і виходить значення вимірювання. Воно використовується для поновлення частоти відправлень і скидання лічильників, що дозволяє оновлювати ймовірність p першої точки.

Як показано на рисунку 3.10, «Розрахунок ймовірності відкидання на

основі затримки» виконується з використанням значення довжини черги і швидкості відправки пакетів з черги. Плюс до цього через періодичне поновлення допускаються сплески р. Будь-пакет, що прийшов в цей період часу, буде оброблятися без відкидання пакетів, не змінюючи ймовірність відкидання.

На рисунку 3.11 показано порівняння роботи алгоритмів RED і PIE. Результати моделювання взяті з [28], де проводяться порівняння характеристик PIE по відношенню до інших алгоритмам AQM.

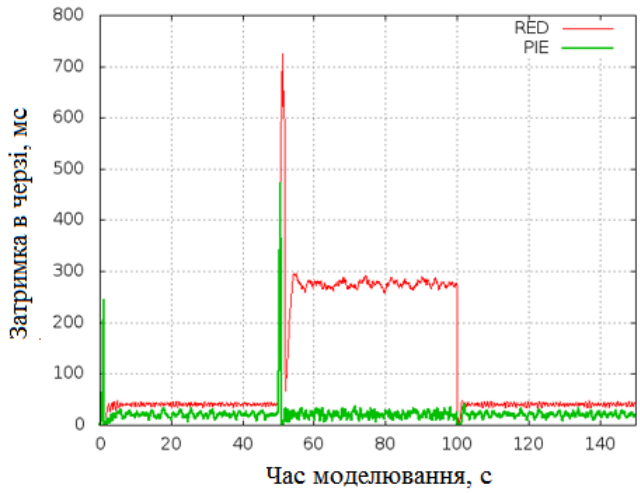

Рисунок 3.11 – Поведінка RED і PIE при різної пропускної здатності

На рисунку 3.11 показана реакція RED і PIE на зниження пропускної здатності каналу. Від 0 до 50 с пропускна здатність каналу становить 100 Мбіт/с, потім з 50 до 100 с пропускна здатність зменшується до 20 Мбіт/с і, нарешті, з 100 до 150 знову збільшується до 100 Мбіт/с. RED контролює довжину черги, збільшуючи затримку в черзі при зміні сценарію. Проте, PIE заснований на затримці в черзі, швидко змінюючись в нових умовах і пристосовуючись до рівня затримки 20 мс.

На рисунку 3.12 показано порівняння ефективності RED і PIE при зміні трафіку. З 0 по 50 з трафік складається з 10 потоків TCP, потім з 50 по 100 с число потоків TCP збільшується до 30, після цього з 100 по 150 с знову

збільшується до 50 потоків, і з 150 по 250 с зменшується до 30 потоків TCP. RED працює з налаштованими параметрами, збільшуючи затримку в черзі при збільшенні навантаження. З іншого боку, PIE контролює затримку, автоматично адаптуючи параметри.

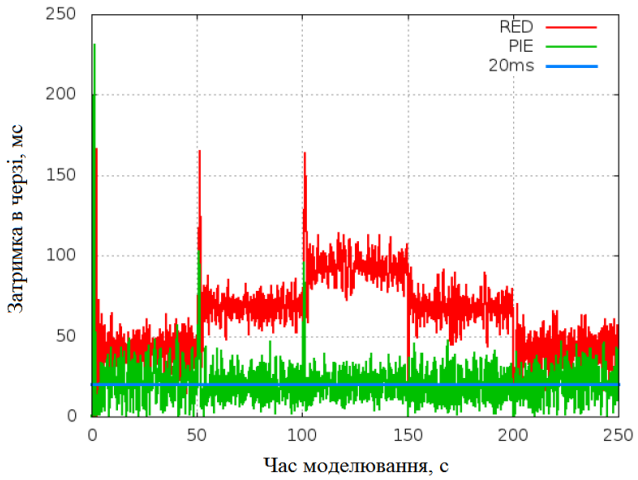

Рисунок 3.12 – Ефективність RED і PIE при зміні трафіку [28]

PIE забезпечує контроль над проблемою буферного заповнення, використовуючи випадкові скиди пакетів відповідно до тенденції зміни розміру черги. Він використовує автоматичну настройку для параметрів, що діють в різних сценаріях.

Однак PIE генерує великі сплески і затримки в черзі в сценаріях з великими значеннями RTT. Ця проблема пов'язана з використанням пропорційного інтегрального контролера, який підтримує середнє значення чергиі на бажаному рівні. Таким чином, ймовірність скидання пакету збільшується, а коефіцієнт використання каналу зменшується.

4 РОЗРОБКА МЕТОДА АКТИВНОГО КЕРУВАННЯ ЧЕРГОЮ

4.1 Алгоритм FQ_CoDel

CoDel передбачає успішний контроль в широкому діапазоні пропускної здатності і числа потоків. Крім того, він скорочує сплески затримки. Однак при дуже великій кількості потоків виникають проблеми. Створення безлічі черг для обробки перевантаження дає можливість оптимізувати «розріджені» і «мишачі» потоки TCP. Таке рішення надається алгоритмом FQ_Codel, який може генерувати до 64 КБ черг, незалежно застосовуючи механізм CoDel в кожній черзі. «Мурашиний» потік – це короткі, зазвичай поодинокі, керуючі пакети, такі як SYN/ACK, DNS, DCHP або NTP. Крім них, деякі потокові, ігрові або VoIP-пакети теж мають «мурашині» характеристики. Таким чином, Інтернет складається з потоків «слонів», «мишей» і «мурашок». Потоки «мишей» і «слонів» складають основу трафіку. Проте, «мурахи» – це важливі пакети, які слід доставляти якомога швидше з певним пріоритетом.

Отже, FQ_CoDel являє собою комбінацію CoDel і схеми планування декількох черг, яка включає пріоритезацію і ізоляцію потоку. Він створює одну підчергу для кожного потоку, і кожна підчерга використовує CoDel для оптимального управління.

Основні властивості алгоритму FQ_CoDel:

- пріоритезація і ізоляція потоку для трафіку;
- розглядаються розріджені потоки і невеликі пакети, критичні за часом;
- запобігає виснаження потоку.

Механізмів відкидання пакетів AQM може виявитися недостатніми для задоволення суворих вимог до затримки. Для цього алгоритми планування можуть ізолювати трафік для кожного потоку або класу, гарантуючи певні обмеження.

Алгоритм FQ_CoDel використовує планувальник циклічного перебору (DRR) для циклічного перебору підчерг.

FQ_CoDel генерує вкладені черги відповідно до параметрів потоку вхідних пакетів і заснований на трьох списках вкладених черг: нових, старих і порожніх. Спочатку все підчерги знаходяться в режимі порожнього стану, але як тільки перший пакет потрапляє в буфер, алгоритм переводить підчерги в новий статус. Кожна підчерга має певну кількість байтів для вилучення пакетів з черги, коли вона використовується, черга стає «старою».

З іншого боку, CoDel використовується як модель AQM для використання всередині підчерг. FQ_Codel – це розширення алгоритму CoDel, яке пропонує рішення з декількома чергами для забезпечення низької затримки і забезпечення якості обслуговування для конкретного пакунка.

Множинна черга забезпечує ізоляцію окремої черги в обмін на використання обчислювальних ресурсів маршрутизатора. Однак, FQ_Codel забезпечує кращі умови для додатків, чутливих до затримок.

4.2 Alternative Backoff (ABE)

В раніше проаналізованих моделях повідомлення про перевантаження могло здійснюватися шляхом відкидання або маркування пакета.

До сих пір використовувався варіант відкидання, проте використання маркування пакета може помітно знизити затримку потоків. Зокрема, включення явного повідомлення про перевантаження (ECN) забезпечує зворотний зв'язок шляхом додавання нової інформації в заголовки протоколу IP. Таким чином, інформація в доставлених пакетах може бути використана для повідомлення відправника про зміну перевантажувального вікна.

Альтернативний відкат (ABE) з ECN полягає в маркуванні пакетів, що дозволяє окремим відправникам TCP використовувати більший коефіцієнт мультиплікативного зменшення. Таким чином, це модифікація на стороні відправника, можливість поступово розгортати її, просуваючи використання

ECN на вузлах Інтернету. Згідно зі специфікацією ECN, повідомлення ECN слід розглядати як сигнал перевантаження в TCP, який не підтримує ECN. В цьому випадку вікна перевантаження зменшаться вдвічі і запускається алгоритм повільного старту. Для забезпечення малих затримок і високої пропускної здатності пропонується наступне:

- втрата пакетів: TCP повинен зменшувати cwnd вікна перевантаження як зазвичай (NewReno зменшує на 50% і CUBIC на 30%);
- мітка ECN: відправник TCP зменшує cwnd менше, ніж при втраті пакету.

Отже, на стороні відправника зміниться розмір перевантажувального вікна в залежності від значення, що є коефіцієнтом, який зменшується. Фактично, будуть існувати два значення: одне, створене повідомленням ECN, і інше, створене через втрату пакета. Альтернативний відкат з ECN забезпечує оптимізацію управління чергою на маршрутизаторах.

Однак у ABE є велика проблема. Для забезпечення ефективності вирішальне значення має наявність пристроїв з підтримкою ECN в відправника, одержувача і вузлах AQM між ними. Якщо вузол не підтримує ECN, ABE не братиме участь в скороченні затримки.

З іншого боку, слід застосовувати традиційні міркування безпеки ECN, незважаючи на те, що не змінюються повідомлення транспортного протоколу для обміну.

ABE – гарне удосконалення всередині методів AQM, що використовує для цього бік відправника. Використання ECN-мітки забезпечує необхідні переваги, покращуючи результати використання мережевих ресурсів.

4.3 Порівняння алгоритмів

Цілями моделей AQM є зменшення затримки в черзі, мінімізація відкидання пакетів і збільшення пропускної спроможності до максимально можливої.

Першою точкою порівняння буде кумулятивна ймовірність затримки в онлайн-іграх, які використовують мережу спільно з іншими типами додатків (FTP, Інтернет і VoIP).

Серед додатків, чутливих до затримки, онлайн-ігри мають величезний вплив в мережі. В цілому онлайн-ігри вимагають низьких затримок і низької втрати пакетів, проте між ними є відмінності.

В цілому, онлайн-ігри вимагають швидкої відповіді мережі, для цього моделі AQM протестовані в декількох сценаріях (низький, середній і високий трафік) [29].

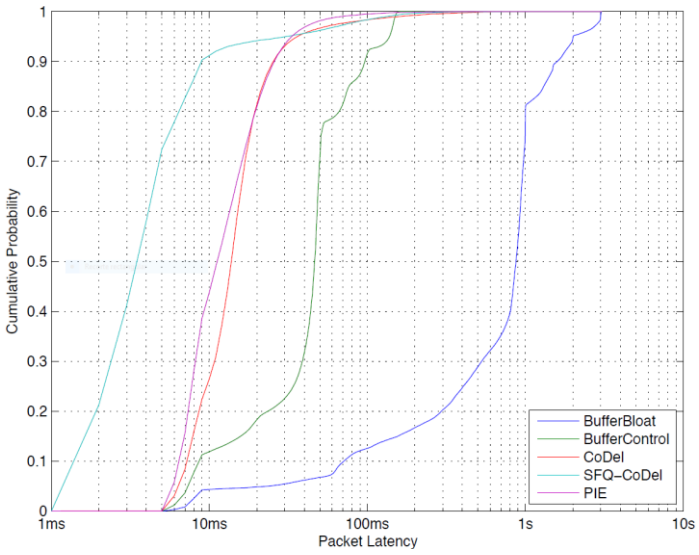

Рисунок 4.1 – Затримки для різних AQM

Кумулятивна ймовірність показує ймовірність отримання певної або меншої затримки пакета, наприклад, Bufferbloat (проблема зайвої мережевої буферизації при буферах великої місткості) має 10% ймовірності затримки пакета 100 мс (або менше). На графіку показані моделі і їх оцінки, які є традиційним Drop Tail Bufferbloat і Buffer-Control методами, але з метою оцінки затримки буферизації 50 мс.

Якщо подивитися на рисунок 4.1, обидва методи Drop Tail відчувають найгірші затримки. З іншого боку, методи AQM забезпечують найкращі

результати, оскільки на них менше впливають сценарії перевантаження. В деталях, CoDel і FQ_CoDel вимірюють затримку безпосередньо (незалежно від вихідної швидкості передачі даних), а PIE прогнозує затримку на основі недавньої історії вихідної швидкості передачі даних, краще адаптуючись до перевантаження. FQ_CoDel перевершує всі інші методи. FQ_CoDel може забезпечити малу затримку в черзі для невеликих пакетів FTP. Однак при великих потоках пакетів продуктивність помітно знижується.

Таким чином, FQ_CoDel є активним методом управління чергою, який найкращим чином забезпечує досягнення порогових значень як затримки, так і втрат, навіть в ситуаціях з інтенсивним трафіком [29].

4.4 Розробка методу управління чергою маршрутизатора

Пропоноване рішення засноване з урахуванням компонентів, які складають набір алгоритмів FQ_CoDel і ABE.

Ефективність алгоритму RED залежить від правильного налаштування чотирьох параметрів (minthreshold, maxthreshold, wq, maxp), які варіюються в залежності від ситуації в мережі. Установка відповідних значень є критичною проблемою, яка обмежує реалізацію алгоритму в реальній мережі. Контрольована затримка (CoDel) дозволяє уникнути використання «настроюваних» параметрів, використовуваних для прогнозування часу перебування пакета при перевантаженні, замість середнього розміру черги. Крім того, він легко адаптується до кількох мережевим сценаріям через незалежность таких параметрів (розмір черги, середній розмір черги, частота відкидання, використання каналу). Всі ці функції роблять CoDel одним з найцінніших алгоритмів для управління чергою.

Основна ідея CoDel полягає в тому, що, управління чергою засноване на вимірі кількості часу, який пакет проводить всередині черги, будучи часом з моменту надходження пакету до часу, коли пакет пересилається або відкидається.

Одна з цілей CoDel – розрізняти «хороші» і «погані» черги, щоб позбутися від «поганих» черг. Звичайна проблема TCP полягає в тому, що відправник зосереджується на зменшенні розміру перевантажувального вікна, не беручи до уваги пакети, що знаходяться в транзиті. З одного боку, вузькі місця, які створюють чергу, яка вносить затримку більше одного RTT, відомі як «погані» черги. З іншого боку, черги, які обробляють пакети менш ніж за одне кругове звернення, є «хорошими» чергами. Таким чином, необхідно виявляти «погані» черги і контролювати їх.

CoDel алгоритм можна розділити на три компоненти: блок оцінки, блок цільової уставки і контур управління. Блок оцінки з'ясовує ситуацію з чергою, блок цільової уставки встановлює, що потрібно черзі, а контур управління вживає заходів для переходу до точки стабілізації черги.

Блок оцінки – це компонент, який займається виявленням переходу черги від «хорошої» до «поганої». Довжина черги – це звичайний індикатор метода AQM, проте його складно швидко адаптувати до змін. Крім того, іноді при відкиданні пакетів виникають сплески, які необхідно ігнорувати через сплески збільшення довжини черги. Компонент має бути зосереджений на величині надлишкової затримки, яку відчуває пакет через постійну чергу, іншими словами, на часі перебування.

Методи AQM орієнтовані на зменшення затримки черги, тому розумно використовувати час перебування в якості контрольованої змінної.

Як тільки час перебування встановлюється в якості контрольного значення, потрібно відокремити «хороші» черги від «поганих». Зазвичай в якості індикатора використовується середнє значення, що представляє собою середнє значення половини довжини черги, проте воно залежить від часу, коли обчислюється середнє значення.

CoDel використовує як індикатор мінімальний час перебування, що є найкращим варіантом. Якщо в черзі знаходиться тільки один пакет, час перебування дорівнює нулю. Таким чином, індикатором алгоритму CoDel є мінімальний час перебування.

Ще один момент для визначення – це інтервал часу поновлення індикатора. Це повинно бути між максимальним часом, коли діє мінімальна затримка, і часом виснаження найбільшого сплеску в черзі. Знову ж правильним значенням є максимальне RTT всіх з'єднань.

Таким чином, блок оцінювача використовує час перебування в якості спостережуваного значення, а локальний мінімум часу перебування в якості статичного для моніторингу. Він забезпечує надійне і точне вимірювання черги, зменшуючись тільки при виключенні пакета з черги, іншими словами, CoDel відзначає тільки момент видалення з черги, уникаючи блокувань пакетів.

На даний момент відомо значення для відстеження перевантаження, тим не менше, потрібно встановити точку, яка повідомляє, коли алгоритм починає діяти. Якщо ця точка викликає ранній акт відкидання CoDel, коли оцінка не дорівнює нулю, швидкість скидання збільшиться, але від цього серйозно постраждає корисна пропускна здатність.

Отже, цільова установка – це максимальне використання пропускної здатності при мінімізації затримки. Для нього, як це пояснюється в [28], потужність буде значенням, яке відображає залежність між номінальною пропускною спроможністю і номінальною затримкою.

RFC8289 визначає, що цільове значення установка має становити 5% від номінального RTT. Це значення було оцінено в сценаріях Reno, Cubic і Westwood, і воно працює правильно. Таким чином, з огляду на 100 мс RTT, цільова уставка повинна бути 5 мс. Тобто, коли локальний мінімальний час перебування вище цільової уставки (5 мс), алгоритм CoDel почне діяти.

Робота з теорією управління в чергах – складна робота, тому що вони не є лінійною системою, а AQM працює з максимальною нелінійністю. Крім того, через поєднання потоків вони не залежать від часу і залежать від протоколу, використовуваного для дзвінків (наприклад, TCP або UDP). Навіть в цьому випадку класична поведінка контролерів несумісна з визначенням стійкості черги.

При цьому необхідний контроль, який переводить постійні черги, які знаходяться вище цільового значення, в черзі нижче цільового значення. Рішення полягає в відкиданні (маркування) пакетів, однак відкидання з високою швидкістю надає поганий вплив на пропускну здатність. Таким чином, слід визначити відповідний період часу між скидами.

Кращий метод – почати з низької швидкості відкидання і повільно збільшувати, поки постійна черга не стане менше цільової точки. Таким чином уникає надмірне відкидання і в кінцевому підсумку розсіюється постійна черга.

Час між скидами має бути встановлено точно, контролер повинен визначити його до наступного скидання, інакше CoDel скине зайві пакети. Мінімальний час, протягом якого контролер повинен побачити ефект – це час RTT плюс час очікування в локальній черзі. Хоча місцевий час очікування має тенденцію змінюватися, тому відповідний час буде трохи більше, ніж RTT, що є зручною мірою інтервалу оцінки. Інтервал гарантує, що контролер врахував прийнятну затримку перевантаження, уникаючи надмірних скидів пакетів.

Крім того, часовий інтервал повинен повільно зменшуватися, щоб вийти зі стану постійної черги. Для цього час інтервалу буде зменшено відповідно до лічильників скидання:

$$NextDropTime = \frac{INTERVAL}{\sqrt{\text{drop\_count}}}. \tag{4.1}$$

CoDel необхідно контролювати спустошення черги, щоб припинити відкидання пакетів. Для цього він перевіряє, чи залишилося в буфері хоча б MTU байтів. Якщо немає, то пакет не відкидається, і CoDel виходить зі стану відкидання. Розмір MTU відповідає найбільшому пакету, який він бачив.

CoDel є одним з методів AQM, який працює проти зайвої мережевої буферизації, які мають дуже хорошу продуктивність з низькою затримкою, і

з простою реалізацією. Потрібно лише встановити значення INTERVAL в 100 мс для використання в Інтернеті. З іншого боку, цільова установка дає оптимізацію значення потужності (максимальне використання каналу з мінімальною затримкою). Її ідеальне значення складає 5-10% від RTT з'єднання, тому для Internet цільова установка буде 5 мс.

Таким чином, процедура CoDel наступна: коли блок оцінювання виявляє затримку вище цільової установки, алгоритм переходить в стан скидання і встановлює скидання через час рівне значенню INTERVAL. Після цього алгоритм розраховує час наступного скидання відповідно до кількості відкинутих пакетів, поки затримка не стане нижче цільової.

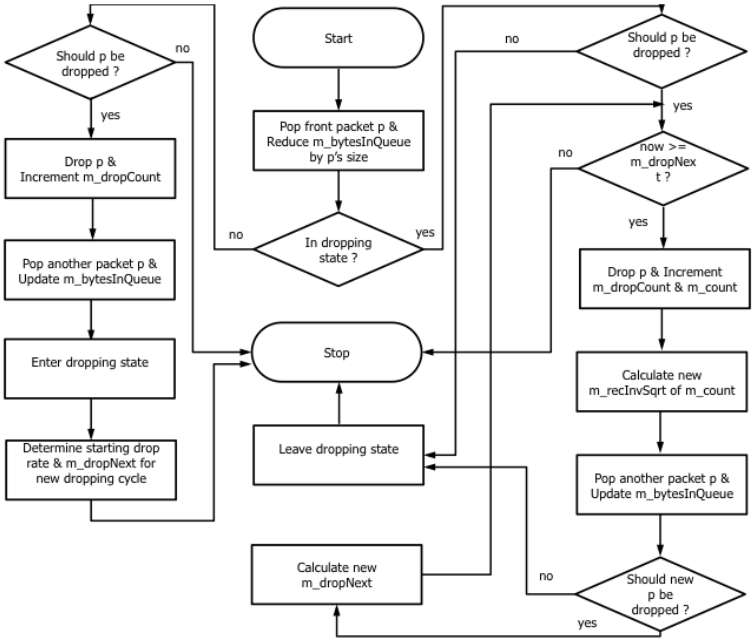

Рисунок 4.2 – Алгоритм CoDel

4.4.1 Детальний розгляд алгоритму FQ_CoDel

FQ_CoDel – це алгоритм, здатний керувати більш ніж однією чергою, застосовуючи CoDel в кожній з них. Він працює з декількома потоками, зменшуючи вплив проблеми зайвої мережевої буферизації і забезпечуючи

ізоляцію потоків низького трафіку. Фактично, управління чергою FQ_CoDel відбувається завдяки комбінації циклічного перебору (DRR) і CoDel.

Алгоритм підтримує таку термінологію:

- потік: визначається як група пакетів з однією і тією ж вихідою IP-адресою і місцем призначення;
- кортеж: IP-адреса, номер порту джерела, номер порту призначення і номер протоколу;
- FQ_CoDel хеширує потоки в різні черги для управління ними;
- квант: максимальна кількість байтів, яка може бути видалена з черги за один раз (за замовчуванням встановлено 1514 байтів, MTU Ethernet плюс довжина апаратного заголовка 14 байтів).

Алгоритм стохастично класифікує вхідні пакети і розміщує в різних чергах, потім планувальник вибирає чергу для видалення пакетів з черги.

4.4.2 Організація черги потоків

Кожному потоку, що надходить у вузол, потрібна власна чергу, яка є терміном «організація черги потоку». За замовчуванням створюється 1 024 черги, кожній з яких присвоюється один хеш.

Цей хеш створюється комбінацією випадкового числа (обраного за ініціалізації алгоритму) з хешем з 5 кортежів (IP-адреса, IP-адреса джерела, порт призначення, вихідний порт і протокол). Усередині кожної черги пакети розташовані в порядку FIFO, а реалізована модель AQM – CoDel.

FQ_CoDel вибирає чергу для застосування AQM за допомогою механізму дефіцитного циклічного перебору (DRR), заснованого на байтах.

Кожна чергу має кількість байтів-кредитів, ініціалізованих в «квантовій» змінної (ступінь деталізації видалення з черги). Як тільки у черзі з'являється шанс вилучення пакета з черги, вона зменшує кількість кредитів на розмір кожного пакета, і буде робити це до тих пір, поки значення кредитів не стане рівним нулю.

Після цього покажчик збільшується на один такт, і можливість вилучення з черги припиняється.

DRR застосовує справедливість на основі байтів. В деталях DRR діє в двох наборах черг: «нова» і «стара».

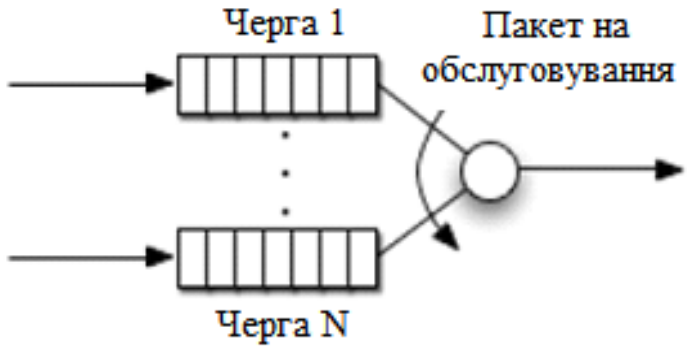

Рисунок 4.4 – Робота алгоритму Round Robin

4.4.3 Планувальник

Як тільки пакет прибуває в маршрутизатор, і не належить до іншого сегменту черги, він додається в нову чергу і стає активним і фіксується час постановки його в чергу. Фактично процедура постановки в чергу ділиться на три етапи: класифікацію, отримання тимчасових міток і необов'язкове видалення.

Коли пакет прибуває, його необхідно класифікувати в сегменті черги. Для цього розраховується хеш пакета IP з 5 кортежів із значенням за модулем кількості черг. У хеш додається випадкове число (обирається під час ініціалізації) для запобігання DoS-атак. В цей час пакет класифікується до відповідної черги і запускає час CoDel і тимчасову мітку. Пакет, відповідно до порядку FIFO, буде поміщений в кінці обраної черги, а чергу буде позиціонуватися як остання серед нових черг з квантовим числом кредитів. Крім того, лічильник байтів в черзі буде оновлюватися відповідно до розміру пакета.

Кожна черга FIFO відстежується двома змінними стану, кредитами черги і загальною кількістю байтів в черзі, крім змінних стану CoDel. Крім того, можуть бути включені інші змінні, наприклад, підрахунок кількості відкинутих пакетів. Загальна кількість байтів в черзі використовується для обмеження кількості пакетів у всіх чергах.

В алгоритмі передбачено механізм захисту від перевантаження. У разі, якщо загальна кількість поставлених в чергу пакетів перевищує ліміт, вибирається черга з найбільшою кількістю байтів і половина пакетів відкидається. Використання лічильника байтів для вибору черги знижує завантаження ЦП, оскільки немає необхідності знаходити найдовшу чергу.

Спочатку вибирається черга для отримання пакету, потім витягується пакет і, нарешті, обчислюються деякі значення.

Спочатку FQ_CoDel переглядає список нових черг для вибору черги вилучення пакета. Читання починається з голови списку і якщо черга має номер негативного кредиту, їй дається додатковий обсяг кредитів і вона переміщається в кінець списку старих черг. Якщо в черзі є позитивна кількість кредитів, вона вибрається для вилучення пакета.

Якщо список нових черг порожній, планувальник запускає ту ж процедуру в списку старих черг.

Після вибору черги запускається алгоритм CoDel. Якщо пакети необхідно відкинути або позначити CoDel вибирає їх з голови черги, а потім обрані пакети витягуються з черги.

Якщо жоден пакет не може бути витягнутий з черги, значить черга порожня, тому:

- кщо це нова черга, вона переміщається в список старих черг;
- якщо це стара чергу, вона буде переміщена в список порожніх черг (до приходу нового пакета).

Процес видалення з черги складається з повернення пакета від механізму CoDel до планувальника. Коли це відбувається, планувальник віднімає розмір пакета з кредитів черги.

Нарешті, переміщення порожніх нових черг в список старих черг запобігає їх дефіцит. Порожня нова черга може з'явитися знову (з надходженням нового пакету) до того, як будуть відвідані старі черги; таким чином, він обслуговує всі старі черги до того, як порожня черга буде видалена.

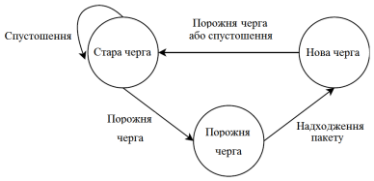

Рисунок 4.5 – Модель FQ_CoDel

4.4.4 Відмінності від CoDel

Як було пояснено, кожна черга підтримує змінні стану для FQ_Codel, тому, якщо активна тільки одна черга, FQ_Codel діє ідентично CoDel. Однак між обома двома алгоритмами існують деякі відмінності.

Пошук точки сходження відкидання (тієї, яка мінімізує затримку і максимізує пропускну здатність), вимагає певного часу для алгоритмів. Алгоритм FQ_CoDel вимагає обробки конкуруючих потоків і розгляду старих і нових черг. Таким чином, для досягнення ідеальної швидкості витягання потрібно більше часу. І навпаки, ізоляція потоків в FQ_CoDel забезпечує більш точне відкидання пакетів з найбільших потоків.

4.4.5 ABE: явне повідомлення про перевантаження з ECN

Маркування пакетів за допомогою явного повідомлення про перевантаження (ECN) забезпечує зворотний зв'язок про перевантаження за допомогою пакетів і дозволяє уникнути втрати пакетів в ситуаціях перевантаження.

Явне повідомлення з ECN (ABE) пропонує використовувати маркування ECN з комбінацією методів AQM, яка забезпечує низьку затримку і високу пропускну здатність. Таким чином, ABE намагається підвищити продуктивність мережі за допомогою регулювання швидкості на стороні відправника, і може бути розгорнута з використанням ECN і можливістю працювати з потоками, що не підтримують ECN.

Пропозиція ABE полягає в зменшенні вікна перевантаження в залежності від типу повідомлення.

AIMD або аддитивне збільшення, мультиплікативне зменшення (AIMD) призводить до зменшення cwnd до β=0,5 через мультиплікативного стилю роботи AIMD. Однак це значення зниження розміру вікна не використовує оптимальним чином пропускну здатність каналу, як показано на рисунку 4.6.

З іншого боку, використання AQM паралельно з механізмами TCP забезпечує кращу продуктивність каналу.

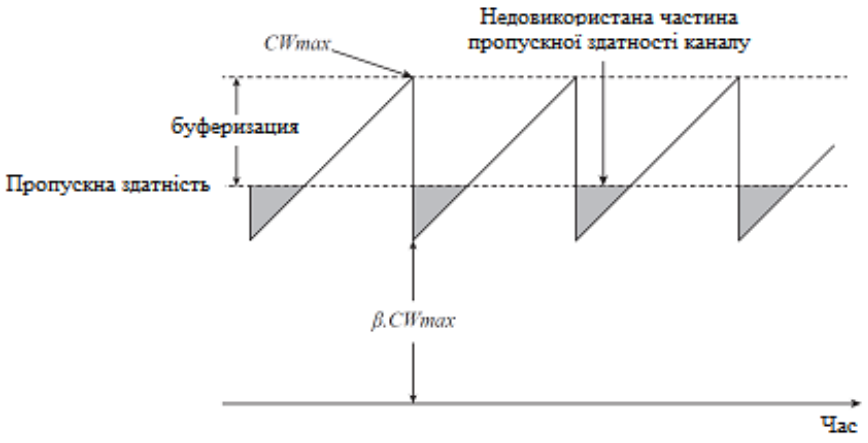

Рисунок 4.6 – Проблема «пили» TCP

ABE передбачає нове значення $\beta_{ECN} > \beta$ у відповідь на мітки ECN, але зниження розмірів cwnd залишається. Це суперечить офіційній рекомендації ECN про те, що мітки ECN обробляються як повідомлення про перевантаження, проте передбачається, що ABE збільшує оптимальне

використання затримки. Отже, поведінка AQM передбачає, що мітка ECN створюється моделлю AQM, що має перевантаження черги.

Беручи до уваги, що чим більше значення β_{ECN}, тим вища продуктивність, критично важливо вибрати значення, яке максимізує пропускну здатність і мінімізує затримку. Таким чином, необхідний баланс між продуктивністю і швидкістю розвантаження вузьких місць. Не існує ідеального значення для всіх сценаріїв, тим не менш, рекомендується значення β_{ECN} від 0,7 до 0,85 як ідеальне, особливо після періоду повільного старту.

Одна з найсильніших сторін ABE полягає в тому, що модифікація знаходиться на стороні відправника, отже, вона не вимагає змін з боку одержувача TCP або мережевих вузлів.

5 ВИБІР ІНСТРУМЕНТАЛЬНИХ ЗАСОБІВ І МОДЕЛЮВАННЯ

5.1 Моделювання

Для перевірки теоретичних викладок, наведених в атестаційної роботі проведено імітаційне моделювання в ns3. Network Simulator 3 – це симулятор мережі з дискретними подіями для Інтернет-систем, призначений в першу чергу для дослідницьких і освітніх цілей. Ns-3 – безкоштовне програмне забезпечення, яке підлягають ліцензуванню за ліцензією GNU GPLv2 і загальнодоступне для досліджень, розробок і використання.

Змодельовані сценарії запускаються з різних ситуацій перевантаження і з різними значеннями часу RTT, щоб перевірити реакцію алгоритмів на стан перевантаження.

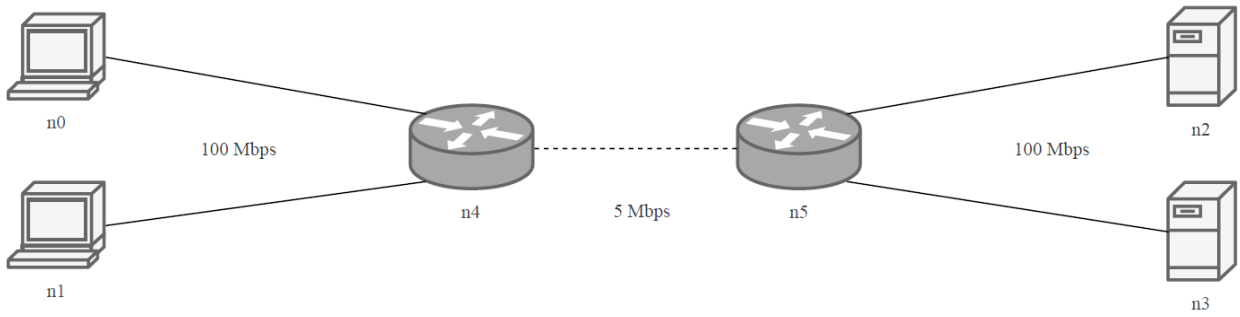

Рисунок 5.1 – Топологія мережі для моделювання

5.2 Топологія

Топологія, обрана для моделювання, представляє собою просту топологію гантелі, яка генерує трафік від лівих вузлів до правих.

Топологія складається з 5 мереж, розділених двома маршрутизаторами. Граничні канали мають пропускну здатність 100 Мбіт/с, а з'єднання між маршрутизаторами – 5 Мбіт/с. Власне роутер n_4 – це вузьке місце мережі, з

реалізацією алгоритму AQM. Мінімальна ємність буфера визначається практичним правилом, яке залежить від кожного сценарію, в будь-якому випадку, для моделювання використовується значення 2 BDP («bandwidth delay product» або мережева затримка, помножена на пропускну здатність канал). Трафік буде передаватися від вузлів n_0 і n_1 до вузлів n_2 і n_3 відповідно. n_0 створює TCP-з'єднання з великими потоками, і навпаки, n_1 генерує з'єднання з легким потоком UDP, наприклад, чутливі до часу програми, що вимагають низьких значень затримки. Значення пропускної здатності кожного вузла варіюють в залежності від сценарію, створюючи більше чи менше перевантаження у вузькому місці.

Нарешті, на всіх вузлах, крім n_4, за замовчуванням встановлено управління чергою DropTail.

З іншого боку, n_4 порівнює алгоритми Droptail, CoDel, FQ_Codel і FQ_CoDel + ABE для кожного сценарію. Значення ЦІЛЬОВЕ ЗНАЧЕННЯ і INTERVAL є значеннями за замовчуванням (5 мс і 100 мс відповідно), для ABE значення β_{ECN} встановлено в 0,7.

5.3 Сценарії

Для оцінки продуктивності алгоритмів топологія гантелей розглядається при різних сценаріях.

Зміни вносяться в рівні перевантаження і значенні часу прийому-передачі (RTT). Зокрема, розглядаються такі сценарії:

- мінімальне значення часу прийому-передачі (RTT = 50 мс);
- середнє значення часу прийому-передачі (RTT = 100 мс);
- велике значення часу прийому-передачі (RTT = 400 мс).

Час RTT було вибрано для трьох ситуацій. Звичайне час прийому-передачі становить близько 50 мс, потім 100 мс часу прийому-передачі – це звичайний час в мережі Інтернет, і нарешті, 400 мс RTT відповідає великим міжконтинентальним з'єднанням (і деякі супутникові з'єднання).

При імітаційному моделюванні створюються два потоки: один великий потік TCP (швидкість джерела 4 Мбіт/с) і другий невеличкий поток UDP (швидкість джерела 4 Мбіт/с). Потік UDP буде працювати тільки протягом 10 секунд моделювання, змушуючи алгоритм управління чергою адаптуватися до змін трафіку. Час імітаційного моделювання 30 секунд, останні 3 секунди – без потоків трафіку. Таким чином, трафік TCP буде присутній протягом 27 секунд, а UDP буде присутній в інтервалі від 10 до 20 секунди.

Час прийому пакетів і пропускна здатність відображаються за допомогою Wireshark, відстежуючи .pcap трасування потоків вузьких місць. Час проходження (RTT) різний при кожному сценарії: 50, 100 і 400 мс; однак час очікування в черзі можна розрахувати з різниці між теоретичним RTT шляху і реальним, виміряним у pcap.

За допомогою функції FlowMonitor відображається зведення моделювання. При відкиданні всього пакету (до і після черги) середня пропускна здатність та інша інформація.

5.4 Результати моделювання

Сценарій 1: RTT = 50 мс

На цьому етапі змішування потоків TCP і UPD викличе серйозну ситуацію перевантаження. Сумарний потік складається з «слонячого» потоку TCP зі швидкістю 4 Мбіт/с, змішаного з «мишачим» потоком UDP зі швидкістю 4 Мбіт/с. Поток TCP відправляє 10 МБ даних, в той час як UDP тільки 5 КБ, передача йде від n_0 до n_2 для TCP і від n_1 до n_3 для UDP, обидва через «вузьке місце» 5 Мбіт/с.

Час прийому-передачі 50 мс з ємністю черги n_4, що означає RTT нижче, ніж значення ІНТЕРВАЛУ CoDel за замовчуванням. FQ_CoDel поводиться інакше, ніж CoDel, він розділяє потоки по чергам, застосовуючи ізоляцію окремих потоків і пріоритет до нових потоків, застосовуючи алгоритм CoDel для кожного з них.

Оцінка обох основних алгоритмів проводиться з перевіркою затримки і пропускної здатності потоку TCP. З 10 по 20 секунди додається трафік потоку UDP, як показано на рисунках 5.2 і 5.3.

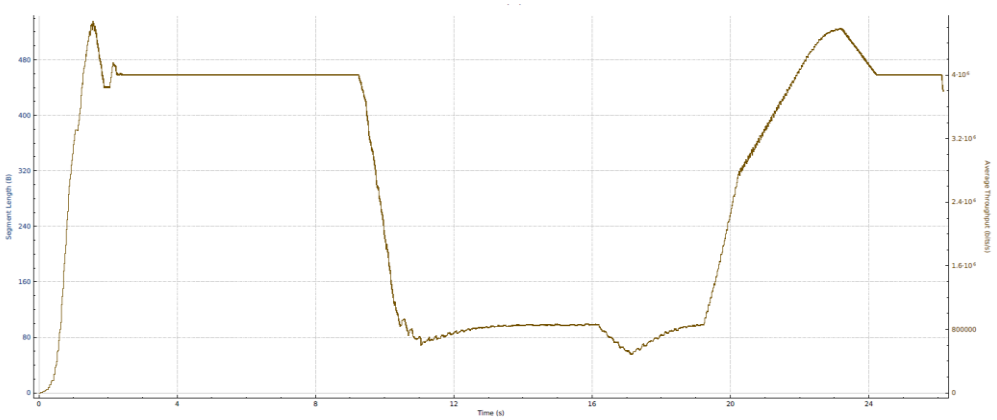

Рисунок 5.2 – Пропускна здатність CoDel (сценарій 1)

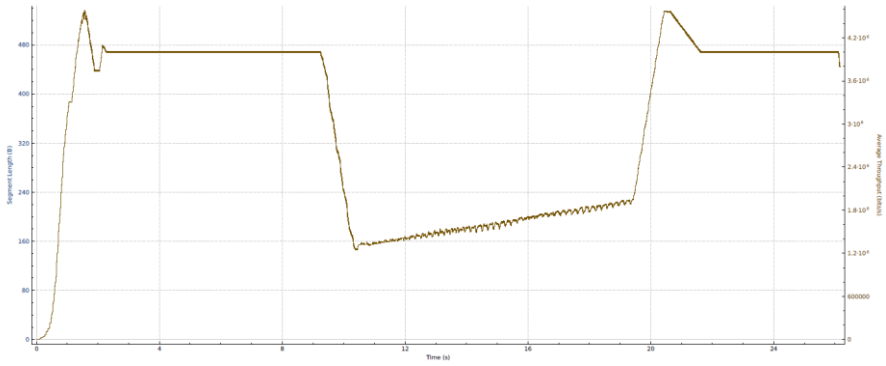

Рисунок 5.3 – Пропускна здатність FQ_CoDel (сценарій 1)

На рисунках показано зміну пропускної здатності (продуктивності) під час змішування трафіку. CoDel помітно знижує значення, досягаючи значень менше 1 Мбіт/с. З іншого боку, FQ_CoDel зменшує значення до 1,4 Мбіт/с, але починає плавно збільшувати його, поки потік UDP не закінчиться. Він забезпечує кращий результат, що пов'язано з ізоляцією потоків, що забезпечується FQ_CoDel, проте поведінка CoDel краща з великим RTT, як це видно нижче.

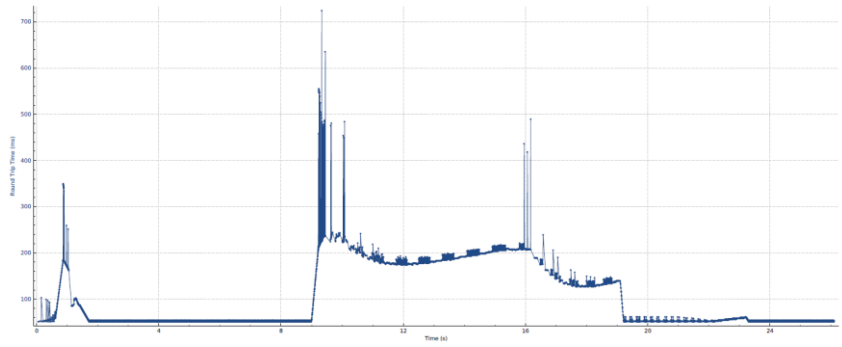

Рисунок 5.4 – RTT CoDel (сценарій 1)

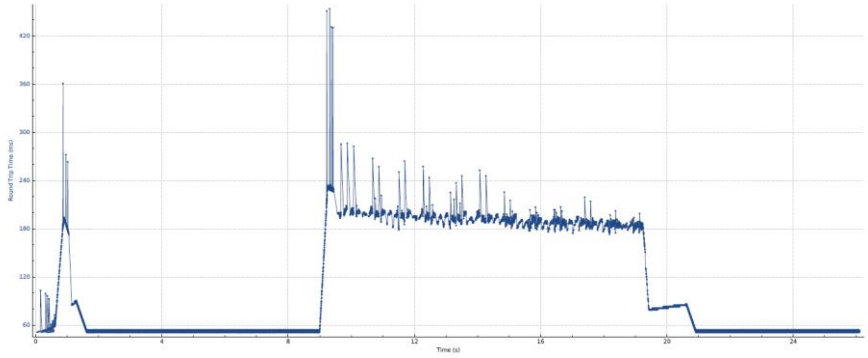

Рисунок 5.5 – RTT FQ_CoDel (сценарій 1)

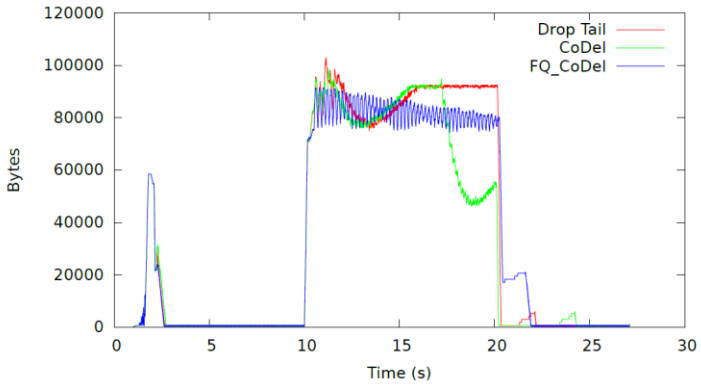

Рисунок 5.6 – Довжина черги маршрутизатора n_4 (сценарій 1)

На рисунку 5.6 показана зміна довжини черги в «вузькому місці» під час моделювання. Drop Tail має гіршу адаптованість до змішування трафіку, так як більшу частину часу перебуває з максимальною чергою. CoDel, зі

свого боку, пропонує кращу якість. Нарешті, результати FQ_CoDel нижче межі черги, з більш стабільним розвитком черги, іншими словами, з менш різким відкиданням пакетів.

Сценарій 2: RTT = 100 мс

Сценарій 2 працює зі змішаним трафіком при звичайному часу проходження Інтернет-з'єднання туди і назад. Пропускна здатність Drop Tail має найбільші значення завдяки TCP Reno, який завжди максимізує пропускну здатність, завдаючи шкоди іншим транспортним протоколам, таким як UDP. FQ_CoDel отримує кращі значення затримки через відкидання пакетів з черги, а саме, він відкидає більшу кількість пакетів, ніж CoDel. Така поведінка допомагає «мишачим» потокам долати вузьке місце без великих затримок. Крім того, ізоляція потоків FQ_CoDel забезпечує хороші значення пропускної здатності (продуктивності), отримуючи кращі результати, ніж у CoDel.

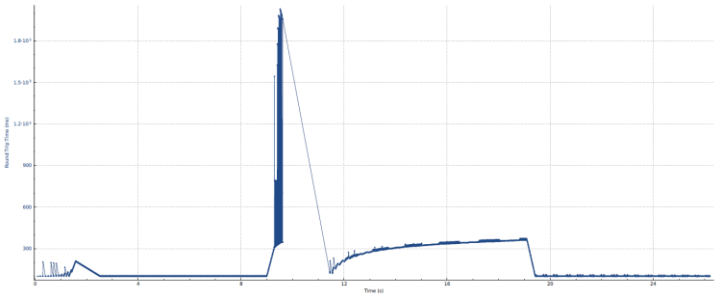

Рисунок 5.7 – RTT Drop Tail (сценарій 2)

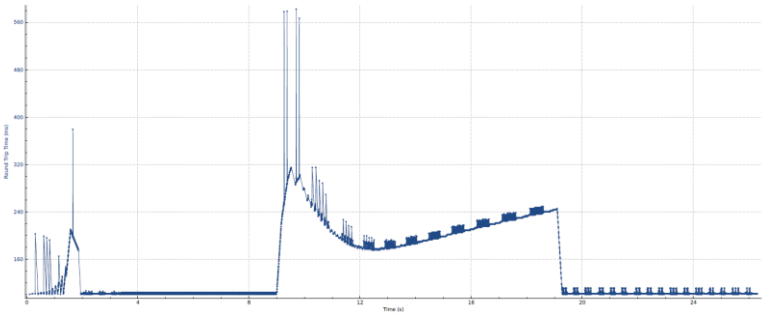

Рисунок 5.8 – RTT CoDel (сценарій 2)

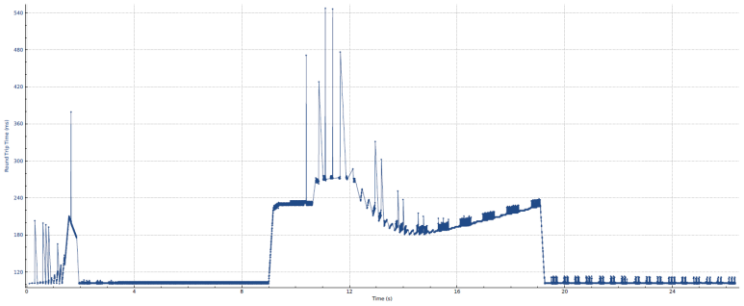

Рисунок 5.8 – RTT FQ_CoDel (сценарій 2)

Після повільного старту черга Drop Tail. не заповнюється (працює з швидкістю TCP 4 Мбіт/с при пропускній здатності вузького місця 5 Мбіт/с), незважаючи на це після того, як в чергу впроваджується потік UDP починає швидко заповнюватися (4 Мбіт/с UDP плюс 4 Мбіт/с потоку TCP), досягаючи межі черги через кілька мілісекунд. Після цього Drop Tail починає відкидати всі пакети, змушуючи TCP переходити до повільного старту (12 секунда). Потім черга знову швидко заповнюється, підтримуючи стабільне високе перевантаження під час присутності обох потоків.

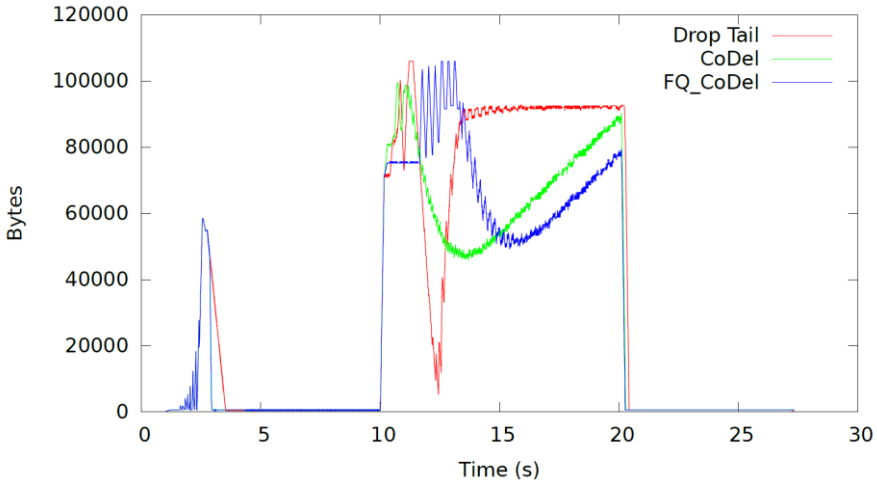

Рисунок 5.9 – Довжина черги маршрутизатора n_4 (сценарій 2)

CoDel, зі свого боку, реагує на зміну трафіку швидше, ніж Drop Tail і FQ_CoDel, швидко зменшуючи заповнення буфера нижче цільової точки.

Однак в цей період продуктивність по потоку нижче. FQ_CoDel найгірше працює з збільшенням навантаження на 1-2 секунді, але потім він адаптується до перевантаження, максимізуючи пропускну здатність і мінімізуючи затримку в шляху.

Сценарій 3: RTT = 400 мс

Оскільки результати Drop Tail найгірші і порівнювати їх не варто, далі розглядаються результати роботи лише алгоритмів CoDel і FQ_CoDel.

Значення затримки в обох випадках однакові, проте CoDel має більше сплесків і досягає великих значень затримки. Навіть в цьому випадку обидва алгоритму AQM швидко адаптуються до потоків, зменшуючи затримку до прийнятних значень, таких як 400 мс.

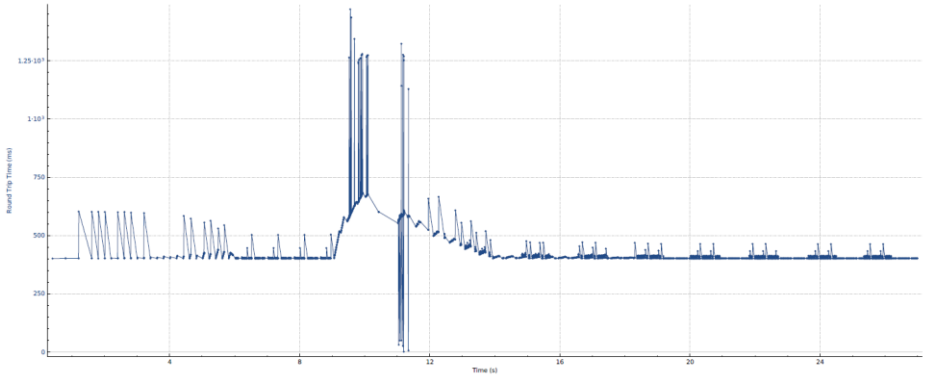

Рисунок 5.10 – RTT CoDel (сценарій 3)

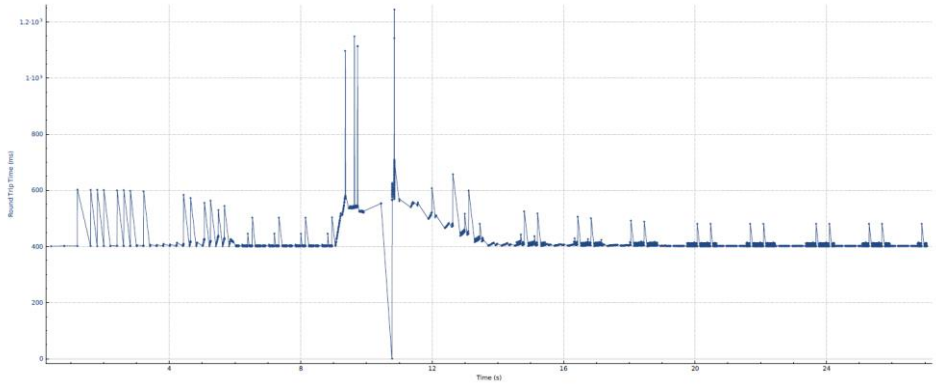

Рисунок 5.11 – RTT FQ_CoDel (сценарій 3)

На рисунку 5.12 показано, як алгоритми заповнюють свої черги під час моделювання. Черга Drop Tail, заповнюється через 12 секунд моделювання, досягаючи 333 пакетів черги (встановленої в 2 BDP), після чого він починає відкидати пакети, намагаючись повідомити відправнику про перевантаження в каналі.

CoDel і FQ_CoDel заповнюють третю частину черги (100 пакетів) тільки на дві секунди, потім вони управляють чергою для досягнення цільових результатів.

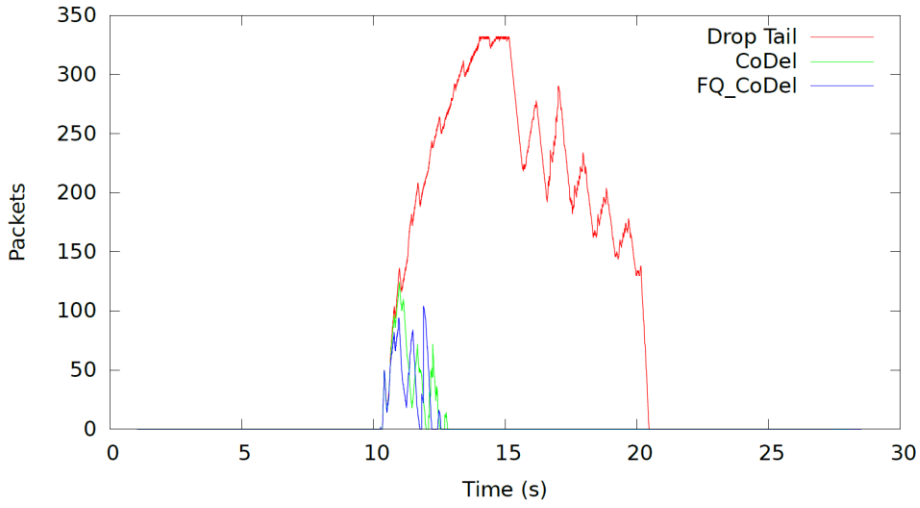

Рисунок 5.12 – Довжина черги маршрутизатора n_4 (сценарій 3)

Комбінація FQ_Codel і ABE є запропонованим методом, однак ns3 не дозволяє моделювати суміш цих двох алгоритмів, що робить необхідним ручне обчислення очікуваних результатів.

Згідно [29], використання β_{ECN} вище, ніж значення за замовчуванням 0,5, покращує пропускну здатність від 13% до 31% для потоку TCP Reno з керованою чергою CoDel. Встановивши 0,7 в якості корисного значення β_{ECN}, точний відсоток поліпшення складе 13% для RTT від 50 до 200 мс і 11% (з високою варіабельністю) для більш високого RTT, наприклад 400 мс. Застосовуючи ці значення, отриманий результат показано на рисунках 5.13, 5.14 і 5.15.

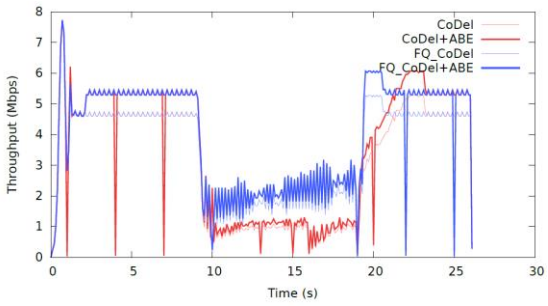

Рисунок 5.13 – Покращення пропускної здатності для RTT 50 мс

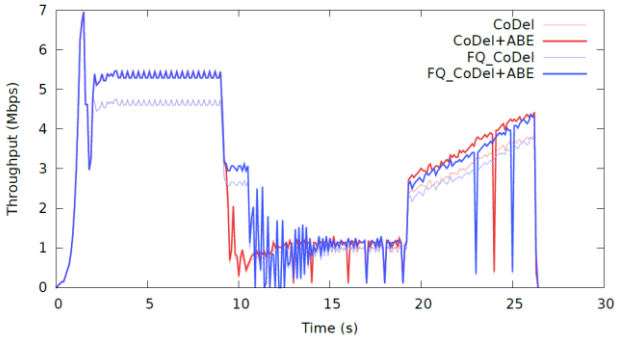

Рисунок 5.14 – Покращення пропускної здатності для RTT 100 мс

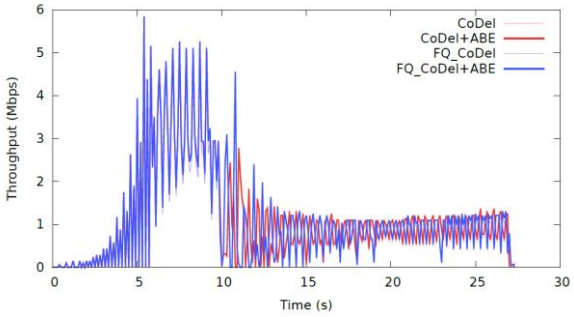

Рисунок 5.15 – Покращення пропускної здатності для RTT 400 мс

В цілому, середнє значення пропускної здатності для RTT 50 мс збільшується з 2,96 Мбіт до 3,34 Мбіт для CoDel і з 3,35 Мбіт до 3,78 Мбіт для FQ_CoDel; для RTT 100 мс воно змінюється з 2,51 Мбіт на 2,83 Мбіт для CoDel і з 2,53 Мбіт на 2,85 Мбіт для FQ_CoDel; нарешті, для RTT 400 мс воно збільшується з 0,99 Мбіт до 1,09 Мбіт для CoDel і FQ_CoDel.

ВИСНОВКИ

Схеми активного управління чергою (AQM) використовуються для забезпечення якості обслуговування (QoS) в телекомунікаційних мережах. Однак вони чутливі до налаштувань параметрів і мають слабкі сторони в виявленні і контролі перевантаження в динамічно мінливих мережевих ситуаціях. Перевантаження виникає в мережевих маршрутизаторах, коли кількість вхідних пакетів перевищує доступні мережеві ресурси, такі як буферний простір і виділення смуги пропускання. Це може привести до зниження продуктивності мережі щодо середньої затримки в черзі пакетів, швидкості втрати пакетів і пропускної здатності.

В рамках магістерської атестаційної роботи успішно вирішені наступні завдання:

- проведено огляд сучасних методів контролю перевантажень в комп'ютерних мережевих каналах;
- проведено аналіз факторів, що впливають на продуктивність мереж;
- проведено огляд різних методів управління трафіком комп'ютерних мереж;
- запропоновано метод управління чергами маршрутизаторів відповідно до поточного мережевого стану;
- проведено імітаційне моделювання теоретичних викладок;
- проведено аналіз отриманих результатів.

В атестаційній роботі запропоновано метод управління чергами маршрутизаторів шляхом сумісного використання алгоритму FQ_CoDel з механізмом явного повідомлення ABE (Alternative Backoff with ECN). Проведене імітаційне моделювання підтвердило, що їх сумісне застосування сприяє значному зростанню пропускної здатності, а також дозволяє знизити ймовірність виникнення значних перевантажень в IP мережах.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Floyd, S., & Fall, K. “Promoting the use of end-to-end congestion control in the Internet”. IEEE/ACM Transactions on Networking (TON), 7(4), 458-472.

2. Lapsley, D. and S. Low. 1999. “Ransom Early Marking for Internet Congestion Control”, Proceeding of GlobeCom’99, pp.1747-1752.

3. Gibbens, R. and F. Kelly. 1999. “Distributed Connection Acceptance Control for a Connectionless Network”, Proceeding of the 16th Intl. Teletraffic Congress, Edinburgh, Scotland, pp. 89-97.

4. Hoolot, C., V. Misra, D.Towlsey and W.Gong. “On Designing Improved Controllers for AQM Routers Supporting TCP Flows”, UMass CMPSCI Technical Report 00-42.

5. Jain R. “Congestion control in computer networks: Issues and trends,” IEEE Network Magazine, May 1990, pp 24–30.

6. Shang H. and Wills C. E. “Making better use of all those TCP ACK Packets,” tech. rep., Computer Science Technical Report Series, Worcester Polytechnic Institute, WPI-CS-TR-05-13, 2005.

7. Nagle J. “Congestion Control in IP/TCP Internetworks”, RFC 896, Ford Aerospace and Communications Corporation, January 1984.

8. Jacobson V. “Congestion Avoidance and Control”, in Proceedings of ACM/SIGCOMM, pp 314-329, 1988.

9. Mahmud Etbega, M.E. Woodward, A. G. Ali and Hussein A. “A New Version of Adaptive RED with Reduced Dependency on Parameterisation”, Proceedings of the Fourth International Working Conference on Heterogeneous Networks (HETNETs’ 06), pp WP10/1- WP10/8, llkley, UK, September 2006.

10. Tanenbaum S. “Computer Networks Fourth Edition, Section 5.3: Congestion Control Algorithms”, Prentice Hall, March 2003.

11. Feng W. C., Kandlur D., Saha D., and Shin K. “A self-configuring red

gateway,” In Infocom’99, March 1999.

12. Yang C. and Reddy A. V. S. A Taxonomy for Congestion Control Algorithms in Packet Switching Networks, IEEE Network, vol 9(5), pp 34-44, 1995.

13. Welzel M. “Network Congestion Control: Managing Internet Traffic”, Wiley Series in Communications Networking & Distributed System, West Sussex, England, September 2005.

14. Yung Y and Shakkottai S “Hop-by-hop congestion control over a wireless multihop network”, IEEE INFOCOM, vol 4, pp 2548–2558, 2004.

15. Allman M., Paxson V., and Stevens W. “TCP Congestion Control”, April 1999, RFC 2581.

16. Stevens R. TCP IP illustrated the protocols. Addison-Wesley professional computing series, Reading, Mass: Addison-Wesley Pub. Co, vol 1, 1994.

17. Brandauer C., Iannaccone G., Diot C., Ziegler T. “Comparison of Tail Drop and Active Queue Management Performance for Bulk-date and Web-like Internet Traffic”, Proc. ISCC, pp. 122-129, 2001.

18. Wang L., Min G., Awan I. “Modelling Active Queue Management with Different Traffic Classes," Proc. Int. Conference on AINA, pp. 442-446, Vienna, Austria, April 18-20, 2006.

19. Zadeh H. Y., Habibi A., Jafarkhani H., Bauer C. “Optimal Statistical Tuning of the RED parameters,” in Proceedings of IEEE ICC, pp. 27-32, Beijing, China, May 2008.

20. Wang L., Min G., and Awan I. “Modeling and Analysis of Active Queue Management Schemes under Bursty Traffic,” Journal of Wireless Information Networks, vol. 13, no. 2, pp. 161-171, 2006.

21. Feng W. C., Kandlur D. D., Saha D., Shin K. G. “Blue: A New Class of Active Queue Management Algorithms,” Univ. of Michigan, Ann Arbor, Tech. Rep. CSETR-387-99, 1999.

22. Kunniyur S., Srikant R. “An Adaptive Virtual Queue (AVQ) for Active

Queue Management," IEEE/ACM Transactions on Networking, vol. 12, pp. 286-299, April 2004.

23. Long C., Zhao B., Guan X. "SAVQ: Stabilized Adaptive Virtual Queue Management Algorithm," IEEE Communications Letters, vol. 9, no. 1, pp. 78-80, January 2005.

24. Agarwal M. K., Gupta R., Kargaonkar V. "Link Utilization Based AQM and its Performance," IEEE GLOBECOM, vol. 2, pp. 713=718, December 2004.

25. Athuraliya S., Lapsley D.E., Low S.H. "Random Exponential Marking for internet congestion control" IEEE Transactions on Network, June 2001.

26. Deng X., Yi S., Kesidis G., Das C. R. "Stabilised Virtual Buffer (SVB)-An Active Queue Management Scheme for Internet Quality of Service," IEEE GLOBECOM, vol. 2, pp. 1628-1632, November 2002.

27. Sun J., Zukerman M. "RaQ: A Robust Active Queue Management Scheme Based on Rate and Queue Length," Computer Communications, vol. 30, no. 8, pp. 1731-1741, February 2007.

28. Rong Pan Preethi Natarajan Chiara Piglione Mythili Suryanarayana Prabhu Vijay Subramanian Fred Baker and Bill VerSteeg. PIE: A Lightweight Control Scheme to Address the Bufferbloat Problem, 2013.

29. Процик О.В., Янковський О.А., Баранова О.А. Аналіз і прогнозування трафіку в комп'ютерних мережах. / О.В. Процик, О.А. Янковський, О.А. Баранова // Збірник тез доповідей 8 Міжнародної науково-технічної конференції «Проблеми Інформатизації», – Черкаси– Харків-Баку– Бельсько-Бяла. – 2020. – С. 55.