Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Програмно-алгоритмічні методи оптимізації трафіку комп'ютерних мереж

(тема)

Виконав:
студент II курсу, групи СПм-20-2
Бровенко І.М.
(прізвище, ініціали)

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

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

Освітня програма
Системне програмування
(повна назва освітньої програми)

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

Допускається до захисту

В.о. зав. кафедри ЕОМ _____ Волк М.О.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Бровенку Івану Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-алгоритмічні методи оптимізації трафіку комп'ютерних мереж

затверджена наказом по університету від " 24 " березня 2022 р. № 413 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 2022 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 управління перевантаженням та середньою затримкою

3) моделі 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)

Слайдів презентації – 18 шт.

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	29.03.22–31.03.22	
	вирішення		
2	Огляд технологій управління перевантаженням	01.04.22–04.04.22	
3	Розробка моделі управління мережним	05.04.22 –18.04.22	
	трафіком		
4	Вибір програмних та апаратних засобів	19.04.22 –25.04.22	
	реалізації		
5	Тестування запропонованого метода	26.04.22–28.04.22	
6	Оформлення пояснювальної записки	29.04.22–-06.05.22	

Дата видачі завдання 28 березня 2022 р.

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 93 с., 22 рис., 1 дод., 18 джерел.

AQM, КАНАЛ ЗВ’ЯЗКУ, МЕРЕЖЕВИЙ РІВЕНЬ, МЕТРИКА, МОДЕЛЮВАННЯ, ПЕРЕВАНТАЖЕННЯ, ПОРІГ СКИДАННЯ, ПРОПУСКНА СПРОМОЖНІСТЬ, ТОПОЛОГІЯ.

Мета кваліфікаційної роботи – розробка моделей та методів управління перевантаженням та довжиною черг маршрутизаторів.

В роботі досліджуються проблеми методів активного керування чергою (AQM) для контролю перевантажень у мережах TCP. Випадкове раннє виявлення (RED) і стратегії на основі RED, які використовують підхід AQM, оцінюються за допомогою моделювання. Розглянуто дві основні проблеми RED та його варіантів. Пропонуються нові стратегії та моделюється їх поведінка за допомогою мережевого симулятора.

ABSTRACT

Master’s thesis: 93 pages, 22 figures, 1 appendices, 18 sources.

AQM, COMMUNICATION CHANNEL, NETWORK LEVEL, METRICS, SIMULATION, OVERLOAD, DOWN THRESHOLD, CAPACITY, TOPOLOGY.

The purpose of the qualification work is to develop models and methods for managing congestion and queue length of routers.

The paper investigates the problems of active queue management (AQM) methods for overload control in TCP networks. Random early detection (RED) and RED-based strategies that use the AQM approach are estimated using simulations. Two main problems of RED and its variants are considered. New strategies are proposed and their behavior is modeled using a network simulator.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACK – підтвердження нового пакету TCP (англ., New TCP packet acknowledgment)

AQM – активне управління чергою (англ., Active queuing mechanism)

ARED – Адаптивне випадкове раннє виявлення (англ., Adaptive random early detection)

AS – автономна система (англ., Autonomous System)

Cwnd – вікно перевантаження TCP (англ., TCP congestion window)

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

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

GRED – пологий RED (англ., Gentle RED)

FIFO – першим прибув, першим обслужений (англ., First In First Out)

FRED – потоковий RED (англ., Flow RED)

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

PQ – пріоритетні черги (англ., Priority Queuing)

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

RED – випадкове раннє виявлення (англ., Random Early Detection)

REM – випадкове раннє маркування (англ., Random Early Marking)

RTO – очікування повторної передачі (англ., Retransmission timeout)

RTT – час подвійного оберту (англ., Round trip time)

SRED – стабілізований RED (англ., Stabilized RED)

Ssthresh – поріг повільного запуску (англ., Slow start threshold)

TCP – протокол керування передачею (англ., Transmission Control Protocol)

UDP – протокол дейтаграм користувача (англ., User Datagram Protocol)

ВСТУП

Контроль перевантажень є важливим для забезпечення належного функціонування Інтернету. Без контролю перевантажень Інтернет став би непридатним для використання через перевантаження. Механізм контролю перевантажень протоколу керування передачею TCP виявився надійним для забезпечення стабільності Інтернету, а алгоритми контролю перевантажень, запозичені з TCP, в основному застосовуються також іншими транспортними протоколами.

Протягом багатьох років структура інтернет-трафіку змінилася. Однією з основних причин є всесвітня павутина WWW. Незабаром після свого винаходу в 1989 році Інтернет став дуже популярним, і з тих пір велике частине Інтернет-трафіку становить веб-трафік. Веб-сторінки передаються за допомогою протоколу передачі гіпертексту HTTP, який накладається поверх протоколу керування передачею TCP. Веб-трафік складається здебільшого з коротких потоків TCP, і часто відбувається чергування активних і неактивних періодів передачі. Веб-браузери зазвичай використовують паралельні з’єднання, щоб зменшити затримку під час отримання веб-сторінки, яка складається з багатьох об’єктів HTTP.

TCP має вбудований механізм контролю перевантажень для забезпечення стабільності Інтернету, запобігаючи виникненню перевантажень та використанню надмірної швидкості надсилання пакетів протягом тривалого періоду часу. Механізм контролю перевантаження TCP самотактується за допомогою пакетів підтвердження ACK як зворотного зв’язку і намагається дотримуватися «принципу збереження пакетів», який дозволяє надсилати новий пакет даних до мережі лише тоді, коли інший пакет залишив її. Контроль перевантаження TCP має дві основні фази: повільний старт і уникнення перевантажень. Відправник TCP спочатку використовує повільний старт, щоб перевірити доступну ємність на

наскрізному шляху, який невідомий відправникові. Таймер ACK завантажується для повільного запуску шляхом надсилання пакетів до мережі TCP Initial Window (IW). Потім відправник експоненціально збільшує швидкість надсилання під час повільного запуску. Відправник повинен підтримувати принцип збереження пакетів щоразу, коли він збільшує швидкість відправлення, тому що дотримання принципу дозволить лише підтримувати ту саму швидкість надсилання. Повільний запуск триває до тих пір, поки потік TCP не досягне «стелі», яку відправник розпізнає через пакети, скинуті в мережу через перевантаження. Після отримання належної швидкості відправлення TCP-відправник продовжує працювати в режимі запобігання перевантаженню, що є набагато менш агресивним порівняно з повільним запуском. Можна порівняти ці дві фази контролю заторів із штормом і спокійною водою, причому шторм є повільним стартом, а спокійна вода відповідає уникнення заторів. Що ще гірше, шторми, викликані повільним запуском, не вщухають, доки відправник не знайде стелю через відкидання пакетів.

Використання коротких потоків TCP з веб-трафіком підкреслює важливість фази Slow-Start, яка зараз є нормою, а не рідкістю. Це також робить початкове вікно важливим, оскільки веб-трафік часто використовує паралельні потоки для передачі веб-сторінки. Повільний старт створює експоненційні перехідні процеси навантаження. Вплив експоненційних перехідних процесів навантаження на навантаження каналу є найбільш значущим поблизу межі мережі, де агрегація трафіку обмежена лише одним або кількома користувачами, які спільно використовують канал.

Така мережа доступу сама по собі є також доволі часто вузьким місцем, тобто найвужчим каналом на всьому наскрізному шляху. Коли розташування вузького місця знаходиться близько до кінцевого користувача, дуже ймовірно, що його може наситити лише один кінцевий користувач, на відміну від ядра мережі, де рівень агрегації трафіку вищий, ніж біля краю мережі.

1 МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

Коли швидкість відправлення вища за швидкість зв’язку вузького місця, на маршрутизаторі утворюється черга перед вузьким місцем. Маршрутизатору потрібно якось керувати чергою. Традиційно використовується проста політика «першим прийшов, першим вийшов» (FIFO) разом із видаленням з хвоста, коли буфер заповнений (Taildrop). Через проблеми з постійно заповненими або майже заповненими буферами було введено активне керування чергою (AQM), щоб проактивно скидати пакети до того, як буфер стане заповненим, щоб залишити місце в буфері для тимчасових пакетів. Під час повільного запуску черга, як правило, тимчасова, оскільки канал ще не насичений. Черга зменьшується під час проміжних періодів простою кожного RTT. Однак повільний старт швидко переходить від низького навантаження до повного та перевантаження всього за кілька RTT через експоненційний характер збільшення швидкості надсилання пакетів.

Виконання магістерської кваліфікаційної роботи передбачає розробку методів контролю за довжиною черг мережевих маршрутизаторів застосовуючи різноманітні характеристики існуючого трафіку.

Виконання кваліфікаційної магістерської роботи передбачає:

- проведення докладного аналізу сучасних літературних джерел, присвячених проблематиці підвищення пропускної здатності комп’ютерних мереж;
- виконаня аналізу існуючих моделей управління мережевим перевантаженням та довжиною черг маршрутизаторів;
- розробку методу управління станом черг маршрутизаторів з виконанням імітаційного моделювання запропонованого методу;
- проведення аналізу отриманих результатів.

2 ОГЛЯД СУЧАСНОГО СТАНУ ПРОБЛЕМИ

Протокол керування передачею (TCP) — це протокол транспортного рівня, який керує 90% передачі даних в Інтернеті [1-3]. Оскільки Інтернет еволюціонував і кількість користувачів різко зросла, виникла потреба у розробці нових методів для забезпечення справедливого розподілу ресурсів між користувачами.

Перевантаження створюється, коли попит перевищує доступну потужність. Через неузгоджений розподіл ресурсів Інтернет страждає від проблем тривалих затримок доставки даних, витрачання ресурсів через втрачені або скинуті пакети і навіть можливого колапсу перевантаження, який виникає, коли зупиняється вся мережа [4].

Затор можна вирішити численними підходами. Традиційно маршрутизатори використовували систему First In First Out (FIFO), відкидаючи пакети з хвоста черг для контролю перевантаження.

2.1 Огляд проблеми заторів

Основною одиницею потокової передачі даних в TCP є байт. TCP призначає порядковий номер кожному переданому байту; потім це використовується для керування потоком і підтвердження даних. Однак в Інтернеті дані передаються сегментами (пакетами). Після прибуття сегмента даних пункт призначення підтверджує отримання сегмента, надсилаючи підтвердження (ACK) з наступним очікуваним номером сегмента даних. Якщо ACK не отримано протягом інтервалу тайм-ауту, дані передаються повторно.

Приймач повідомляє про кількість байтів, які він може отримати за межами останнього отриманого сегмента TCP. Це запобігає переповненню внутрішнього буфера приймача. Оголошена кількість байтів називається

рекомендованим вікном і включається в заголовок кожного підтвердження, надісланого до джерела. Час руху в обидва боки (RTT) – це час, проведений між відправленням сегменту і його ACK, отриманий відправником. Він включає затримки поширення, передачі, перебування в чергах та обробки на всіх проміжних маршрутизаторах.

У мережах TCP сегменти проходять через черги маршрутизаторів. Характеристикою трафіку TCP є те, що сегменти можуть надходити пакетами з одного або кількох джерел. Буфери допомагають маршрутизаторам поглинати всплески кількості надійшовших пакетів, поки вони не зможуть відновити свій попередній стан.

У разі надмірного трафіку буфери перевантажуються, а нові вхідні пакети відкидаються. Статистичні рішення, такі як збільшення розміру буферів або додавання більшої кількості буферів, неефективні [5], оскільки надмірна буферизація може викликати тривалі затримки [6]. Таким чином, вікно перевантаження cwnd використовується для запобігання переповненню буфера шлюзу.

Перевантаження мережі вказується двома подіями: перша – це втрата пакетів, яка виявляється за тайм-аутом; а другий – це дубльоване підтвердження [7].

У разі перевантаження TCP зменшує розмір вікна (cwnd), який вказує швидкість надсилання даних. Це коригування залежить від швидкості підтверджень, що надходять до джерела.

Вузол джерела TCP налаштовує вікно перевантаження на основі сигналів перевантаження. Він зменшує його, коли рівень заторів зростає, і збільшує, коли рівень заторів знижується. Цей механізм зазвичай називають адитивним збільшенням/мультиплікативним зниженням (AIMD) [8]. Зрозуміло, що приріст/зменшення надходження ACK залежить від стану проміжних маршрутизаторів. Іншими словами, швидкість надходження та відправлення даних ACK буде співпадати, якщо немає перевантаженого маршрутизатора.

Це автоматичне виявлення перевантажень TCP існує відповідно до трьох основних режимів:

- повільний старт і уникнення перевантажень: TCP-Tahoe [4];
- швидка повторна передача та швидке відновлення: TCP-Reno [7];
- TCP-Vegas [8].

2.2 Управління чергою та контроль завантаженості

Інтернет – це система доставки пакетів. Перевантаження може призвести до великих втрат пакетів і збільшення затримок; які знижують продуктивність мережі. Отже, контроль перевантажень є найважливішим елементом TCP. Будь-яке обговорення заторів, природно, включатиме черги. Для управління мережевим буфером використовуються різноманітні методи черги. Правильне керування чергою мінімізує кількість скинутих пакетів і перевантаження мережі, а також покращує продуктивність мережі. Джерела TCP виявляють перевантаження через повторювані ACK.

Проміжні маршрутизатори використовують підходи Active Queue Management (AQM) для виявлення перевантаженості мережі. Ці методи працюють безпосередньо з буфером маршрутизатора, вимірюючи та відстежуючи середній розмір черги.

Поточні стратегії обробки перевантажень в Інтернеті використовуються для підвищення продуктивності, що зазвичай проявляється високою пропускною здатністю, високим рівнем використання каналів, низьким рівнем втрат і низькою середньою затримкою із кінця в кінець. Відповідно, було запропоновано багато підходів щодо контролю заторів. Перевантаження можна контролювати за допомогою стратегій, які застосовує відправник TCP. Такими стратегіями є вихідні алгоритми, наприклад, Vegas [9] і Tahoe [4]. Інший тип стратегії, яку застосовують проміжні маршрутизатори, називається мережевим алгоритмом, таким як Tail Drop [10] і RED [11].

2.2.1 Відновлення після заторів

Класичні мережеві маршрутизатори керують перевантаженнями, встановлюючи максимальну довжину черги (поріг). Коли розмір черги перевищує дозволений поріг, усі вхідні пакети згодом відкидаються, доки не стане доступним місце в буфері.

Ця техніка називається Tail Drop (TD), оскільки пакети відкидаються з хвоста черги. TD вважався ефективним методом контролю заторів до виявлення двох серйозних недоліків. Такими недоліками є повна черга та локаут. Tail Drop реактивно контролює перевантаження буфера. Це також може спричинити глобальну синхронізацію, коли всі джерела мережі одночасно знижують швидкість надсилання пакетів. Це може призвести до низького рівня використання каналу або проблеми з блокуванням, коли кілька джерел монополізують всю пропускну здатність каналу.

Глобальні проблеми синхронізації та блокування можуть бути вирішені шляхом застосування техніки випадкового скидання на проміжному маршрутизаторі. Щоразу, коли новий пакет надходить до перевантаженого шлюзу, пакет випадковим чином вилучається з черги. Отже, ймовірність відкидання пакету з конкретного потоку пропорційна частці пропускної здатності цього потоку. Ця технологія інформує агресивних користувачів про те, що їхній трафік сприяє перевантаженню більше, ніж інші користувачі в мережі. Random Drop (RD) забезпечує справедливий розподіл ресурсів між з’єднаннями і покращує пропускну здатність мережі для з’єднань з більш тривалими RTT.

2.2.2 Уникнення заторів

Методи відновлення заторів повільно реагують на затор. Ці методи часто виявляють перевантаження після переповнення буфера. Вони не вирішують повну проблему черги, і з цієї причини був запропонований

підхід уникнення перевантажень. Явища повної черги можна уникнути, передбачивши перевантаження на його ранніх стадіях. Таким чином, у проміжному маршрутизаторі має бути реалізований проактивний підхід для виявлення початкових етапів перевантаження. Цей підхід часто називають активним керуванням чергою (AQM). Замість фактичного відкидання пакетів AQM позначає пакети на перевантаженому маршрутизаторі та повідомляє джерелам, щоб вони сповільнилися. Техніка маркування може бути реалізована за допомогою біта, встановленого в заголовку пакета.

AQM досягає таких цілей:

- запобігає глобальній синхронізації;
- визначає середній розмір черги та зменшує затримки;
- усуває упередження проти швидкого трафіку;
- визначає агресивних користувачів, які спричиняють перевантаження;
- зменшує швидкість відкидання пакетів за допомогою техніки маркування.

Випадкове відкидання спочатку було запропоновано як підхід до відновлення після заторів. Однак завдяки покращенню Early Random Drop (ERD) його можна використовувати як техніку уникнення перевантажень. Швидкість відкидання в ERD обмежена рівнем заторів у шлюзі. Основна ідея ERD полягає у встановленні фіксованого порогу. Щоразу, коли розмір черги перевищує цей поріг, пакети випадковим чином вилучаються з черги. Більш складні форми ERD скидають пакети на основі експоненціально зваженого ковзного середнього (EWMA) замість фактичного розміру черги. ERD перевершує традиційну техніку TD з точки зору сегрегації потоків, однак методика ERD не забезпечує справедливого розподілу ресурсів або не стримує агресивних користувачів. ERD має упередження щодо раптових сплесків трафіку.

Випадкове раннє виявлення (RED) було запропоновано Флойдом і Джейкобсоном [11] для подолання недоліків ERD. Основна мета RED – забезпечити ефективну швидкість відмітки/відкидання для контролю

середнього розміру черги та уникнення упереджень щодо швидкого трафіку. Початкові етапи перевантаження визначаються за середнім розміром черги. Якщо середній розмір черги перевищує встановлений поріг, RED скидає/позначає пакети, що надходять, з імовірністю відкидання; це функція середнього розміру черги. Шлюзи RED підтримують низький середній розмір черги.

RED був надзвичайно впливовою стратегією в області AQM. Він підтримує набір параметрів для контролю заторів. Однак RED дуже важко параметризувати, оскільки продуктивність RED може працювати подібно до відкидання хвоста за деяких умов трафіку. Вимірювання параметрів і рекомендовані значення з часом можуть змінюватися. Згодом багато стратегій, заснованих на RED, було запропоновано після початкової пропозиції RED.

3 ПЕРЕВАНТАЖЕННЯ КАНАЛІВ

Перевантаження мережі – це явище, викликане перевантаженням попиту на кінцеві мережеві ресурси. Коли попит перевищує доступну пропускну здатність мережі, продуктивність мережі знижена, що призводить до довгих затримок і втрат пакетів. Найважчим результатом перевантаження є колапс мережі, при якому зупиняється вся мережа [5].

Дуже важливо розробити стратегії контролю перевантажень, які забезпечують оптимальну роботу мережі. Стратегії контролю перевантажень повинні знизити попит, регулюючи швидкість надсилання пакетів в перевантажені канали. Виділення буферів більшого розміру або застосування більш швидших каналів, які є особливостями багатьох статичних рішень, не запобігають перевантаженню мереж [5]. Ці рішення можуть спричинити серйозні затори, що призводять до поганої продуктивності.

Термін керування потоком іноді плутають із керуванням перевантаженням. Контроль потоку – це метод на основі вікон, який використовується джерелами TCP, щоб запобігти перевантаженню повільних одержувачів швидкими відправниками. Таким чином, ці стратегії називаються вихідними алгоритмами. Вікно перевантаження (cwnd) — це кількість даних, яку можна надіслати до отримання підтвердження. При відсутності заторів вікно збільшується для збільшення швидкості відправки. У разі перевантаження вікно зменшується, щоб зменшити швидкість надсилання, доки не зменшиться затор.

Методи керування потоком TCP, такі як TCP-Tahoe і TCP-Reno, використовуються протягом тривалого періоду часу. Однак буферний простір у маршрутизаторах обмежений, і тому потрібні методи для керування розміром черги. Отже, різні стратегії контролю заторів; такі як Random Early Detection (RED), були запропоновані для запобігання перевантаження на проміжному маршрутизаторі. Цей тип стратегії є мережевою стратегією,

оскільки вона застосовується мережевими компонентами, такими як маршрутизатори, для контролю перевантажень.

Відповідно, існує два підходи до боротьби з перевантаженням мережі; реактивний і проактивний підходи. Реактивний підхід, який застосовується такими вихідними алгоритмами, як Tahoe і Reno, починає відновлюватися після переповнення мережевого буфера. Проактивний підхід спрямований на запобігання переповнення буфера за допомогою стратегій, що застосовуються в проміжному маршрутизаторі. RED [5] є прикладом проактивного підходу.

Це нормально, коли кілька IP-пакетів надходять одночасно на проміжний маршрутизатор; який очікує на пересилання через перевантажений канал. Таким чином, проміжні маршрутизатори використовують буферний простір, щоб поставити пакети в чергу, поки їх не можна буде почати обслуговувати. На деяких етапах швидкість надсилання мережевих джерел перевищує доступний буферний простір у маршрутизаторах, і пакет потрібно відкинути. У цьому випадку традиційні маршрутизатори, які використовують техніку керування чергою «Перший прийшов, перший вийшов» (FIFO), скидають пакети з хвоста черги.

Нескінченний буфер не вирішить перевантаження; оскільки створена черга буде необмеженою, що збільшує наскрізну затримку. Пакет із тривалими затримками, можливо, вже вважається втраченим та був повторно переданий джерелом. Невеликий розмір буфера краще, ніж великий; оскільки це зменшить затримки та заощадить ресурси пам’яті.

3.1 Колапс заторів

Перевантаження збільшує затримки доставки даних і витрачає ресурси мережі через втрачені пакети. Це також може призвести до перевантаження, коли припиняться всі мережеві з’єднання. Зростання перевантаження різко погіршує пропускну здатність мережі, як показано на рисунку 3.1. Зростання

заторів з’являлося в різних формах з перших днів існування Інтернету. Перший випадок був описаний у 1984 році. Нейгл помітив повторні передачі для пакетів, які вже були отримані або все ще перебувають у транзиті. Ця форма перевантаження знизила пропускну здатність мережі.

Важка форма колапсу перевантаження викликається недоставленими пакетами. Ця форма перевантажень витрачає пропускну здатність мережі через відкидання пакетів, перш ніж вони досягнуть кінцевого пункту призначення. Рішення цієї форми заторів полягає в зменшенні пропонованого навантаження. Існують також дві форми перевантаження на основі фрагментації та від застарілих пакетів. Перший є результатом передачі фрагментів пакетів, які будуть неодмінно відкинуті одержувачем, оскільки вони не можуть бути повторно зібрані в дійсний пакет.

Останнє викликано марнуванням пропускної здатності мережі на пакетні передачі, які більше не потрібні користувачеві.

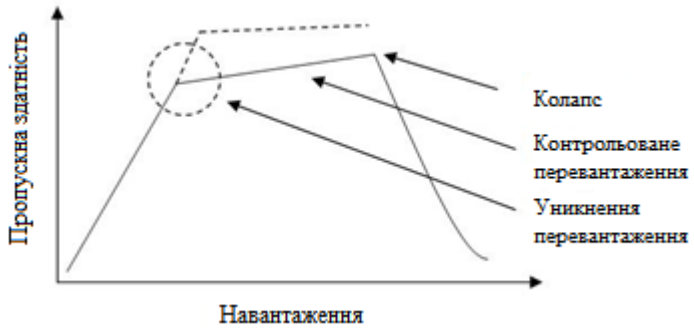


Рисунок 3.1 – Залежність пропускної здатності від навантаження

3.2 Перевантаженість і неправильна поведінка користувачів

Погані користувачі ігнорують сигнали перевантажень і, як правило, використовують пропускну здатність більше, ніж дозволена. Таким чином, користувачі, що погано себе ведуть, отримують краще обслуговування, ніж користувачі, які співпрацюють, і погіршують стабільність і роботу мережі.

Трафік TCP – це чутливий трафік, який зменшує швидкість передачі у відповідь на сигнали перевантаження. TCP призначений для спільного використання мережі з іншими типами трафіку, наприклад UDP, який не реагує на сигнали перевантаження. За відсутності належних методів контролю заторів; користувачі UDP-трафіку, будуть використовувати більше, ніж їхню частку мережевих ресурсів, таких як пропускна здатність. Тому було запропоновано, що користувачів, які погано поводяться, слід відключити. Стабільність вимірюється рівнем співпраці в мережі, оскільки вона дозволяє справедливо розподіляти ресурси.

3.3 Справедливий розподіл ресурсів

Справедливість стає проблемою, коли користувачі змагаються за свою частку ресурсів. Справедливий розподіл ресурсів задовольняється, коли пропускна здатність для кожного користувача відповідає всім іншим користувачам, які мають те саме вузьке місце.

Отже, Internet Engineering Task Force (IETF) розглядає принцип справедливості таким чином:

- ресурси розподіляються в порядку зростання попиту;
- користувачеві ніколи не виділяється частка, що перевищує його вимоги;
- усім користувачам із незадоволеними потребами розподіляються рівні частки.

3.4 Підходи контролю заторів

Розподіл ресурсів складається з двох частин: наскрізного та поланкового управління. Тому контроль перевантаження класифікується як методи на основі хостів і маршрутизаторів; стосовно місця, в якому ці методи реалізовані.

Керування потоком реалізовано в кінцевих хостах. Джерела мережі відповідають за наскрізний контроль потоків залежно від зворотного зв'язку маршрутизаторів. Коли шлюз ось-ось буде перевантажений, джерела TCP знижують швидкість надсилання, поки шлюз не відновиться після перевантаження.

Зворотний зв'язок може бути реалізований кінцевим хостом, щоб уникнути переповнення локального буфера у випадку, якщо швидкий відправник переповнює повільний буфер одержувача.

Однак зворотний зв'язок може бути реалізований окремо проміжними маршрутизаторами; в яких відкидання пакетів є основним сигналом перевантаження. TCP неявно реагує на відкидання пакетів, зменшуючи швидкість відправлення, що є найпростішою формою контролю перевантажень. Однак трактування маршрутизатора як чорного ящика має деякі обмеження щодо контролю розподілу ресурсів і зменшує послуги, що надаються мережею.

Маршрутизатори знають про початкові етапи перевантаження; тому вони повинні відігравати більшу роль у боротьбі з заторами. Маршрутизатори використовують дві методики для управління буфером і пропускною здатністю.

Перший варіант – зробити пряме виділення пропускної здатності на вихідному каналі. Другий варіант – зробити непряме виділення пропускної здатності, керуючи розміром черги маршрутизатора.

Існує два підходи до контролю перевантажень: перший – це відновлення після перевантажень, яке часто плутають з терміном керування перевантаженням. Другий підхід – уникнення заторів. Підхід до відновлення після перевантажень починає контролювати перевантаження після перевантаження черги шлюзу. І навпаки, уникнення перевантажень застосовує заходи для зменшення перевантажень до того, як шлюз буде перевантажено.

3.5 TCP

3.5.1 Параметри продуктивності TCP

У літературі використовуються різні параметри з різними визначеннями для вимірювання продуктивності мереж TCP. Для цієї мети зазвичай використовуються п’ять параметрів, а саме:

- пропускна здатність;
- використання каналу;
- втрата пакетів;
- затримка;
- тремтіння.

Нижче наведено деякі інші параметри, які використовуються для вимірювання продуктивності мережі:

- затримка в обидва боки – час, необхідний для того, щоб IP-дейтаграма подорожувала від джерела до пункту призначення і назад (цей параметр включає час поширення та затримку в черзі);

- затримка в одну сторону – через характеристики мережі, такі як обмеження маршрутизації та пропускної здатності; затримка в один бік не завжди дорівнює половині затримки в обидва боки (цей параметр дуже важливий в інтерактивних програмах, таких як передача голосу через IP);

- максимальна затримка – це максимально дозволена затримка в одну сторону для високої продуктивності мережі;

- тремтіння затримки – впливає на мінливість односторонньої затримки між дейтаграмами IP (більше тремтіння означає нижчу продуктивність мережі, особливо для мультимедійних програм);

- швидкість втрати пакетів – пакети можуть бути відкинуті проміжними маршрутизаторами через переповнення буфера (швидкість втрат – це відношення правильно отриманих пакетів до кількості розповсюджених пакетів;

- ефективна пропускна спроможність – її також називають ефективною пропускною здатністю, яка є кількістю байтів, які передаються через мережу за одну секунду;

- зміна пропускної спроможності – мінливість пропускної здатності протягом заданого часового масштабу;

- час передачі файлу – час, необхідний файлу або об’єкту для передачі від джерела до місця призначення;

- справедливість – цей параметр відображає справедливість розподілу ресурсів між клієнтами мережі;

- споживання ресурсів – методи TCP, які споживають менше мережевих ресурсів, відображають високу продуктивність мережі.

3.5.2 ACK

Для підтвердження доставки даних до місця призначення використовуються три типи підтверджень. Сегмент лише для підтвердження та підключення – одержувач посилає ACK пакет за пакетом. ACK може містити нульове корисне навантаження. Інший варіант включення номера ACK в пакет даних називається Piggybacking.

Відкладений ACK – замість підтвердження усіх пакетів один за одним, для групи сегментів можна надіслати відкладене підтвердження.

Дублікат ACK – дублікат ACK може використовуватися, щоб неявно вказувати відсутні пакети в мережах TCP. Наприклад, якщо n-1, n і n+1 є наступними пакетами. Коли отримано пакет з номером n-1, одержувач надсилає ACK номер n. Це інформує одержувача, що пакет номер n-1 отримано безпечно, а наступний очікуваний пакет – номер n. Якщо пакет з номером n втрачено в мережі і надійшов пакет з номером n+1, одержувач неявно знає, що пакет з номером n відсутній. У цьому випадку одержувач надсилає ACK номер n+2, щоб підтвердити одержувача пакета з номером n+1 і повідомити відправника про наступний очікуваний пакет n+2.

Незважаючи на те, що ACK номер n був раніше ініційований відправником, дубльований ACK з номером n повторно надсилається відправнику для зміни порядку сегмента з номером n. Деякі варіанти TCP використовують переваги цієї техніки для контролю перевантажень.

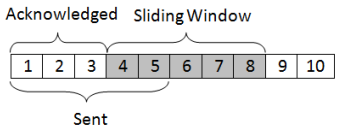


Рисунок 3.2 – Приклад ковзаючого вікна в мережах TCP

3.5.3 Регулювання потоку і ковзаюче вікно

Тайм-аут або потрійне підтвердження використовуються як сигнали перевантаження в мережах TCP. Коли відбувається перевантаження, алгоритм контролю перевантаження TCP повідомляє відправнику TCP про зменшення швидкості надсилання, зменшуючи вдвічі розмір вікна перевантаження. Поведінка самосинхронізації – це метод, який використовується мережевими джерелами для регулювання швидкості відправлення. Максимальна кількість байтів, які можуть бути передані відправниками TCP непідтвердженими, називається розміром вікна. Оскільки це вікно збільшується і зменшується для регулювання пропускної здатності, воно відоме як ковзне вікно і показано на рисунку 3.2.

У цьому прикладі розмір вікна становить п’ять байтів, і надіслано байти з номерами від 1 до 5. Технічно наступне вікно міститиме байти від 6 до 10, але фактична операція дещо складніша. Оскільки байти з 1 по 3 були підтверджені, але 4 і 5 ще не підтверджені, вони не будуть виключені з наступного вікна; якщо їх не визнають. Замість цього наступне вікно міститиме байти 4 і 5 до того, як воно ковзає, щоб охопити байти 6, 7 і 8. Цей метод корисний у випадку, якщо байти 4 і 5 буде втрачено і відправник повинен повторно передати їх.

3.6 Традиційний контроль перевантаження джерела

Методи керування потоком у TCP використовуються для динамічного налаштування вікна відправників. Критерієм для цього налаштування є наявний буферний простір на приймачі. Завдяки принципам конструкції TCP проміжні маршрутизатори не надсилатимуть підтвердження відправнику. Їм також заборонено регулювати вікно перевантаження cnwd. Таким чином, для індикації перевантаженості на проміжних маршрутизаторах застосовується інший набір механізмів. Ці механізми залежать від втрати пакетів або часу очікування для налаштування вікна перевантаження. Фактичне вікно передачі встановлюється на мінімум початково оговореного вікна та вікна перевантаження.

3.6.1 Повільний старт

Механізм повільного запуску вимагає від відправника повільно починати передачу, а потім збільшувати швидкість надсилання. Максимальна кількість пакетів, які можна надіслати, не визначена заздалегідь. Отже, механізм Slow Start повільно передає пакети, щоб перевірити пропускну здатність мережі. Відправник починає з відправлення одного пакета. Коли ACK повертається, відправник збільшує розмір вікна на одиницю. Під час цієї фази зростання вікна має експоненційний характер [6]. Коли ACK перестають надходити, відправник визначає, що він досяг ємності мережі. Отже, вікно зменшується до одного пакета, що повертає відправника до фази повільного старту.

Повільний старт не є технікою запобігання заторів. Він згладжує швидкість відправлення, щоб запобігти негайному перевантаженню. Перевантаження шлюзу неминуче, і в кінцевому підсумку сегмент потрібно відкинути. Таким чином, цей метод не буде ефективним способом уникнути перевантажень або уникнути довгих затримок доставки даних.

3.6.2 Уникнення перевантажень

Щоб запобігти перевантаженню шлюзу, цей метод обмежує експоненціальне збільшення вікна перевантаження повільного старту. Коли розмір вікна досягає порогового значення, яке називається порогом повільного запуску (ssthresh), вікно збільшується лінійно, на один пакет на отриманий ACK.

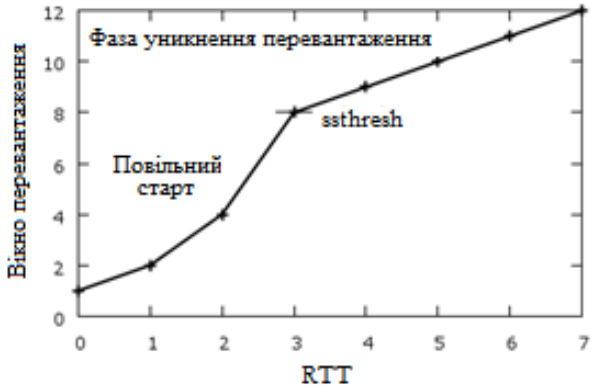


Рисунок 3.3 – Повільний старт і уникнення заторів

3.6.3 Адитивне-збільшення/Мультиплікативне-зменшення (AIMD)

Класичні варіанти TCP використовуються для підтримки попередніх параметрів cwnd і ssthresh. Новіші варіанти TCP, такі як Reno, динамічно коригують ці два параметри. На етапі уникнення перевантажень параметр cwnd збільшується на фіксовану кількість даних, як правило, на один пакет. Це збільшення називається адитивним збільшенням. Тайм-аути або дублікати Acks вдвічі зменшують ssthresh і зменшують cwnd, зазвичай до одного пакета, повертаючи відправника назад до фази повільного старту. Ці два декременти cwnd і ssthresh називаються мультиплікативним зменшенням. У разі послідовних подій тайм-ауту параметр ssthresh буде зменшуватися експоненціально, поки не досягне значення два.

3.6.4 Ініціалізація з’єднання

На початку TCP-з'єднання після тристороннього «рукостискання» відправник TCP починає передачу даних, надсилаючи кількість сегментів, дозволених початковим вікном. З типовим максимальним блоком передачі (MTU) 1500 байт, у початковому вікні надсилається до трьох сегментів. Сегменти в початковому вікні використовуються для завантаження таймера ACK, постійного потоку пакетів ACK, що протікають у зворотному напрямку, який використовується для відправки пакетів після початкового вікна. Початкове вікно є «неконтрольованим» трафіком у тому сенсі, що він надсилається, не знаючи, що пакети залишають мережу.

Відносно нещодавно було запропоновано більше початкове вікно для експериментального використання для зменшення затримки. Більше початкове вікно зменшує кількість RTT, необхідних для завершення коротких передач, які є типовими, наприклад, для веб-об’єктів. Більше початкове вікно дозволяє надсилати до 10 сегментів, не перевіряючи наявність для них ємності. На рисунку 3.4 показано запуск потоку з початковим вікном із трьома і десятьми сегментами. Дослідження веб-пошукового трафіку показує, що початкове вікно з десяти сегментів замість трьох зменшує середню затримку приблизно на 10% [12]. Дослідження вимірювань показує, що від однієї четвертої до однієї п’ятої TCP-з’єднань стають більш агресивними через початкове вікно, що перевищує три сегменти [13].

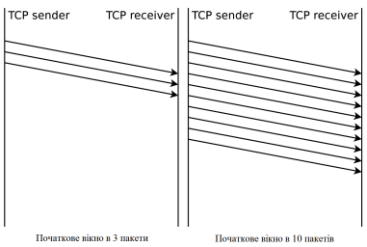


Рисунок 3.4 – Початкове вікно TCP (IW)

Поєднання більшого початкового вікна з паралельними потоками, що запускаються в один і той же момент часу, може призвести до введення великої кількості сегментів у мережу.

Такий великий неконтрольований сплеск пакетів може бути шкідливим для інших потоків у мережі. Якщо розглядати агрегований ефект, вимірювання в мережі доступу показують, що для 5%-12% випадків спостерігається посилений ефект за рахунок комбінованих більших початкових вікон від кількох потоків.

З одним або двома потоками вплив є невеликою, але проблеми стають більш серйозними коли кількість потоків, які починаються одночасно, збільшується.

З пропонованим більшим початковим вікном із десяти сегментів навіть вплив одного потоку перевищує вплив шести паралельних потоків із використанням початкового вікна з трьох сегментів.

Для того, щоб пом’якшити проблеми, викликані надмірними сплесками через більше початкове вікно, було запропоновано початкове розповсюдження. Початкове розповсюдження розділяє пакети Initial Window на окремі, менші перехідні процеси над очікуваним RTT, який був виміряний під час тристороннього «рукостискання».

Видалення початкового вікна дуже шкідливо для інших потоків, оскільки кількість неконтрольованих пакетів, дозволених для відправки, буде набагато більша. Вважається, що навіть розповсюдження пакетів по початковому RTT не в змозі вирішити проблеми, викликані надмірною початковою швидкістю відправлення. Всякий раз, коли відправник порушує принцип збереження пакетів, існує можливість завдати значної шкоди конкуруючому трафіку.

Оскільки відправник початкового вікна не може навіть знати про шкоду, не кажучи вже про реакцію на неї, до того, як шкода вже сталася, єдиний спосіб уникнути шкоди іншому трафіку – це не продовжувати надсилати величезні неконтрольовані початкові пакети.

Використання повільного старту для визначення пропускної здатності з невеликим початковим вікном є хорошим способом обмежити ступінь шкоди, завданої початковим вікном, оскільки воно дає кожному відправнику час для отримання відповіді. Було висловлено твердження, що в інтересах відправника обмежити початкове вікно, щоб уникнути проблем через вичерпання місцевих ресурсів, але цього, ймовірно, недостатньо, оскільки конкуруючий трафік може походити не з того самого джерела.

У такому випадку вичерпані ресурси можуть бути не локальними, а поблизу межі мережі на іншому кінці з'єднання. Таким чином, шкода, заподіяна конкуруючому трафіку, може бути не виміряна відправником, використовуючи надмірну швидкість відправлення. У гіршому випадку відправник може не піклуватися про конкуруючий трафік, але готовий завдати шкоди іншому трафіку, щоб отримати «краще обслуговування» для власного трафіку, і може розглядати можливість монополізувати ресурси.

Альтернативу неконтрольованому надсиланню на початку передачі даних дає старе рішення, відоме як Quick-Start. Відправник TCP за допомогою Quick-Start отримує інформацію про доступні ресурси від маршрутизаторів на наскрізному шляху за допомогою параметрів IP і TCP. На практиці, однак, Quick-Start вимагає підтримки технології від кожного маршрутизатора на наскрізному шляху, і тому його розгортання надзвичайно ускладнене.

3.6.5 Фаза зондування пропускної здатності

Зондування пропускної здатності – це фаза контролю перевантажень на ранніх етапах з'єднання. Метою зондування є визначення пропускної здатності наскрізного шляху, яка зазвичай невідома кінцевому хосту.

Зондування пропускної здатності з самосинхронізацією. Відправник, який використовує зондування пропускної здатності з самосинхронізацією, покладається на вхідні підтвердження (ACK) для визначення часу вихідних

пакетів. На додаток до перевірки доступної ємності, фаза визначення пропускної здатності використовується відправником для завантаження тактового сигналу ACK, який є постійним потоком ACK, що протікають у зворотному напрямку. Таймер ACK передає відправнику кредити, з яких відправник може зробити висновок, скільки пакетів залишило мережу.

Кредити повідомляють відправнику, скільки нових пакетів можна «безпечно» ввести в мережу, тобто без збільшення навантаження на мережу.

На додаток до «безпечних» пакетів, відправнику необхідно ввести більше пакетів, ніж те, що залишило мережу, щоб перевірити швидкість відправлення, більшу, ніж використовується зараз. Відправник припиняє фазу зондування пропускної здатності, як тільки буде виявлений сигнал перевантаження або досягнута досить висока швидкість передачі.

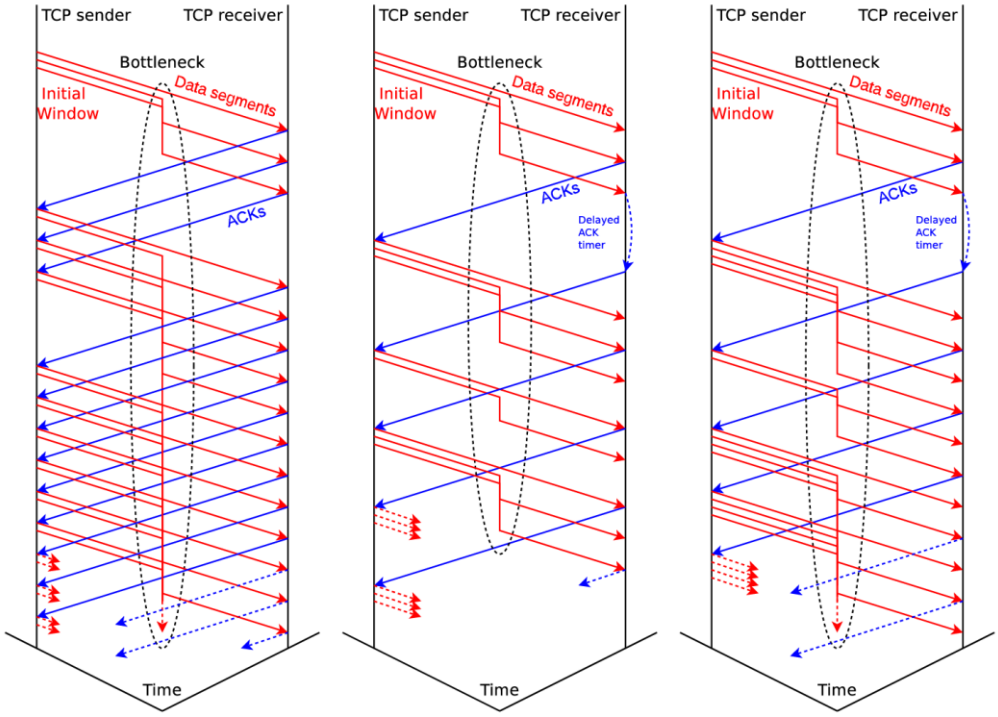
У TCP процес зондування пропускної здатності з самотактовою частотою виконується за допомогою Slow Start. На рисунку 3.5(а) показано процес повільного запуску. Ліва сторона – це відправник TCP, а права – приймач TCP.

Спочатку відправник надсилає три сегменти відповідно до стандартного початкового вікна.

Після початкового вікна відправник чекає на отримання підтверджень. У режимі Slow Start відправник надсилає додатковий пакет при кожному надходженні ACK, крім пакетів, які відповідно до кредитів залишили мережу. Додаткові пакети перевіряють доступну ємність. Кількість пакетів, які можуть бути відправлені, відстежується відправником за допомогою змінної вікна перевантаження (cwnd). Процес повільного запуску призводить до подвоєння вікна перевантаження і, отже, швидкості передачі для кожного RTT, іншими словами, швидкість відправлення та навантаження, пов’язане з потоком, зростають експоненціально.

Експоненціальне збільшення швидкості відправки призводить до експоненційних перехідних процесів навантаження також на мережевих маршрутизаторах. За допомогою зондування пропускної здатності з

самотактовою частотою відправник повинен посилати пакети групами. За групою слідує період простою, доки швидкість відправлення ще не використала всю доступну ємність мережі. Процес повільного запуску дозволяє подвоїти швидкість надсилання за RTT.



(a) Slow Start (b) Slow Start with Delayed ACK (c) ABC corrected Slow Start with Delayed ACK

Рисунок 3.5 – Варіанти TCP Slow-Start

Однак зростання швидкості надсилання може бути меншим через відкладений ACK, який був реалізований десятиліття тому, щоб приблизно вдвічі зменшити кількість пакетів ACK. Менша швидкість для ACK була необхідна, щоб зменшити частоту переривань від мережевого адаптера та пов'язану з цим вартість обробки ACK. Повільний старт із затримкою ACK показано на малюнку 3.5(б).

Коли приймач TCP використовує відкладений ACK, він надсилає одне підтвердження після отримання двох сегментів замість кожного сегмента. Якщо другий сегмент не отримано, одержувач замість цього чекає, поки не

закінчиться таймер відкладеного ACK, щоб надіслати очікуване підтвердження для окремого сегмента. Через відкладені ACK замість подвоєння можливе зростання до 1,5 фактора при типовому повільному запуску, але фактичний коефіцієнт також залежить від розміру таймерів RTT і відкладеного ACK. Якщо приймач не реалізує або не використовує затримку (Delayed ACKs), коефіцієнт зростання в Slow Start дорівнює рівно двом.

Також реалізація TCP-відправника може застосовувати менше збільшення вікна перевантаження через відкладені ACK під час уникнення перевантажень за допомогою відповідного підрахунку байтів (ABC), що зображено на рисунку 3.5(c).

Використання ABC під час повільного запуску дозволено для експериментального використання, однак рекомендовано не збільшувати вікно перевантаження більш ніж на один сегмент на один ACK. На практиці деякі реалізації TCP вирішили використовувати ABC з більшим збільшенням вікна перевантаження також під час повільного запуску (наприклад, реалізація TCP Linux включає механізм на основі ABC, який адаптований до реалізації TCP на основі пакетів).

Повільний старт із відкладеними ACK і ABC показано на рисунку 3.5(c). ABC відновлює коефіцієнт зростання до двох, поки відправник TCP перебуває в повільному запуску.

Під час визначення пропускної здатності з самотактовою частотою ACK, як правило, надходять приблизно зі швидкістю найвужчого каналу на наскрізному шляху, який називається вузьким місцем, оскільки сегменти даних рознесені вузьким каналом.

Це означає, що швидкість надсилання додаткових пакетів, надісланих під час фази визначення пропускної здатності, перевищує швидкість зв’язку вузького місця. Оскільки пакети надходять швидше, ніж можуть бути передані на вузьке місце, на маршрутизаторі перед вузьким місцем утворюється черга.

На рисунку 3.6 показано приклад тимчасових стрибків черги, викликаних повільним запуском TCP, які виникають на маршрутизаторі перед вузьким місцем. Якщо канал ще не насичений поточним навантаженням, черга є лише тимчасовою, і черга буде передана в вихідний канал, як тільки закінчиться коротка група пакетів.

Ці тимчасові стрибки черги збільшуються в амплітуді, оскільки кількість пакетів зростає експоненціально, і під час кожного RTT половина пакетів надсилається з вищою швидкістю, ніж вузьке місце може негайно переслати. Згодом навантаження стає настільки великим, що черга більше не може вичерпатися до того, як надходить наступна група пакетів. Вузьке місце переповнюється, і утворюється постійна черга зі швидко зростаючою довжиною. Припускаючи нескінченну довжину потоку, черга зростає, доки маршрутизатор не сигналізує про перевантаження, що призводить до того, що відправник припиняє фазу визначення пропускної здатності.

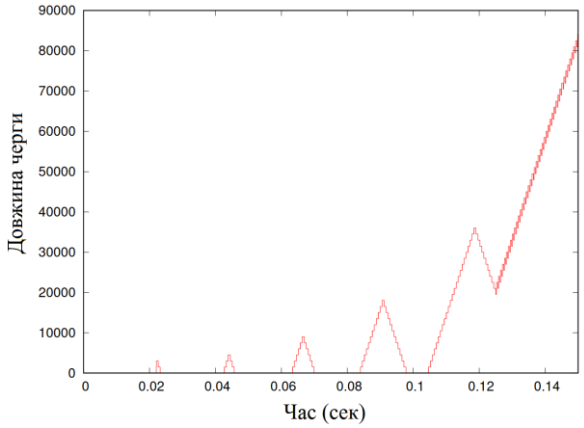


Рисунок 3.6 – Перехідні процеси в черзі під час зондування пропускної здатності

Як альтернатива, зондування пропускної здатності може закінчитися без сигналу перевантаження від мережі. Якщо вікно перевантаження (cwnd) досягає порогу повільного запуску (ssthresh), відправник продовжує працювати в режимі запобігання перевантаженню, але зазвичай стеки TCP

спочатку встановлюють ssthresh дуже велике значення. Крім того, рекомендоване вікно одержувача (rwnd) може встановлювати граничну верхню межу для вікна перевантаження, навіть якщо відправник не отримав жодного сигналу перевантаження.

Зондування пропускної здатності. Друга категорія зондування пропускної здатності заснована на відправленні пакетів за допомогою таймера. Це надсилання на основі таймера відоме як стимулювання.

Метою стимуляції є зменшення кількості сеансів, рівномірно розподіляючи надіслані пакети по виміряному RTT. Пакети надсилаються за допомогою таймера, який часто закінчується, а не коли приходять підтвердження. Якщо пакети зондування пропускної спроможності повторно синхронізуються до такої частини RTT, що швидкість зв’язку вузького місця не буде перевищена через занадто близьку передачу іншого пакету, пакети зондування не утворюють чергу на вузькому місці. Повторне визначення часу передачі можливе, поки вузьке місце ще не насичене. Після насичення більше немає доступних слотів для заповнення за допомогою таймера, оскільки завжди є попередній пакет, надісланий так недавно, що відправник перевищить швидкість зв’язку вузького місця навіть із таймером.

Одним з останніх прикладів динамічного зондування пропускної здатності є контроль за пропускною здатністю вузьких місць і RTT (TCP BBR) для TCP. TCP BBR реалізує темпову відправку всіх пакетів після початкового вікна, згідно з темпом під час визначення пропускної здатності.

Як і у випадку з повільним запуском, швидкість відправлення з TCP BBR ефективно подвоюється за один RTT, але черга не утворюється, поки вузьке місце не стане. Можна використовувати стимуляцію для всіх фаз контролю перевантаження, але він є найкориснішим під час зондування пропускної здатності, оскільки дозволяє підтримувати швидкість надсилання, меншу за швидкість зв’язку вузького місця, доки вузьке місце не буде насичене. Таким чином, перед насиченням не утворюється перехідна черга з частотним зондуванням пропускної здатності.

Однак, якщо фазі визначення пропускної здатності дозволено продовжувати після того, як зв’язок вузького місця стане насиченим, стимуляція не зменшить пікову довжину черги, оскільки пікова довжина черги виникне після насичення зв’язку вузького місця. Тому було б дуже корисно мати можливість завершити зондування пропускної здатності точно в потрібний момент часу, щоб отримати переваги зменшення затримки в черзі від використання стимуляції.

Оцінка пропускної здатності з швидким збільшенням швидкості надсилання. Третя основна категорія зондування пропускної здатності – це оцінка пропускної здатності з швидким збільшенням швидкості надсилання до передбачуваної доступної потужності. Ключова мета в цій категорії – уникнути додаткових RTT, необхідних для «повільного» збільшення швидкості надсилання, і замість цього швидко перейти до оціненої доступної ємності, збільшуючи швидкість надсилання безпосередньо до значення, отриманого з оціночної ємності. Одним з таких підходів є Quick-Start, який виконує узгодження доступної потужності з маршрутизаторами на наскрізному шляху. Однак Quick-Start може здійснювати збільшення швидкості передачі вже під час ініціалізації потоку.

Інші підходи, такі як TCP RAPID, оцінюють пропускну здатність без допомоги маршрутизаторів на шляху. Оцінка пропускної здатності в TCP RAPID базується на алгоритмі pathChirp і має на меті спробувати великий діапазон швидкостей надсилання за дуже короткий період часу.

Як правило, зондування використовує попередню оцінку доступної ємності як середню для швидкості відправлення, але швидкість надсилання, як нижча, так і вища, ніж середня швидкість надсилання, пробується в коротшій перспективі. Тобто, замість того, щоб надсилати пакети якомога рівномірніше, таймер вносить навмисну нерівномірність.

Оскільки час надсилання з високою швидкістю набагато коротший, ніж RTT, висока швидкість надсилання виглядає так само, як коротка група до мережі без створення великої постійної черги, на відміну від двох інших

механізмів зондування пропускної здатності. Вимірювання часу потім використовуються для визначення швидкості відправки, яка перевищила швидкість зв’язку з вузьким місцем. Тільки після того, як відправник отримав позитивне підтвердження від оцінювача пропускної здатності, що здатність підтримувати певну швидкість передачі, ймовірно, доступна, він дійсно завантажує мережу цією швидкістю.

Контроль перевантажень традиційно базується на уникненні перевантажень TCP і працює в стабільному режимі. Однак зростання використання Інтернету призвело до збільшення кількості коротких, непостійних потоків і стало дуже значним фактором перевантаження. Ранні, часто все ще використовувані версії HTTP залежать від паралельних потоків для зменшення затримки веб-транзакції, що призвело до великого налаштування за замовчуванням для кількості паралельних потоків, дозволених веб-браузерами.

Оскільки паралельні потоки у веб-транзакції щільно упаковані в часі, у гіршому випадку, починаючи точно в один і той же момент часу, початкове вікно потоків може спричинити значне накопичення перевантажень. вимірює вплив, який паралельні потоки та запропоноване збільшення початкового вікна до десяти сегментів TCP мають на конкуруючий трафік, чутливий до затримок.

Це показує, що з одним або двома потоками вплив конкуруючих потоків TCP на затримку невеликий при використанні початкового вікна з трьох сегментів.

Якщо кількість потоків вища або використовується початкове вікно з десять незалежно від кількості потоків, затримка, ймовірно, знизить якість конкуруючого інтерактивного медіа-потоку.

Під час запуску потоку відправник використовує фазу зондування пропускної здатності, щоб визначити швидкість передачі, яку він повинен використовувати пізніше в стабільному режимі. Під час фази визначення пропускної здатності TCP використовує повільний запуск, що часто викликає

перехідні процеси експоненційного навантаження на вузькому маршрутизаторі. Експоненційні перехідні процеси навантаження мають значний вплив, особливо поблизу межі мережі, де зазвичай є вузьке місце і де рівень агрегації трафіку нижчий, ніж у маршрутизаторах основної мережі. Оскільки веб-трафік має періоди увімкнення та вимкнення, ці експоненційні перехідні процеси навантаження виникають часто і призводять до швидких коливань навантаження між простою або низьким використанням і перевантаженням.

3.7 Варіанти TCP

Деякі недоліки були виявлені в алгоритмах контролю перевантажень класичних реалізацій TCP. Як обговорювалося раніше, створення заторів було однією з серйозних проблем, викликаних цими недоліками. Тому для подолання цих проблем було запропоновано багато модифікацій TCP.

3.7.1 TCP-Tahoe

Закінчення тайм-ауту в традиційних варіантах TCP інтерпретується як втрачений пакет, який змушує відправника повторно передати втрачені пакети. TCP-Tahoe використовує різні методи для повторної передачі втраченого пакета [4].

Повторювані ACK використовуються для прискорення повторної передачі пакетів до закінчення часу таймера. TCP-Tahoe реагує на повторювані ACK, зменшуючи параметр ssthresh до половини поточного cwnd, а сам cwnd зменшується до одного пакета. Це означає, що мережа перейшла на фазу повільного запуску. Оскільки трафік накопичується, а швидкість відправлення значно збільшується, мережа переходить у фазу уникнення перевантажень.

3.7.2 TCP-Reno

TCP-Tahoe і TCP-Reno [7] використовують повторювані ACK для індикації втрати пакетів. Замість повернення до фази повільного запуску в TCP-Tahoe, TCP-Reno переходить у фазу уникнення перевантажень, зменшуючи вдвічі параметр cwnd і призначаючи новий cwnd параметру ssthresh (cwnd = ssthresh). Ця нова фаза TCP-Reno називається фазою швидкого відновлення. TCP-Reno відстежує всі втрачені пакети та намагається повторно передати їх на цьому етапі. TCP-Reno не переходить у фазу повільного запуску, якщо вікно перевантаження не стає дуже малим через багаторазові втрати пакетів.

3.7.3 TCP-Newreno

TCP NewReno не виходить з фази швидкого відновлення через багаторазові втрати пакетів [14]. Він зменшує вікно перевантаження на кількість підтверджених пакетів мінус один, потім припускає, що пакет після останнього підтвердженого втрачено, і повторно передає його.

3.7.4 TCP-SACK

Selective ACK (SACK) – це розширення, яке можна розглядати разом із NewReno. SACK надсилається відправнику при втраті кількох пакетів. Відправник використовує цей SACK як образ черги одержувача, потім виявляє та повторно передає втрачені пакети, не чекаючи тайм-ауту. Нові пакети не можуть бути надіслані, якщо не підтверджено всі невиконані пакети. Якщо ACK не отримано вчасно, TCP-SACK використовує техніку тайм-ауту Reno для повторної передачі втрачених пакетів. Під час фази швидкого відновлення параметр, який називається pipe, встановлюється на кількість непідтвержених пакетів у транзиті. Швидкість відправлення має

бути меншою за cwnd. Тому відправник надсилає дані лише тоді, коли значення каналу менше, ніж cwnd. Значення каналу збільшується на одиницю з кожним надісланим пакетом і зменшується на один з кожним ACK, включеним до SACK. Канал скорочується вдвічі лише при багаторазових втратах пакетів.

3.7.5 TCP-FACK

TCP Forward Acknowledgement (FACK) призначений для повторної передачі кількох втрачених пакетів. Ця технологія зберігає інформацію про найвищий порядковий номер підтверджених пакетів, використовуючи два параметри: fack і retran-data. Перший зберігає порядковий номер останнього підтвердженого пакета SACK. Другий, підтримує кількість повторно переданих, але ще не підтверджених пакетів. Обсяг непідтверджених даних оцінюється за допомогою рівняння:

$$OSD=EFDS\text{-}fack+retran\text{-}data,$$

де:

- OSD – кількість необроблених даних;
- EFDS – передіслано орієнтовні дані.

FACK не змінює значення cwnd під час фази швидкого відновлення. Замість цього він зберігає рівень непідтверджених даних одним сегментом у межах cwnd. Швидка повторна передача в TCP-FACK відбувається швидше.

3.7.6 TCP- Vegas

Vegas [9] — це варіант TCP, який не зменшує вікно перевантаження, пов’язане із втратою сегмента. Vegas pipe присвоюється значення очікуваної пропускної здатності. Коли мережа стає перевантаженою, фактична

пропускна здатність буде меншою за очікувану, і cwnd коригується відповідно до цієї події. Час відправлення кожного сегмента записується, а відповідний час передачі в обидва боки оцінюється після прибуття ACK. Для кожного з'єднання зберігається базовий час RTT (BaeeRTT).

Цьому параметру призначається значення найнижчого записаного часу передачі в обидва боки серед пакетів, які були відправлені через те саме з'єднання. Очікувана пропускна здатність оцінюється за допомогою рівняння:

$$Ex_{th} = WZ/BaseRTT,$$

де:

- Ex_{th}: очікувана пропускна здатність;
- WZ: поточний розмір вікна перевантаження.

Фактична пропускна здатність розраховується за час передачи туди й назад. Рисунок 3.7 ілюструє алгоритм TCP-Vegas, де Diff – різниця між фактичною та очікуваною пропускною здатністю. Два параметри, α та β, зазвичай використовуються для представлення великої та малої ваги мережевих даних відповідно.

3.7.7 Модифікації TCP-Vegas

З TCP-Vegas була запропонована модифікація повільного запуску незалежно від втрати пакетів.

Вікно перевантаження коригується експоненціально кожного другого часу RTT, а інший час обертання використовується для обчислення параметра Diff. Якщо Diff менший за попередньо встановлений параметр α, Vegas переходить з фази повільного старту до фази уникнення перевантажень. Це забезпечує кращу оцінку доступної пропускної здатності з'єднання.

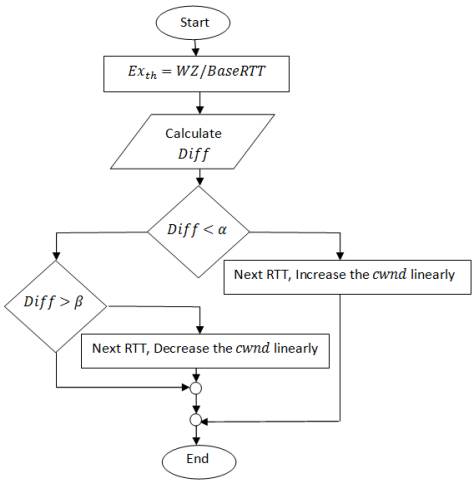


Рисунок 3.7 – Алгоритм TCP Vegas

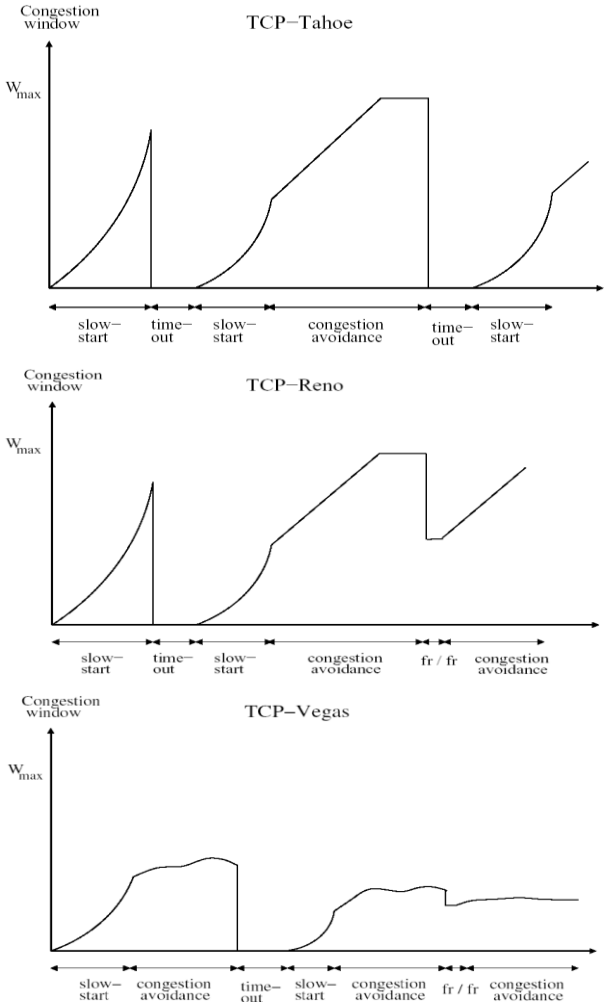


Рисунок 3.8 – Зміна розміру перевантажувального вікна TCP Tahoe, TCP Reno та TCP Vegas

Як новий метод повторної передачі, Vegas повторно передає пакети для кожного дубліката ACK, замість очікування трьох повторюваних ACK.

Ця нова методика застосовна лише в тому випадку, якщо передбачуваний RTT перевищує значення часу очікування. Якщо в одному блоці втрачено багато пакетів або якщо розмір вікна невеликий, то потрійне підтвердження стає неможливою умовою.

Для усунення наявного несправедливого розподілу мережевих ресурсів між різними потоками TCP можна застосовувати алгоритми, які дозволяють маршрутизатору виявляти ті підключення, поведінка яких носить агресивний характер стосовно використання пропускної здатності каналу.

4 АЛГОРИТМИ УПРАВЛІННЯ ЧЕРГАМИ

4.1 Пасивне та активне управління чергами

Мережі TCP застосовують алгоритми керування чергою для контролю перевантаженості проміжних маршрутизаторів. Існує два основних підходи до контролю заторів:

- перший підхід – це пасивне керування чергою (PQM);
- другий підхід – це Active Queue Management (AQM).

PQM – це підхід запобігання перевантажень, який запускає контроль перевантаження після перевантаження буфера шлюзу. AQM – це підхід до запобігання перевантаженню, коли контроль починається до перевантаження буфера шлюзу.

У PQM, якщо пакет скидається на проміжному маршрутизаторі через переповнення буфера, відправники повинні чекати події тайм-ауту, перш ніж вони зменшать швидкість надсилання. У цьому випадку перевантаження виявляється відправниками неявно. Ця реакція відправників називається неявним сповіщенням про перевантаження (ICN). На відміну від цього, явне сповіщення про перевантаження (ECN) – це техніка, яка дозволяє проміжним маршрутизаторам позначати біт перевантаження в заголовку пакета, а не скидати його. Коли пакет, позначений ECN, прибуває до пункту призначення, одержувач встановлює біт перевантаження в заголовку ACK перед тим, як він буде відтворений відправнику. ECN дуже ефективний при помірних випадках заторів. Якщо перевантаження є надмірним, то скидання пакетів вважається більш ефективним рішенням [15].

Зазвичай поріг PQM є межею буфера, але поріг AQM є достатньо меншим за межу буфера. У PQM, коли розмір черги досягає встановленого порогового значення, всі вхідні пакети відкидаються по порядку. І навпаки, AQM скидає пакети випадковим чином.

Багато стратегій прийняли підхід PQM. Drop-From-Front (DFF) – це стратегія PQM, яка скидає пакети з початку черги, щоб звільнити місце для нещодавно надійшли пакетів. Push-Out (PU) – це ще одна реалізація PQM, в якій пакети відкидаються з кінця черги, щоб звільнити місце для нових пакетів, що надійшли. Добре відома стратегія Tail Drop (TD) видаляє всі пакети, що надходять з хвоста черги.

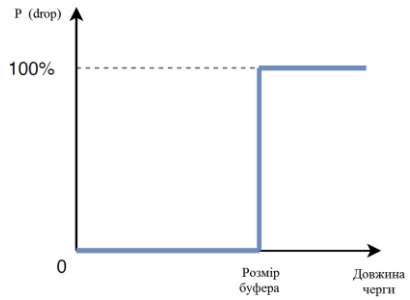


Рисунок 4.1 – Ймовірність відкидання пакетів в Drop-Tail та Drop-From-Front

Рисунок 4.2 – Смуга пропускання з Drop-Tail

Незважаючи на переваги керування чергою та обробки перевантажень, PQM страждає від деяких проблем. Проблеми з повною чергою виникають, коли шлюз постійно надсилає сигнали повної черги джерелам протягом тривалого періоду часу. Блокування відбувається, коли TD дозволяє кільком підключенням монополізувати весь буферний простір. Глобальна синхронізація відбувається, коли всі TCP-відправники знижують швидкість

надсилання одночасно; що призводить до низької пропускної здатності мережі. Ці проблеми докладніше описані в наступних розділах. Стратегії активного керування чергою (AQM), такі як Random Early Detection RED [11], були запропоновані для вирішення проблем стратегій пасивного керування чергою (PQM). AQM покращує пропускну здатність мережі за рахунок зменшення кількості скинутих пакетів у проміжному маршрутизаторі. Крім того, малий розмір черги, який підтримується AQM, зменшує середню затримку мережі [16].

4.2 Класифікація алгоритмів AQM

До впровадження DECbit в 1988 році маршрутизатори підтримували великі буфери, щоб вмістити тимчасові затори, викликані високошвидкісними мережами. Ці великі буфери призвели до великих затримок. Таким чином, необхідно було мінімізувати розмір черги без погіршення пропускної здатності.

Замість використання втрати пакетів для вказівки на перевантаження, було запропоновано виявляти перевантаження за допомогою інших індикаторів. Ці показники включають: передбачуваний час обслуговування каналу вузького місця, зміни пропускної спроможності, а також зміни наскрізної затримки. Проте було показано, що найефективніше місце для виявлення заторів знаходиться в самому проміжному маршрутизаторі.

DECbit [17] є однією з перших стратегій контролю перевантажень, яка застосовувалася до проміжних маршрутизаторів. Коли середній розмір черги перевищує встановлений поріг, DECbit чітко інформує відправників про перевантаження. Він встановлює біт перевантаження в заголовку пакета, щоб інформувати відправників про перевантаження. Для кожного приходу пакета розраховується середній розмір черги для поточного періоду зайнятості шлюзу. Крім того, накладні витрати попередніх періодів зайнятості та простою впливатимуть на обчислення поточного середнього розміру черги.

Якщо обчислене середнє значення більше одиниці, біт перевантаження встановлюється в заголовку пакета. Для кожного другого часу RTT, якщо було позначено половину або більше пакетів з останнього вікна, то джерело зменшує розмір вікна відправлення експоненціально. В іншому випадку вікно збільшується лінійно.

Early Random Drop (ERD) є проміжною стратегією маршрутизатора. Коли розмір черги перевищує попередньо визначений рівень, пакети відкидаються з фіксованою ймовірністю.

Дослідження показують, що ERD вирішує глобальні проблеми синхронізації, викликані TD. Однак ERD не в змозі контролювати користувачів, які погано поводяться. Це може дозволити користувачам, які погано себе ведуть, призначати несправедливу частку пропускної здатності.

IPSource Quench – це стратегія контролю перевантажень, яка виявила використання двох рівнів або порогів зниження. Шлюз надсилає вихідне повідомлення про перевантаження, коли середній розмір черги перевищує перший поріг. Якщо розмір черги перевищує другий поріг, шлюз може відкинути пакети, що надходять, крім пакетів ICMP.

4.2.1 RED

RED (Random Early Detection, або Random Early Drop) використовує випадковий механізм drop для скидання пакетів навіть до переповнення буфера.

Ця випадковість ефективно розподіляє відкидання пакетів між усіма пакетами, що приходять, і, отже, розподіляє їх між усіма потоками.

Цей шаблон скидання пакетів ефективно запобігає глобальній синхронізації. Під час перевантаження ця випадковість дозволяє RED сповіщати конкретне з’єднання, щоб зменшити його вікно приблизно пропорційно частці цього з’єднання в пропускній здатності через маршрутизатор.

А стратегія раннього видалення допомагає постійно резервувати деякий доступний буферний простір черги, щоб поглинати випадкові сплески та уникнути переповнення.

Поки канал не сильно перевантажений, це дозволяє RED бути справедливим у ставленні до потоків «слонів» і «мишей», що частково вирішує проблему несправедливості. З цієї точки зору RED є більшим кроком вперед, ніж Drop-Tail.

RED приймає чотири параметри, а саме, w_q (середня вага черги), th_{min} (мінімальний поріг(minthreshold)), th_{max} (максимальний поріг) і P_{max} (максимальна ймовірність скидання пакету).

RED спочатку обчислює середній розмір черги на основі алгоритму експоненціально зваженого ковзного середнього (EWMA), який визначається рівнянням:

$$avgQ = (1 - w_q)avgQ + w_q q, \tag{4.1}$$

де:

- avgQ – середній розмір черги;
- w_q – вага черги;
- q – миттєвий розмір черги.

Потім RED обчислює ймовірність випадкового відкидання відповідно до рівняння:

$$p(t) = \begin{cases} 0, & avgQ(t) \le th_{\min} \\ \dfrac{P_{\max}}{th_{\max} - th_{\min}}(avgQ(t) - th_{\min}) & th_{\min} < avgQ(t) < th_{\max} \\ 1, & інакше \end{cases}. \tag{4.2}$$

Функція ймовірності відкидання алгоритму RED і значення її параметрів зображені на рисунку 4.3.

Псевдокод алгоритму RED представлено в лістингу 4.1.

Лістинг 4.1 – Псевдокод алгоритму RED

```
Initialization:

avgQ<—0
count<— —1
for each packet arrival
  calculate the new average queue size avgQ:
   if the queue is nonempty
    avgQ<—(1—Wq)•avgQ+Wq•q
   else
    m<—f(time—q_time)
    avgQ<—(1—Wq)^m•avgQ
  if thmin≤avgQ<thmax
    increment count
    calculate probability pa:
      pb<—Pmax(avgQ—thmin)/(thmax—thmin)
      Pa<—pb/(1—count•pb)
    with probability pa:
      mark the arriving packet
      count<—0
  else if thmax≤avgQ
    mark the arriving packet
    count<—0
  else count<— —1
when queue becomes empty
  q_time<—time

Saved Variables:
  avgQ: average queue size
  q_time: start of the queue idle time
  count: packets since last marked packet

Fixed parameters:
  Wq: queue weight
  thmin: minimum threshold for queue
  thmax: maximum threshold for queue
  Pmax: minimum value for pb

Other:
  pa: current packet-marking probability
  q: current queue size
  time: current time
  f(t): a linear function of the time t
```

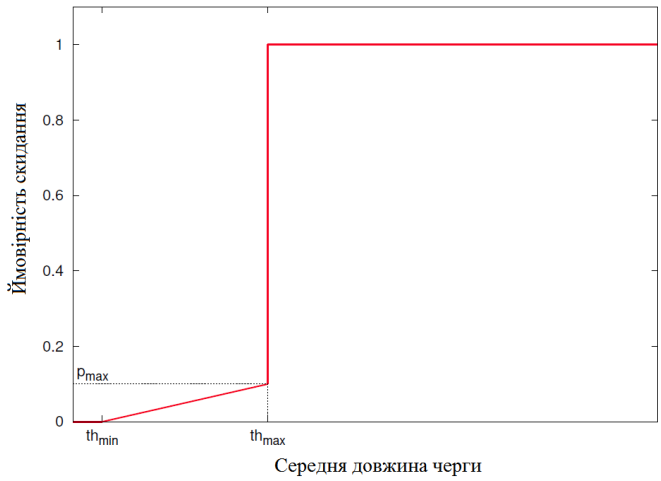


Рисунок 4.3 – Функція ймовірності відкидання пакету в RED, де th_{min} – мінімальний поріг, а th_{max} – максимальний поріг

Алгоритм EWMA, як показано в рівнянні 4.1, функціонує як фільтр низьких частот, який відфільтровує випадкові спалахи на миттєвій довжині черги і, залежно від вибору параметра w_q , робить обчислювану змінну avgQ більш гладкою. І на основі значення avgQ розраховується ймовірність відкидання p(t), яка також є більш гладкою. Таким чином, RED здатний відфільтрувати випадкові спалахи трафіку, що надходить, зберігає середній розмір черги низьким, залишаючи деякий буферний простір для поглинання випадкових спалахів.

Середній розмір черги avgQ відображає рівень перевантаження каналу, і, застосовуючи механізм EWMA для обчислення avgQ, випадкові коливання миттєвої довжини черги ефективно згладжуються. Тому змінну p(t), розрахована за рівнянням 4.2 можна інтерпретувати як «ціну», пропорційну середньому розміру черги або рівню перевантаженості. Коли рівень перевантаженості вищий, джерела TCP за статистикою платять вищу ціну або втрачають більше пакетів пропорційно їх частці пропускної здатності через маршрутизатор, а натомість алгоритм уникнення перевантажень і контролю, що знаходиться в джерелах TCP, допомогає утримувати швидкість відправлення. Таким чином, алгоритм керування активною чергою RED і

механізм уникнення перевантажень і контролю всередині відправника TCP утворюють систему керування зворотним зв’язком. А за допомогою контролю пари змінних (p, avgQ) можна ефективно керувати рівноважною робочою точкою цієї системи керування зі зворотним зв’язком і вибирати багато варіантів керування.

4.2.2 Gentle RED (подвійний RED)

Однак одна з проблем RED полягає в тому, що його функція ймовірності скидання пакетів не є безперервною від нуля до одиниці. Це викликає проблему нестабільності, коли канал стає настільки перевантаженим, що середній розмір черги наближається до максимального порогу th_{max}. Рекомендації щодо встановлення одного з параметрів RED P_{max} (максимальна ймовірність скидання) неодноразово змінювалися в міру розвитку Інтернету та зміни характеристик трафіку. Рекомендується модифікований варіант RED, який називається подвійним RED (DSRED). Коли канал стає сильно перевантаженим, середній розмір черги RED наближається до його максимального порогу th_{max}, і його ймовірність скидання коливається між його максимальною ймовірністю скидання P_{max} і одиницею. І зазвичай P_{max} набагато менше одиниці. Це викликає нестабільність і може змусити RED вести себе більше як Drop-Tail в крайніх випадках. Для вирішення цієї проблеми рекомендується подвійний схил в RED.

В оригінальному RED є три основні проблеми:

- низька пропускна здатність;
- велика затримка/джиттер;
- викликання нестабільності в мережах.

Gentle RED забезпечує безперервну функцію ймовірності скидання від нуля до одиниці. Gentle RED має очевидний прогрес у порівнянні з вихідним RED за пропускною здатністю та затримкою/джиттером. Будучи

«покращеною» версією оригінального RED, Gentle RED також приймає ті самі чотири параметри, а саме, w_q (середня вага черги), th_{min} (мінімальний поріг), th_{max} (максимальний поріг) і P_{max} (максимальна ймовірність скидання пакету).

Функція ймовірності скидання DSRED визначається рівнянням:

$$p(t) = \begin{cases} 0, & avgQ(t) \le th_{\min} \\ \dfrac{P_{\max}}{th_{\max} - th_{\min}}(avgQ(t) - th_{\min}) & th_{\min} < avgQ(t) < th_{\max} \\ \dfrac{1 - P_{\max}}{th_{\max}}(avgQ(t) - th_{\max}) & th_{\max} < avgQ(t) < 2 \cdot th_{\max} \\ 1, & \textit{інакше} \end{cases}. \quad (4.3)$$

Функція ймовірності скидання Gentle RED і значення її параметрів можна зобразити на рисунку 4.4.

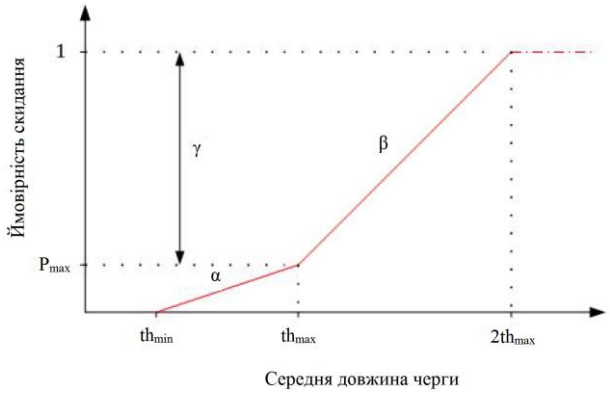


Рисунок 4.4 – Функція ймовірності скидання пакетів в Gentle RED

Ідея подвійного нахилу RED полягає в тому, що загальна функція ймовірності скидання від th_{min} до $2xth_{max}$ визначається двома лінійними сегментами з нахилом α і β відповідно. Нахили для цих двох лінійних сегментів є взаємодоповнювальними і регулюються перемикачом режимів.

4.2.3 BLUE алгоритм

Сучасні методи, такі як RED, мають притаманні недоліки в оцінці серйозності заторів. Усі ці алгоритми, такі як RED, розглядають наявність постійної черги як ознаку перевантаженості. Крім того, вони розглядають більшу довжину черги як ознаку більш серйозних заторів.

Наявність постійної черги дійсно вказує на перевантаження, однак довжина постійної черги дає дуже мало інформації про серйозність перевантаженості. Як наслідок, RED вимагає широкого діапазону параметрів для правильної роботи в різних сценаріях перевантажень.

Хоча RED може досягти ідеальної робочої точки, він може зробити це лише тоді, коли має достатню кількість буферного простору та його правильно параметризовано.

BLUE відкидає довжину черги моніторингу, натомість використовує події переповнення черги та порожні події для управління перевантаженням. BLUE зберігає єдину ймовірність, яку він використовує для позначення (або скидання) пакетів, коли вони стоять у черзі.

Якщо черга постійно відкидає пакети через переповнення буфера, BLUE збільшує ймовірність скидання або позначки, таким чином збільшуючи швидкість, з якою він надсилає сповіщення про перевантаження. І навпаки, якщо черга стає порожньою або якщо з’єднання неактивне, BLUE зменшує ймовірність його скидання або позначки.

Щоб не відставати від зміни рівня перевантаження, BLUE використовує дуже малий буфер черги, щоб переповнення та недоповнення відбувалися частіше, на основі чого BLUE оцінює зміни серйозності перевантажень і відповідно оновлює ймовірність скидання/відмітки.

Алгоритм BLUE добре працює (має низькі втрати пакетів і затримку) за певних сценаріїв, навіть коли працює з невеликим буфером черги. Псевдокод BLUE алгоритму показано в лістингу 4.2.

Лістинг 4.2 – Псевдокод алгоритму BLUE

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

В лістингу 4.2 L – максимальний розмір буфера, який у деяких відношеннях еквівалентний RED параметру th_{max}, freeze_time – мінімальний інтервал часу між двома послідовними оновленнями ймовірності скидання або позначки p_m, а d1 і d2 – два BLUE параметри і обидва є малі константи, p_m – це єдина ймовірність скинути або позначити пакети, коли вони стоять у черзі.

Найбільш корисною особливістю BLUE є те, що BLUE може працювати дуже добре (мати низькі втрати пакетів і затримки), навіть коли працює з дуже малим буфером. BLUE збільшує ймовірність скидання або позначки пакета лише при переповненні буфера і зменшує цю ймовірність лише тоді, коли буфер порожній. Крім того, BLUE алгоритм дуже простий. Але це змушує задуматися про компроміси для такого простого алгоритму для досягнення такої «високої» продуктивності.

Інтуїтивно BLUE не повинен реагувати швидше, ніж RED, коли стикається з надходженням сплесків, оскільки він реагує лише при переповненні або недостатньому буфері. Але це може статися, якщо RED має надзвичайно малу середню вагу черги w_q і набагато більший буфер, ніж BLUE, оскільки з цими налаштуваннями параметрів середній розмір черги RED повільніше реагує на коливання черги. Справа в тому, що для надзвичайно малих значень w_q RED алгоритм відокремлюється від довжини черги і діє так, як BLUE, за винятком значно більшого буфера. І через

надзвичайно малий буфер BLUE, на відміну від RED, затримка в черзі BLUE помітно зменшується. Однак склад інтернет-трафіку, а отже, і характеристики трафіку завжди сильно змінюються навіть у різні години доби. Обмежений буферний простір означатиме недостатню здатність вмістити ці зміни та сплески трафіку, а отже, і зниження пропускної здатності. І, обмінюючи таким чином пропускну здатність на затримку, BLUE може постійно переповнюватися при високих сплесках, і глобальна синхронізація може відбуватися знову й легше, якщо BLUE не встановить правильні параметри.

Існує ще одна відмінність між алгоритмами RED і BLUE. Ймовірність скидання постійно змінюється RED, але змінюється поетапно в BLUE. І ця зміна відбувається лише тоді, коли буфер переповнюється або його недостатньо, що в цілому є ознакою невідповідності між навантаженням і ємністю каналу. Очевидно, BLUE не намагається виявити перевантаження на його початковій стадії, навпаки, він чекає, поки перевантаження посилиться, щоб переповнити буфер або зменшиться, щоб зповільнить буфер, перш ніж вживати подальших дій. Результатом зменшення затримки є яскрава сторона BLUE алгоритму. Крім того, спостереження за заповненістю черги не допомагає визначити статус заторів за природними коливаннями. Проте, для контролю перевантажень і покращення загальної продуктивності, включаючи пропускну здатність, коефіцієнт втрат пакетів і затримку, перевантаження слід виявляти на початкових етапах і негайно вживати належних заходів контролю перевантажень.

4.2.4 Adaptive RED

Незважаючи на велике поширення, RED має ще одну проблему. Коли канал незначно перевантажений або максимальна ймовірність скидання P_{max} висока, середній розмір черги близький до мінімального порогу th_{min}, коли канал сильно перевантажений або P_{max} низька, робоча точка та середній

розмір черги близькі до th_{max} або навіть вище th_{max}. Як результат, середня затримка в черзі RED чутлива до навантаження на трафік і параметрів RED, і тому не є передбачуваною заздалегідь. Затримку, яка є основною складовою якості обслуговування (QoS), що надається своїм клієнтам, оператори мережі, природно, хотіли б мати приблизну апріорну оцінку середніх затримок у своїх перевантажених маршрутизаторах. Щоб досягти таких передбачуваних середніх затримок, RED вимагатиме постійно налаштовувати свої параметри, щоб вони були сумісними з поточними умовами трафіку, що непрактично для оригінального RED.

Adaptive RED зберігає основну структуру RED; Основна відмінність Adaptive RED полягає в тому, щоб автоматично налаштовувати параметр RED P_{max}, щоб зберегти середній розмір черги між th_{max} та th_{max}.

Рекомендації щодо цього можна підсумувати як:

- встановлення ваги черги w_q автоматично на основі пропускної здатності каналу;
- адаптація P_{max} у відповідь на виміряну довжину черги.

Мета цього полягає в тому, щоб усунути необхідність ретельного налаштування цих параметрів RED, щоб можна було постійно досягати високої продуктивності. Параметр w_q встановлюється в Adaptive RED як функція пропускної спроможності каналу і визначається рівнянням:

$$w_q = 1 - e^{-\frac{1}{c}}, \tag{4.4}$$

де c – пропускна здатність каналу в пакетах/секунду, обчислена для пакетів заданого розміру за замовчуванням.

В лістингу 4.3 показано, що адаптація P_{max} відбувається відносно повільно і рідко. Це дозволяє динаміці RED – адаптації ймовірності падіння пакетів у відповідь на зміни середнього розміру черги домінувати на менших часових масштабах.

Лістинг 4.3 – Псевдокод алгоритму Adaptive RED

```
Every interval seconds:
  if (avgQ>target and Pmax<0.5)
    increase Pmax:
    Pmax<-Pmax+α;
  elseif (avgQ<target and Pmax≥0.01)
    decrease Pmax:
    Pmax<-Pmax*β;
Variables:
  avgQ: average queue size
Fixed parameters:
  interval: time; 0.5 seconds
  target: target for avgQ; [thmin+0.4*(thmax-thmin),
          thmin+0.6*(thmax-thmin)].
  a: increment; min(0.01, Pmax/4)
  p: decrease factor; 0.9
```

Таким чином, у природі Adaptive RED вносить такі зміни до оригінального RED:

- шляхом введення адаптивного методу в обробку P_{max}, у більш тривалому часовому масштабі середній розмір черги завжди стабілізується навколо (th_{min}+th_{max}), а проблема налаштування параметрів RED частково вирішена;

- налаштування параметра w_q в Adaptive RED свідчить про те, що правильне налаштування w_q корелятивне з пропускною здатністю каналу с і може бути розраховано на основі рівняння 4.4;

- Adaptive RED використовує широко прийняте рекомендоване значення $th_{max}=3th_{min}$.

Таким чином, параметри RED w_q, th_{min}, th_{max} і P_{max} можна автоматично налаштувати для адаптації до постійно змінного навантаження мережі.

Зважений RED (WRED) дозволяє диференціювати відповідно до класу трафіку, маючи окремі пороги і вагу RED для кожного класу. Налаштувавши клас трафіку з більшими пороговими значеннями, WRED дозволяє надавати цьому класу трафіку пріоритет над іншими класами трафіку.

Хоча на інтуїтивному рівні відно, що алгоритми AQM включають занадто спрощену логіку оцінки навантаження на основі довжини черги.

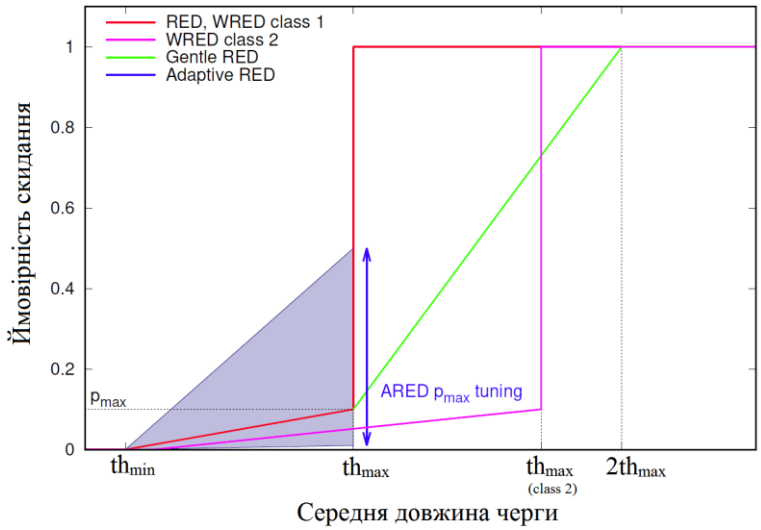


Рисунок 4.5 – Імовірність скидання пакетів з різними варіантами RED

4.2.5 Стабілізований RED

Алгоритм Stabilized RED [16] є ще одним варіантом алгоритму RED. На відміну від більшості існуючих активних алгоритмів керування чергою, які контролюють лише зайнятість черги, алгоритм Stabilized RED інтегрує новий механізм вимірювання в реальному часі, який збирає інформацію про характеристики трафіку. Однак він не призначений для зменшення затримок. Механізм вимірювання використовується для вимірювання кількості активних з'єднань, а мета розробки алгоритму – стабілізувати довжину черги та вирішити проблему несправедливості щодо потоків із тривалими RTT.

Сам по собі моніторинг довжини черги не може відрізнити зміни статусу перевантаженості від природних коливань черги. З цієї причини додатковий механізм вимірювання інформації про характеристики трафіку може бути ключем до відмінності стану перевантаженості від природних коливань черги, і може бути розгорнутий з метою забезпечення вторинного засобу формування політики, такої як контроль затримок.

Коли інші фактори, такі як пропускна здатність, час RTT, залишаються незмінними, кількість ефективних TCP-з'єднань, що конкурують за ту саму пропускну здатність, визначає навантаження.

І коли кількість з’єднань, які використовують ланку, змінюється, змінюється навантаження і змінюється рівноважна робоча точка RED в термінах (p, avgQ). І завдяки схемі AIMD вікна TCP, природа трафіку TCP полягає в тому, щоб захоплювати собі більше пропускної здатності, коли це можливо. Таким чином, більше TCP-з’єднань означає більше джерел, що змагаються за ту саму пропускну здатність, що зазвичай є більш домінуючим фактором для збільшення навантаження на канал, ніж інші пов’язані фактори, такі як час передачі, розмір файлу тощо. А в IP-мережі кількість з’єднань постійно змінюється, а заповнюваність черги постійно змінюється через цю зміну навантаження та природних коливань швидкості передачі даних, що ускладнює стабілізацію черги.

Stabilized RED рекомендується для того, щоб відокремити зміни кількості підключень від зайнятості черги. В широкому діапазоні рівнів навантаження Stabilized RED може стабілізувати свою чергу на рівні, незалежному від кількості активних з’єднань. Stabilized RED робить це, оцінюючи кількість активних з’єднань або потоків. Простішим способом порівняння пакета, що надійшов, з останнім іншим пакетом було б порівняти його з пакетом, який все ще знаходиться в черзі.

Однак буфер черги зазвичай невеликий, що унеможливлює порівняння пакетів, розміщених на більшій відстані один від одного. Щоб надати системі довшу пам’ять про історичне минуле, Stabilized RED доповнює інформацію в буфері за допомогою «списку зомбі», що складається з M елементів, який містить більше інформації про трафік, ніж це можливо в черзі. Його можна розглядати як список M нещодавно побачених потоків. Цей список зомбі або кеш потоку невеликий, і підтримувати цей список не те саме, що підтримувати стан для кожного потоку, який не є масштабованим і споживає багато ресурсів. Він також відрізняється від буфера черги тим, що містить лише необхідну інформацію про потоки, не включаючи дані корисного навантаження, що доставляються в пакетах. Потоки в списку зомбі називаються «зомбі».

Список зомбі спочатку порожній. У міру надходження пакетів, поки список не заповнений, для кожного пакета, що надходить, до списку додається ідентифікатор потоку пакетів (адреса джерела, адреса призначення тощо). Коли список зомбі заповнений, він працює наступним чином: щоразу, коли надходить пакет, він порівнюється з випадково вибраним зомбі в списку зомбі. Якщо ідентифікатор потоку пакета, що надходить, збігається з ідентифікатором зомбі, оголошується «хіт».

Якщо вони не є одним потоком, оголошується «без звернення». У цьому випадку з імовірністю p_r ідентифікатор потоку пакета, що надходить, замінює зомбі, вибраного для порівняння.

З імовірністю 1-p_r , список зомбі не зміниться. Незалежно від того, було звернення чи ні, пакет може бути відкинутий, якщо зайнятість буфера черги така, що система перебуває в режимі випадкового скидання. Імовірність скидання може залежати від того, було влучення чи ні.

Оцінка $P_z(t)$ підтримується для частоти попадань приблизно в момент прибуття t-го пакета в буфер. Для t-го пакета нехай:

$$Hit(t) = \begin{cases} 0 & \textit{якщо нема збігу} \\ 1 & \textit{збіг є} \end{cases} \tag{4.5}$$

$$P_z(t) = (1-\alpha) \cdot P_1(t-1) + \alpha \cdot Hit(t), \tag{4.6}$$

де $0<\alpha<1$ та:

$$\alpha \sim \frac{P_r}{M}, \tag{4.7}$$

де P_r – ймовірність того, що пакет, що надходить, замінить зомбі, вибраного для порівняння, а M – розмір списку зомбі з точки зору кількості зомбі.

Тоді $P_z(t)$ є оцінкою частоти звернень для приблизно останніх пакетів M/P_r перед пакетом t. Це також можна розглядати як ймовірність того, що пакет, що надходить, має збіг.

Пропозиція: $P_z(t)^{-1}$ є хорошою оцінкою ефективної кількості активних потоків за час незадовго до прибуття пакета t.

На основі цієї оцінки ефективної кількості з'єднань розроблено алгоритм Stabilized RED налаштовує ймовірність падіння пакетів з метою стабілізації черги та вирішення RED несправедливість по відношенню до потоків із довшим часом обертання.

Проте є кілька проблем із алгоритмом Stabilized RED. По-перше, його оцінка кількості зв'язків далека від досконалості. Оцінка хороша лише в симетричних випадках. Кількість підключень є лише одним із багатьох факторів, які впливають на навантаження мережі. Інші фактори, такі як RTT, розподіл розміру файлу, також можуть значно впливати на навантаження і, отже, на оцінку кількості з'єднань. В осяжному майбутньому Інтернет зростатиме в геометричній прогресії з кількістю додатків і обсягом трафіку, список зомбі завжди і лише зможе містити невелику вибірку всіх потоків. Отже, це може лише частково вирішити проблему несправедливості; і зусилля в цьому напрямку виявляться неповним рішенням.

Однак Stabilized RED є, якщо не першим, алгоритмом, який використовує масштабований механізм вимірювання в реальному часі для моніторингу інформації про характеристики трафіку. Отримане вимірювання або оцінка кількості активних з'єднань далі використовується як допоміжний засіб для коригування шаблону відкидання або відмітки пакетів.

5 АВТОМАТИЗОВАНА АДАПТАЦІЯ RED

Новий алгоритм управління чергою має намір посилити контроль затримок і зменшити затримку в моменти перевантажень. Для досягнення цієї мети в цьому алгоритмі об’єднано три необхідні компоненти, а саме масштабований механізм вимірювання в реальному часі, адаптивна техніка та контроль робочої точки, орієнтований на затримку.

AARED алгоритм інтегрує масштабований механізм вимірювання в реальному часі, який називається списком зомбі, у поточну версію алгоритму Adaptive RED. Також розширюється оригінальний механізм списку зомбі, щоб зробити його здатним вимірювати характеристики трафіку, пов’язані із затримкою.

Таким чином, на відміну від попередніх варіантів RED, новому алгоритму потрібно контролювати не тільки довжину черги пакетів, але також необхідно виявляти зміни характеристик трафіку, пов’язані із затримкою, такі як середнє значення та дисперсія розподілу швидкості прибуття.

На основі цього додаткового вимірювання алгоритм AARED повинен адаптивно коригувати параметри RED і керувати рівноважною робочою точкою системи керування зворотним зв’язком, щоб рухатися до цільової точки, що зменшує затримку і в той же час зберігає без втрати інші показники продуктивності, такі як продуктивність і коефіцієнт скидання пакетів.

5.1 Механізм розширеного списку зомбі

Щоб виміряти інформацію про характеристики трафіку необхідно внести зміни в оригінальний механізм списку зомбі, і механізм списку зомбі працює наступним чином.

Замість того, щоб зберігати фіксовану кількість M слотів зомбі в списку зомбі, зберігається фіксована кількість M' байтів кешу пам’яті в списку зомбі, який також називаємо zlSize (розмір списку зомбі). Отже, розмір списку зомбі є фіксованим і не залежить від різних розмірів пакетів різних потоків.

Коли механізм вирішує, що щойно надійшовший пакет може увійти до списку зомбі, зі списку зомбі видаляється еквівалентний байт зомбі. Таким чином, пам’ять списку зомбі про історичне минуле відносно фіксується, а підсумовування байтів списку зомбі зберігається таким чином, що воно завжди коливається навколо M'.

Під час надходження кожного пакета останній порівнюється із випадково вибраним зомбі зі списку зомбі, щоб побачити, чи відбувається «влучення», і відповідно оновлюється ймовірність співпадіння зомбі $P_z(t)$, де Z символізує зомбі. Незалежно від того, є влучення чи ні, генерується випадкова змінна і порівнюється з ймовірністю заміни P_r, щобі вирішити, чи повинен щойно надійшовший пакет увійти до списку зомбі.

Замість того, щоб замінювати випадково вибраного та порівняного зомбі, оновлюється список зомбі на основі правила «перший прийшов першим вийшов». Тому не буде шансів, що певний зомбі в списку не буде замінено протягом тривалого часу, що негативно позначиться на правдивості результату вимірювання.

На додаток до оригінального механізму алгоритма вимірювання списку зомбі (який порівнює пакет, що надходить, із випадково вибраним зомбі зі списку зомбі, щоб побачити, чи є «влучення», і відповідно оновити ймовірність зомбі $P_z(t)$, де Z символізує зомбі), продовжуємо порівнювати пакет, що прибув, з іншими зомбі в списку зомбі, щоб побачити, чи є «звернення до списку» і відповідно оновлюємо ймовірність «влучення» в список $P_L(t)$, де L символізує список. На додаток до оригінального механізму зомбі-списку, під час надходження кожного пакета враховується в зомбі-списку кількість окремих потоків S(t).

Подібно до змінної $P_z(t)$, яка підтримується рівнянням 4.6, змінна $P_L(t)$ також підтримується. Для t-го пакета нехай:

$$Hit(t) = \begin{cases} 0 & \textit{якщо в списку нема збігу} \\ 1 & \textit{збіг є} \end{cases} \tag{5.1}$$

$$P_L(t) = (1-\alpha) \cdot P_L(t-1) + \alpha \cdot Hit(t), \tag{5.2}$$

де $0<\alpha<1$ та:

$$\alpha \sim \frac{AvPktSize}{M'}, \tag{5.3}$$

де середній AvPktSize – це середній розмір пакету, виміряний в байтах, а M' – це байти кешу пам’яті в списку зомбі.

Тоді $P_L(t)$ є оцінкою частоти звернень у список приблизно для останніх пакетів M'/AvPktSize перед пакетом t. Оскільки ймовірність влучення зомбі $P_z(t)$ є оцінкою частоти зомбі-влучення, тобто пакет, що надходить, відповідає випадково вибраному зомбі, під час перевантаження, пакети, що прибувають, належать до потоку i з ймовірністю π_i, де:

$$\pi_i = \frac{x_i}{\sum_i x_i}, \tag{5.4}$$

та

$$\sum_i x_i = C, \tag{5.5}$$

де C – пропускна здатність, а x_i – швидкість потоку i в пакетах.

Імовірність влучення зомбі визначається як:

$$P_Z(t) = \sum_i \pi_i^2 = \frac{\sum_i x_i^2}{\left(\sum_i x_i\right)^2}, \tag{5.6}$$

$$\sum_i \pi_i^2 = \frac{1}{F} \cdot \frac{\bar{x^2}}{\left(\bar{x}\right)^2},$$

де F – справжня загальна кількість потоків.

Оскільки ймовірність звернення до списку – це оцінка частоти звернення до списку, тобто пакет, що надходить, відповідає будь-якому зомбі у списку, а S(t) – це кількість окремих потоків у списку зомбі, загальна кількість потоків можна оцінити як:

$$F = \frac{S(t)}{P_L(t)}. \tag{5.7}$$

При перевантаження середня швидкість приходу всіх потоків дорівнює:

$$\bar{x} = \frac{C}{F} = C\frac{P_L(t)}{S(t)}. \tag{5.8}$$

Дисперсія швидкості надходження всіх потоків, що надходять на маршрутизатор, дорівнює:

$$\sigma = C \cdot \sqrt{\frac{P_L(t)}{S(t)} \cdot \left(P_Z(t) - \frac{P_L(t)}{S(t)}\right)}. \tag{5.9}$$

Тепер механізм списку зомбі здатний вимірювати в режимі реального часу середнє та дисперсію розподілу швидкості прибуття. Два важливі фактори розподілу швидкості, а саме його середнє значення та дисперсія, мають кореляцію з інформацією про сплеск трафіку, яку можна використати, щоб знайти належну можливість зменшити затримку.

5.2 AARED

Як згадувалося раніше, лише спостереження за зайнятістю черги не може відрізнити зміни серйозності перевантажень від природних коливань швидкості передачі даних. Спостереження лише за зайнятістю черги здається тим більш неадекватним, коли необхідно контролювати затримки в активних алгоритмах керування чергою, особливо коли є рясний трафік, такий як трафік TCP/IP.

Розподіл швидкості надходження всіх джерел містить більше інформації про характеристики трафіку, ніж коливання заповнюваності черги. І на основі механізму вимірювання розширеного списку зомбі можна виміряти два його найважливіші параметри, а саме середнє значення та дисперсію.

В алгоритмі використовується адаптивний механізм в алгоритмі Adaptive RED для оновлення максимальної ймовірності падіння P_{max}. Крім того, вводиться алгоритм оновлення мінімального порогового значення th_{min}, щоб контролювати робочу точку і, отже, затримку черги. Оновлення th_{min} описано нижче. Використовується механізм списку зомбі для вимірювання дисперсії швидкості надходження потоків та обчислюється її відхилення за рівнянням 5.9.

Подібно до співвідношення між миттєвим розміром черги та середнім розміром черги, виміряна дисперсія швидкості також значно коливається. Щоб згладити це, необхідно використовувати його експоненціально зважене ковзне середнє (EWMA).

Нехай вага – AvPktSize/zlSize, де AvPktSize – це середній розмір пакету, а zlSize – це розмір списку зомбі в байтах. Обчислюється експоненціально зважене ковзне середнє відхилення, weightRateDev. Таким чином, weightRateDev зможе простежити середнє значення виміряного відхилення швидкості минулої історії, що відповідає розміру списку зомбі:

$$WeightRateDev = w \cdot WeightRateDev + (1 - w) \cdot \sigma, \tag{5.10}$$

де w – вага і w=AvPktSize/zlSize.

Обчислюється цільова робоча точка за допомогою рівняння 5.11.

Встановлюється розташування робочої точки за допомогою RED параметра th_{min}. Переміщення розташування робочої точки до цільової робочої точки виконується покроково, зі швидкістю 5% th_{min} кожні 0,5 секунди.

$$T\arg OP = \min\left[th_{min} \cdot \left(1 + \frac{WeightRateDev}{RateDev\,\mathrm{Ref}} \right) \cdot th_{max} \right], \tag{5.11}$$

де RateDevRef є константою, значення якої встановлено зі значенням досвіду 1,50E+05 для сценарію пропускної здатності 2 Мбіт/с.

Розгляд екстремальних випадків допомагає зрозуміти рівняння 5.11. Коли швидкість прибуття всіх потоків є рівномірною і присутні надзвичайно малі коливання, виміряна дисперсія швидкості наближається до нуля, і рівняння 5.11 дасть th_{min}, що означає, що алгоритм буде керувати робочою точкою і підштовхувати її до мінімального значення th_{min}. У цьому випадку без поривів, коротша черга задовольнить вимогу про пропускну здатність і ймовірність скидання, і в той же час спричинить найменшу затримку черги. Коли є лише один потік з надзвичайно високою швидкістю пакетів, виміряна дисперсія швидкості наблизиться до максимуму, у цьому випадку рівняння 5.11 дасть th_{max}, що означає, що алгоритм буде керувати робочою точкою та

підштовхувати її до максимального значення th_{max}. У цьому надзвичайно швидкому випадку найкращий спосіб підтримувати високу пропускну здатність – використовувати максимально дозволений буферний простір для черги бурхливого трафіку.

Коли трафік нормальний, відношення зваженого відхилення швидкості до контрольного значення дає одиницю, рівняння 5.11 дасть $(th_{min} + th_{max})/2$, і в цьому випадку алгоритм буде керувати робочою точкою і штовхати її до середини між мінімальним і максимальним порогом. У цьому випадку алгоритм веде себе дуже схоже на алгоритм Adaptive RED.

Лістинг 5.1 – Псевдокод алгоритму AARED

```
Every packet arrival:
Targ_OP=thmin*(1+WeighRateDev/RateDevRef)

Every interval seconds:
if (targ_OP>(1+γ)*prev_OP)
  increase thmin:
  thmin<-thmin*(1+γ);
else if (targ_OP<(1-γ)*prev_OP)
  decrease thmin:
  thmin<- thmin *(1-γ);
else
thmin remain unchanged;
set other queue thresholds relative to updated thmin:
thmax<-3*thmin;
Qmax<-50*AvPktSize;

Variables:
WeightRateDev: measured by the zombie list mechanism

Fixed parameters:
interval: time; 0.5 second
RateDevRef: reference value; 1.5E+05
γ: change factor; 0.05
```

Для того, щоб досить швидко відстежувати зміни характеристик трафіку, не викликаючи небажаних коливань, алгоритм використовує той самий період адаптації 0,5 секунди, що й алгоритм Adaptive RED. Під час моделювання було пімічено, що вибір розміру списку зомбі zlSize також має

значення. Якщо його значення занадто велике, це не тільки призводить до непотрібних накладних витрат на обчислення, але й розмиває вибірки пакетів нещодавнього минулого. Наприклад, якщо zlSize занадто великий, термін дії деяких потоків уже закінчився, але їх пакети все ще можуть залишатися в списку зомбі в очікуванні заміни новими потоками. Це впливає на вимірювання характеристик трафіку та робить його занадто повільним для відстеження справжніх характеристик трафіку. Взагалі, немає строгих обмежень для налаштування zlSize. Алгоритм відстежує зміни трафіку за останні 0,5 секунди, і послідовно адаптує його параметри.

Алгоритм дотримується рекомендацій Adaptive RED щодо встановлення th_{max} у три рази більше th_{min}, w_q як функції від ємності каналу C і ємності буфера черги Q_{max} на 50*AvPktSize.

Алгоритм ґрунтується на виміряному середньому довжини черги та дисперсії розподілу швидкості прибуття, отриманому на основі вимірювань списку зомбі.

Використовуючи вимірювання швидкості надходження, можна розрахувати бажану середню затримку черги та підштовхнути до неї робочу точку рівноваги системи, адаптувавши параметри RED th_{min} і th_{max}. Політика контролю така, що затримка в черзі помітно зменшується без шкоди для коефіцієнта скидання пакетів і пропускної здатності.

Використовуючи ns2, застосувано алгоритм до типового сценарію зв’язку з одним вузьким місцем із пропускною здатністю каналу 2 Мбіт/с. Трафік включає N постійних джерел FTP, половина джерел надсилає 576-байтні пакети, а половина джерел — 1500-байтні пакети. На рисунку 5.1 проілюстовано, як алгоритм порівнюється з Adaptive RED за період, протягом якого N змінюється від 15 до 50 джерел FTP за час t=80 с. Пунктирні лінії показують поведінку алгоритму Adaptive RED, який змінює P_{max}, щоб повернути середній розмір черги до цільового значення на півдорозі між фіксованими значеннями th_{min} та th_{max}. Суцільні лінії показують поведінку нового алгоритму; ступінчасті лінії показують th_{min} і th_{max}, що

змінюються з інтервалом 0,5 секунди. Слід зазначити, що після t=80 с відбувається початкове збільшення розміру черги, що є типовим для TCP і RED, пропускна спроможність кожного потоку повинна зменшуватися, тому втрати та затримка збільшуються.

Однак обидва алгоритми адаптуються, збільшуючи втрати пакетів, але зменшуючи середню затримку. Запропонований алгоритм здатний адаптуватися до рівноваги меншої затримки в черзі, коли виміряна дисперсія швидкості всіх джерел менша, що зазвичай відповідає більш перевантаженій мережі та менш інтенсивному трафіку, затримку можна зменшити. На рисунку 5.2 наведено порівняння продуктивності та затримки для обох алгоритмів для N = 4, 8, 12, 16, 20, 30 і 40, і показано, що помітно зменшена затримка відбувається за рахунок обмеженого падіння продуктивності.

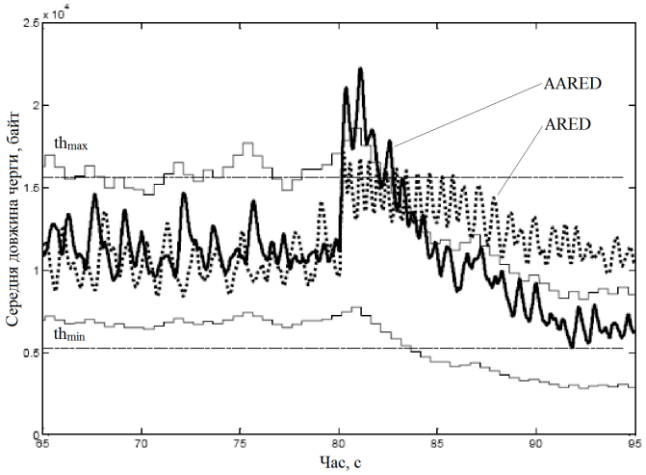


Рисунок 5.1 – Порівняння адаптивних процесів AARED і Adaptive RED

На рисунках 5.1 та 5.2 показано наступне. По-перше, механізм вимірювання розширеного списку зомбі здатний працювати в режимі реального часу, а результати вимірювання можуть відрізняти зміни характеристик трафіку, такі як зміни навантаження трафіку від природних коливань швидкості передачі даних, а отже, і черги. По-друге, адаптивний механізм працює так само, як і в алгоритмі Adaptive RED; але він не тільки

контролює середній розмір черги, щоб коливатись навколо середини між мінімальним порогом th_{min} і максимальним порогом th_{max}, але також контролює значення th_{min}. По-третє, контроль th_{min} веде до успішного контролю робочої точки в термінах (p, avgQ), метою якого є зменшення затримки при визначенні належної можливості роміщення пакетів в черзі.

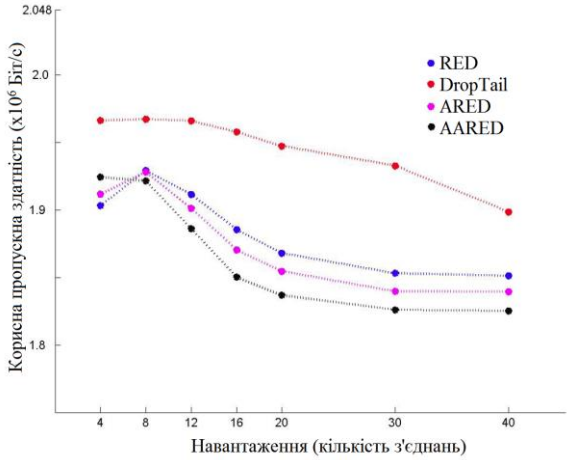


Рисунок 5.2 – Порівняння корисної пропускної здатності (Goodput) різних алгоритмів керування чергою

Детальне дослідження продуктивності нового алгоритму буде зроблено у наступній главі, де він буде всебічно порівнюватися з кількома алгоритмами.

У розділі 5.3 розглядається та порівнюється його продуктивність коефіцієнта падіння пакетів. У розділі 5.4 розглядається та порівнюється його пропускна здатність. У розділі 5.5 розглядається та порівнюється його поведінка goodput. У розділі 5.6 автор намагається пояснити його загальну продуктивність і поведінку під кутом огляду робочих точок.

6 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

Продуктивність AARED порівнюється з іншими популярними алгоритмами управління чергами маршрутизаторів, такими як DropTail, RED і Adaptive RED.

6.1 Опис сценарію

Ефективність алгоритму керування чергою часто досліджується за сценарієм зв’язку з одним вузьким місцем. Це не лише тому, що сценарій простий, а й тому, що він популярний і є остаточним будівельним блоком усіх топологій у реальних програмах.

Використовуючи ns-2, застосовується алгоритм AARED до типового сценарію мережі з топологією «гантелей», в якому є вузьке місце. Обирається канал E1 (2 048 000 біт/с) як вузьке місце.

Наскрізна затримка поширення встановлена на 55 мс, надалі це значення також змінюється на 15 мс, 100 мс і 200 мс для порівняння.

Пакети TCP зі старих реалізацій часто використовують розмір пакета 576 байт, що становить близько 11,5% пакетів і 16,5% інтернет-трафіку. Зараз більшість даних, що передаються в Інтернет, складаються з повнорозмірних кадрів Ethernet (1500 байт), на які припадає близько 10% пакетів, але 37% трафіку. Вирішено застосувати змішаний розмір пакету як 576 байт, так і 1500 байт. У моделюванні використовується парна кількість з’єднань, причому половина з них надсилає пакети розміром 576 байт, а інша половина – пакети розміром 1500 байт.

Крім того, використовується NewReno TCP, оскільки він також здатний обробляти багаторазову втрату пакетів в одному вікні даних і не вимагає змін як для відправників, так і для одержувачів. Застосовуються постійні FTP-з'єднання до вузького місця і змінюється кількість з'єднань, щоб змінити

навантаження каналу. Встановлюється максимальний буферний простір на 50 середніх розмірів пакетів. Симуляції виконуються протягом 150 секунд моделювання і відкинути перші 100 секунд для розгляду стабілізації. Кожний єтап моделювання запускається 20 разів з різними значеннями. Ці початкові числа використовуються для генератора псевдовипадкових чисел у ns-2, і випадкові числа, згенеровані з використанням різних початкових чисел, гарантовано незалежні.

Потім обробляються 20 результатів моделювання і обчислюються їхній 95% довірчий інтервал. Сценарій моделювання показано на рисунку 6.1.

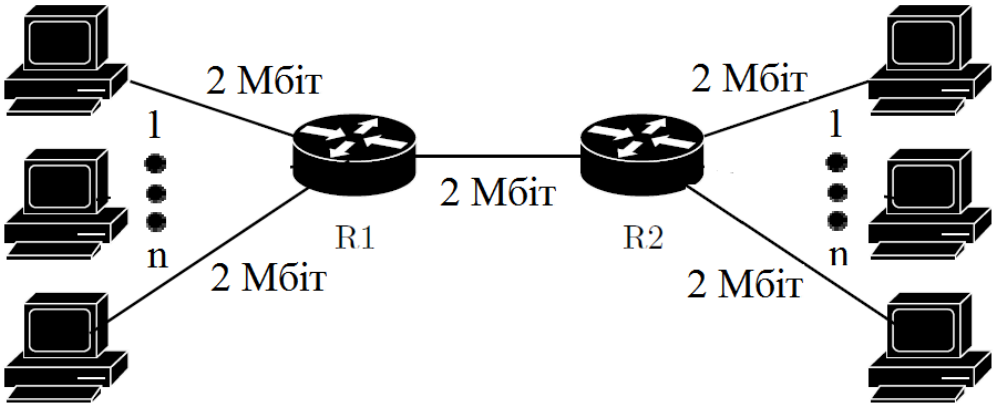


Рисунок 6.1 – Сценарій з одним вузьким місцем

6.2 Поведінка затримки в черзі

Застосовуються постійні FTP-з'єднання до вузького місця і змінюється кількість з'єднань для зміни навантаження. Кількість з'єднань становить N = 4, 8, 12, 16, 20, 30, 40, для того, щоб вивчити, як змінюється довжина черги AARED для випадків легкого та великого навантаження. Для кожного завантаження запускається моделювання 20 разів з 20 різними початковими уставками. Обчислюється 95% довірчий інтервал для 20 симуляцій кожного навантаження. Результати моделювання представлені на рисунку 6.2.

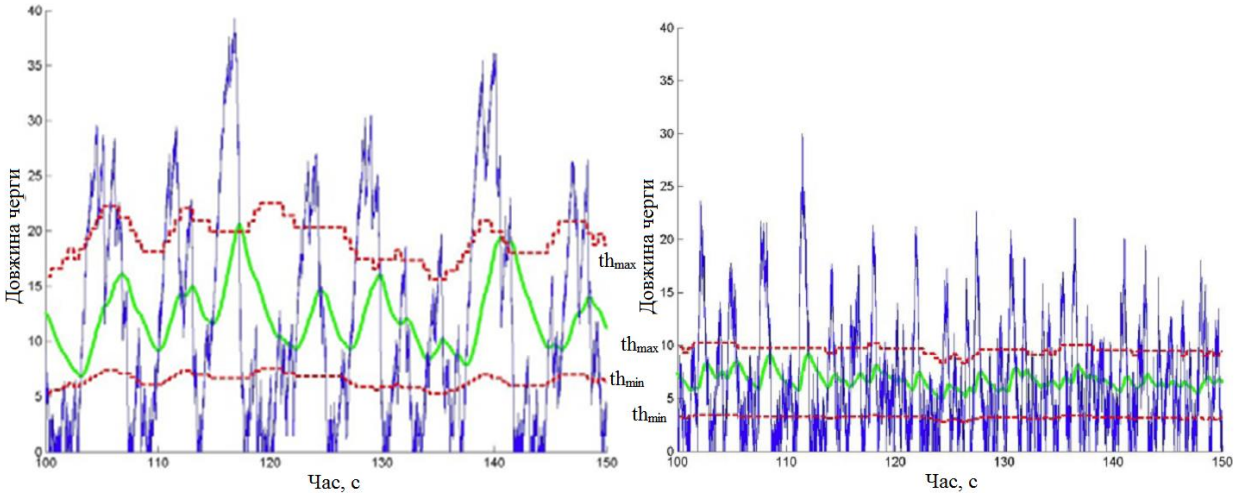


Рисунок 6.2 – Довжина вхідної черги маршрутизатора R1 з алгоритмом AARED при невеликому і великому навантаженні

Можна помітити, що середня довжина черги має менші коливання, а трафік стає менш інтенсивним із збільшенням кількості з'єднань. На рисунку 6.2, коли навантаження дуже невелике (N=4 FTP-з'єднання), миттєвий розмір черги сильно коливається. Це пов'язано з розширеним механізмом списку зомбі в алгоритмі AARED; і параметри RED мінімальний поріг (th_{min}) і максимальний поріг (th_{max}) адаптивно встановлюються з більшим значенням для буфера черги, щоб забезпечити високу пропускну здатність і високу продуктивність.

Коли навантаження дуже велике (N=40 FTP-з'єднань), трафік виглядає менш інтенсивним, а th_{min} і th_{max} адаптивно встановлюються зі значно нижчим значенням, щоб зменшити затримку. Результати моделювання показують, що затримка помітно зменшується, тоді як пропускна здатність і коефіцієнт скидання пакетів залишаються майже незмінними.

Оскільки кількість постійних FTP-з'єднань зростає, вони конкурують за свою частку пропускної здатності; і результатом є зниження пропускної здатності для кожного потоку. Таким чином, дисперсія швидкості прибуття всіх джерел стає меншою, що сприймається механізмом вимірювання розширеного списку зомбі. Алгоритм AARED використовує це вимірювання

і активно адаптує th_{min} і th_{max} до меншого значення, викликаючи відповідно зменшення середнього розміру черги та затримки черги. Ефект зменшення затримки в черзі щодо збільшення кількості з’єднань показано на наступному рисунку.

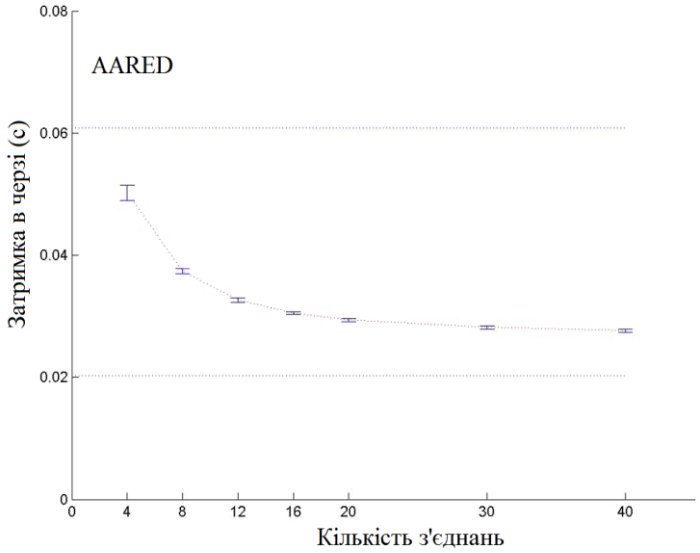


Рисунок 6.3 – Середня затримка в черзі алгоритму AARED

Зміни навантаження (показано з 95% довірчим інтервалом, одиниця осі у є другою, пунктирні лінії приблизно через 0,02 секунди та 0,06 секунди відповідають типовим налаштуванням параметрів RED th_{min} =5 та th_{max} =15 відповідно; одиниця x -вісь - кількість з'єднань, що відповідає зростанню навантаження зліва направо).

На рисунку 6.3, зліва направо, кількість постійних TCP-з'єднань збільшується, отже, пропускна здатність потоку падає, а навантаження зростає. Зі збільшенням навантаження дисперсія швидкості приходу пакетів усіх джерел зменшується, а сукупний трафік стає менш інтенсивним. Коли це відбувається, підтримка меншої завантаженості черги значно зменшує затримку, не вносячи жодних змін у пропускну здатність і коефіцієнт скидання пакетів. Отже, алгоритм зменшує затримку, коли навантаження зростає, а виміряна дисперсія швидкості стає меншою.

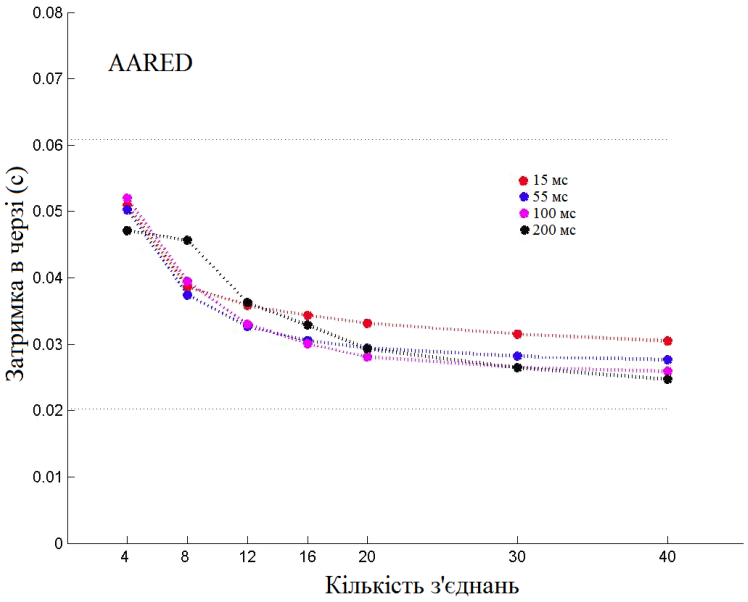


Рисунок 6.4 – Середня затримка в черзі AARED в залежності від різних затримок передачі в каналі

На рисунку 6.4 показано результати моделювання з серією затримок поширення в каналі, які становлять 15 мс, 55 мс, 100 мс і 200 мс. Всі інші налаштування параметрів залишаються незмінними. Запускається моделювання 20 разів для кожної затримки поширення та кожного навантаження. Видно, що алгоритм AARED працює послідовно з різними затримками поширення та навантаженням на трафік. Середній розмір черги і, отже, затримка в черзі постійно падають, коли навантаження збільшується.

Порівняємо AARED з іншими алгоритмами керування чергою, такими як алгоритми DropTail, RED та Adaptive RED з затримкою поширення 55 мс в каналі.

На рисунку 6.5, зліва направо, кількість постійних TCP-з'єднань збільшується, отже, пропускна здатність потоку падає, а навантаження збільшується. Отже, рисунку 6.5 порівнюється затримки черги для різних алгоритмів. Можна побачити, що алгоритм DropTail має найвищу затримку в черзі, оскільки він завжди працює при повній зайнятості черги, без винятку навіть при невеликому навантаженні. Пояснення цьому полягає в природі

трафіку TCP. Відправник TCP завжди намагається захопити більше пропускної здатності для себе, що постійно створює перевантаження. Варто звернути увагу, що для алгоритму DropTail при невеликій кількості з'єднань спостерігається невелике зниження затримки в черзі.

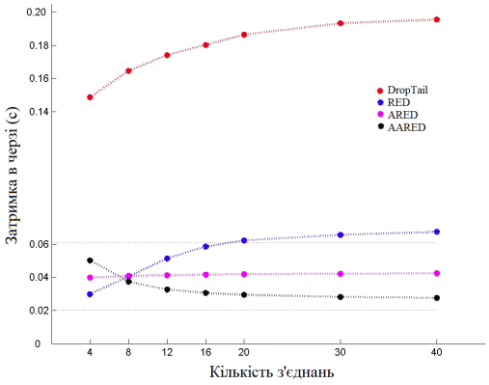


Рисунок 6.5 – Затримки в черзі алгоритмів DropTail, RED, Adaptive RED і AARED

6.3 Порівняння коефіцієнта скидання пакетів

За тією ж топологією ми запускаємо той самий набір симуляцій для алгоритмів Droptail, RED і Adaptive RED і відстежуємо їхні коефіцієнти скидання пакетів. Використовуються широко відомі налаштування параметрів для алгоритмів RED і Adaptive RED. Для алгоритму RED встановлюється P_{max}=0,1, th_{min}=5 і th_{max}=15. Для Adaptive RED використовуєтьсямо th_{min}=5 і th_{max}=15.

На рисунку 6.6, зліва направо, кількість постійних TCP-з'єднань збільшується, отже пропускна здатність потоку падає, а навантаження зростає. На рисунку видно, що коефіцієнт скидання пакетів збільшується зі збільшенням навантаження. Причина, чому всі варіанти RED, включаючи алгоритм AARED, мають значно вищий коефіцієнт падіння пакетів, ніж у DropTail полягає у тому, що під час перевантажень DropTail завжди працює навколо повної зайнятості черги і, отже, має набагато більшу затримку в

черзі, ніж у варіантів RED. DropTail має найнижчий коефіцієнт падіння пакетів. Коефіцієнт скидання пакетів має малопомітні відмінності між алгоритмами варіантів RED, включаючи алгоритм AARED. З іншого боку, відмінності в затримці в черзі можуть бути відносно очевидними, як показано на рисунку 6.6.

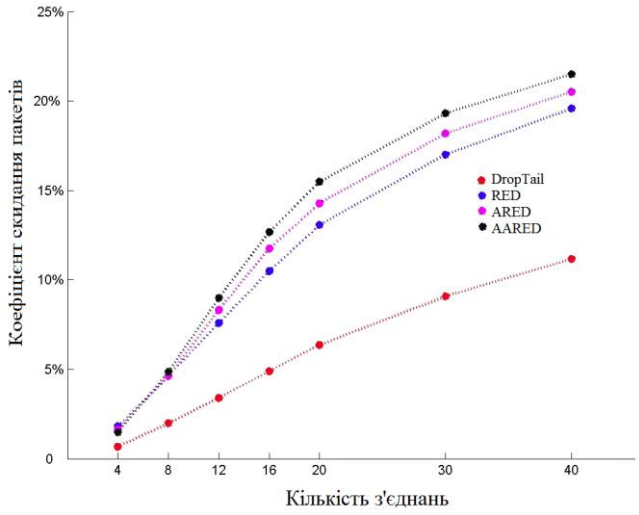


Рисунок 6.6 – Показники коефіцієнта скидання пакетів алгоритмів DropTail, RED, Adaptive RED і AARED

6.4 Порівняння пропускної здатності

Природа трафіку TCP полягає в тому, щоб досліджувати доступну пропускну здатність і захоплювати якомога більше пропускної здатності для себе, отже, TCP завжди підштовхує вузьке місце до перевантажень. Однак трафік TCP також реагуватиме на значні перевантаження, експоненціально стримуючи своє вікно перевантаження, що загалом призводить до експоненційного зниження швидкості передачі даних. З цієї причини Інтернет-трафік інтенсивний, а надзвичайні коливання швидкості передачі даних можуть призвести до зниження загальної пропускної здатності, оскільки втрачений пакет може призвести до періодичного зниження використання каналу. Крім того, якщо алгоритмам активного керування

чергою надано неправильні параметри, черга може по періодично переповнюватися, що погіршує ситуацію та спричиняє падіння пропускної здатності. Глобальна синхронізація є одним із екстремальних прикладів такого роду.

Порівняємо його продуктивність AARED з іншими алгоритмами управління чергами. Результати моделювання представлені на рисунку 6.7.

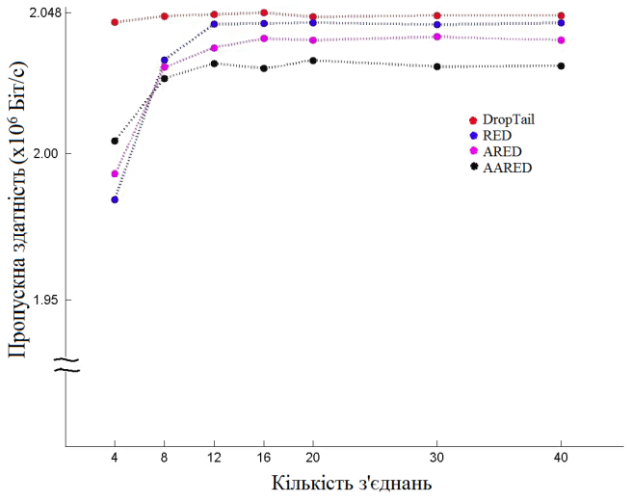


Рисунок 6.7 – Пропускна здатність алгоритмів DropTail, RED, Adaptive RED і AARED

На рисунку 6.7, зліва направо, кількість постійних TCP-з'єднань збільшується, отже, пропускна здатність кожного потоку зменшується, а навантаження збільшується. Можна побачити, що алгоритм DropTail має найвищу пропускну здатність, оскільки він завжди працює в стані повної черги. Однак його пропускна здатність лише трохи краща, ніж у алгоритмів варіанту RED, включаючи алгоритм AARED. Порівняно з рисунком 6.5, його підвищення пропускної здатності в порівнянні з алгоритмами варіанту RED відбувається за ціною значно більшої затримки в черзі. Слід зауважити, що пропускна здатність алгоритму RED помітно падає при невеликому навантаженні. Це пояснюється тим, що його середній розмір черги наближається до мінімального порогу th_{min} при невеликій кількості з’єднань.

А оскільки трафік TCP більш інтенсивний при невеликій кількості з’єднань, як показано на рисунку 6.2, використання каналу та пропускна здатність алгоритму RED падають. З іншого боку, коли навантаження стає все важчим і важчим, середній розмір черги алгоритму RED наближається до максимального порогу th_{max}. Така більша зайнятість черги покращує пропускну здатність за рахунок більшої затримки в черзі. Однак схема відкидання пакетів алгоритму RED за своєю природою передбачає, що для більшого навантаження трафіку і, отже, більш серйозних заторів, ймовірність скидання пакетів має зрости. Крім того, для RED може бути надзвичайно важко вибрати набір параметрів, які можуть досягти високої продуктивності в усіх аспектах і за будь-яких обставин, що змушує персонал з обслуговування мережі скаржитися на труднощі налаштування параметрів RED.

AARED підтримує статистично стабільну зайнятість черги на середині між th_{min} і th_{max}, та його поведінка покращується порівняно з алгоритмом RED, особливо при невеликому навантаженні. На відміну від алгоритму RED, схема відкидання пакетів алгоритму AARED за своєю природою передбачає, що для більшого навантаження трафіку і, отже, більш серйозних заторів, ймовірність скидання пакетів має зростати, однак завантаженість черги не має нічого спільного із заторами і має залишатися стабільною.

Однак, незважаючи на те, що відмінності дуже невеликі, різні політики керування чергою мають різницю в їх продуктивності. Однак відмінності в затримці черги можуть бути відносно очевидними, як було показано раніше на рисунку 6.5, вважаючи, що в наш час все більше додатків і сервісів наголошують на зниженні затримки передачи.

ВИСНОВКИ

Комунікаційні мережі наступного покоління будуть базуватися на парадигмі IP. Хоча Інтернет зростає з експоненційною швидкістю, його пропускна здатність завжди відстає від зростання трафіку. Це означає, що з обмеженими ресурсами пропускної здатності слід знайти підходи, щоб підкреслити та покращити продуктивність мереж, щоб задовольнити потреби критичних у часі та важливих додатків.

Для алгоритму керування чергами маршрутизаторів, орієнтованого на затримку, ключовим компонентом є масштабований механізм вимірювання параметрів трафіку в реальному часі. Алгоритм, запропонований в кваліфікаційній роботі, поєднує цей механізм вимірювання в реальному часі зі звичайною стратегією алгоритму RED та адаптивний підхід керування параметрами алгоритму Adaptive RED, завдяки чому відстежує сплески трафіку та відрізняє його від природних коливань швидкості передачі даних.

Моделювання показує, що запропонований алгоритм функціонує ефективно та успадковує переваги алгоритму RED та алгоритму Adaptive RED в автоматичному налаштуванні параметрів на основі вимірювань параметрів трафіку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. M. Christiansen, K. Je_ay, D. Ott, and F. Smith. Tuning red for web trafic. IEEE/ACM Transactions Networking, 9(3):249-264, Jun 2001.

2. Y. Zhang and L. Qiu. Understanding the end-to-end perfor-mance impact of red in a heterogeneous environment. Technical report, Cornell CS Technical Report 2000-1802, July 2000.

3. L. Guo and I. Matta. The war between mice and elephants. In In Proceedings of the 9th IEEE International Conference on Network Protocols (ICNP), USA, November 2001.

4. V. Jacobson. Congestion avoidance and control. Proc. ACM SIGCOMM 88, 18(4):314-329, August 1988.

5. S. Ryu, C. Rump, and Q. Chunming. Advances in internet congestion control. Communications Surveys & Tutorials, IEEE, 5(1):28-39, 2003.

6. T. Sheldon and B. Sur. Congestion control mechanisms, 2001.

7. W. Stevens. Tcp slow-start, congestion avoidance, fast retransmit and fast recovery algorithms. Technical report, IETF RFC2001, 1997.

8. V. Dumas, F. Guillemin, and P. Robert. A markovian analysis of additive-increase multiplicative-decrease algorithms. JSTOR, 34(1):85-111, March 2002.

9. L. S. Brakmo and L. L. Peterson. Tcp vegas: End-to-end con-gestion avoidance on a global internet. IEEE JSAC, 13(8):1465-1480, 1995.

10. B. Braden, D. Clark, J. Crowcroft, and et al. Recommendations on queue management and congestion avoidance in the internet. Technical report, IETF RFC2309, April 1998.

11. S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. Netw., 1:397-413, 1993.

12. N. Dukkipati, T. Refice, Y. Cheng, J. Chu, T. Herbert, A. Agarwal, A. Jain, and N. Sutin. “An Argument for Increasing TCP’s Initial Congestion

Window”. In: ACM SIGCOMM Computer Communications Review 40.3 (July 2010), pp. 26-33.

13. M. Allman. “Comments on Bufferbloat”. In: ACM SIGCOMM Computer Communications Review 43.1 (January 2012), pp. 30-37.

14. J. Hoe. Improving the start-up behavior of a congestion control scheme for TCP. Technical report, Proceedings of the ACM SIGCOMM, 1996.

15. K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ip. Technical report, IETF RFC2481, 1990.

16. M. Hassan and R. Jain. High Performance TCP/IP Networking: Concepts, Issues, and Solutions. Prentice-Hall, 2003.

17. R. Jain and K. K. Ramakrishnan. Congestion avoidance in computer networks with a connectionless network layer: Concepts, goals and methodology. In in Proc. IEEE Comp. Networking Symp, Washington D.C., pages 134-143, 1988.

18. T.J. Ott, T.V. Lakshman, and L.H. Wong, “SRED: Stabilized RED”, INFOCOM 1999.