Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи боротьби з переповненням буферів маршрутизаторів

(тема)

Виконав:

студент II курсу, групи СПм-23-1

Пилипенко А.О.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

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

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Пилипенку Арсенію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи боротьби з переповненням буферів маршрутизаторів

затверджена наказом по університету від “ 22 ” листопада 2024 р. № 1236 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20 січня 2025 р.

3. Вхідні дані до роботи

1) Моделі та методи для керування мережевими інформаційними потоками

2) Сучасні вимоги до мережних показників

3) Стек протоколів TCP/IP

4) Алгоритми AQM

5) Перелік використаних програмних та апаратних засобів: OpNet 14, NS-3

4. Перелік питань, що потрібно опрацювати у роботі

1) Аналіз сучасного стану проблеми

2) Огляд технологій управління перевантаженням та чергами маршрутизаторів

3) Моделі управління мережним трафіком

4) Вибір програмних та апаратних засобів моделювання

5) Проведення експериментальних досліджень

6) Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 18 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз стану проблеми та сучасних методів її	26.11.24–28.11.24	
	вирішення		
2	Огляд технологій управління інформаційними	29.11.24–04.12.24	
	потоками		
3	Вибір та обґрунтування методики дослідження	05.12.24 –10.12.24	
4	Розробка моделей та алгоритмів управління	11.12.24 –18.12.24	
5	Вибір інструментальних засобів	19.12.24 –23.12.24	
6	Проведення експериментів	24.12.24–26.12.24	
6	Оформлення пояснювальної записки	27.12.24–-03.01.25	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 80 с., 28 рис., 4 табл., 1 дод., 27 джерел.

AQM, ACK, RTT, TCP, АЛГОРИТМ, КАНАЛ, МАРШРУТИЗАТОР, МЕРЕЖА, МЕРЕЖЕВИЙ РІВЕНЬ, МОДЕЛЮВАННЯ, ПОРІГ СКИДАННЯ, ТАЙМ-АУТ, ТРАНСПОРТИЙ РІВЕНЬ, УПРАВЛІННЯ ЧЕРГОЮ.

Метою кваліфікаційної роботи є розробка моделей, алгоритмів та методів управління інформаційними потоками комп’ютерних мереж.

У ході виконання кваліфікаційної роботи проведено огляд теоретичних та практичних аспектів функціонування протоколів транспортного та мережевого рівнів моделі OSI, зроблено аналіз сучасних інструментальних засобів оцінювання ефективності існуючих мережевих протоколів, запропоновано метод управління чергами маршрутизаторів та проведено моделювання роботи алгоритмів AQM при різних мережевих параметрах.

ABSTRACT

Master’s thesis: 80 pages, 28 figures, 4 tables, 1 appendices, 27 sources.

AQM, ACK, ALGORITHM, CHANNEL, NETWORK, NETWORK LAYER, QUEUE CONTROL, SIMULATION, RESET THRESHOLD, RTT, ROUTER, TCP, TIMEOUT, TRANSPORT LAYER.

The purpose of the qualification work is to develop models, algorithms and methods of managing information flows of computer networks.

In the course of the qualification work, an overview of the theoretical and practical aspects of the functioning of protocols of the transport and network levels of the OSI model was carried out, an analysis of modern tools for evaluating the effectiveness of existing network protocols was made, a method of managing router queues was proposed, and the operation of AQM algorithms was simulated under various network parameters.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

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

AQM – алгоритм активної черги (англ., Active Queue Mechanism)

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

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

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

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

FRED – потоковий RED (англ., Flow RED)

GRED – пологий RED (англ., Gentle RED)

QD – затримка в черзі (англ., Queuing Delay)

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

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

RTT – випадкове раннє виявлення (англ., Round Trip Time)

SRED – стабілізований RED (англ., Stabilized RED)

ВСТУП

Протокол керування передачею (TCP) часто використовується для зв’язку через Інтернет, включаючи зв’язок між датчиками та виконавчими механізмами, через його високу надійність. Оскільки загальний обсяг TCP-потоків зростає, їхні комунікаційні пакети можуть спричинити серйозні затори в маршрутизаторах.

Коли маршрутизатор перевантажений, тобто буфер маршрутизатора заповнений пакетами і він не може більше отримувати пакети, усі пакети, що надійдуть після цього, будуть відкидані, доки в буфері не з’явиться вільне місце. Під час зв’язку через TCP на основі втрат хост-одержувач виявляє втрату пакетів як ознаку перевантаження мережі та повідомляє хост-відправника про перевантаження.

Отримавши це сповіщення, хост-відправник зменшує розмір вікна надсилання, щоб контролювати перевантаження, що призводить до зниження швидкості зв’язку.

Завдяки цьому механізму велика кількість пакетів відкидається, якщо виникає серйозне перевантаження мережі, що призводить до ситуації, коли багато потоків TCP, які спільно використовують один канал мережі, одночасно зменшують розміри своїх вікон, таким чином спільне використання мережі призводить до низької пропускної здатності. Це явище називається глобальною синхронізацією, і вона значно знижує ефективність зв’язку, іноді призупиняючи послуги, що надаються через мережу.

Для вирішення цієї проблеми було запропоновано метод контролю перевантаження, який називається активним керуванням чергою (AQM) [1]. AQM відкидає пакети в маршрутизаторі з вузьким місцем до того, як його буфер заповниться, щоб уникнути серйозного перевантаження потоків TCP. Відкидаючи пакети до того, як станеться переповнення буфера, AQM уникає великої затримки в черзі та глобальної синхронізації.

Одним із найвідоміших методів AQM є випадкове раннє виявлення (RED) [2]. RED випадково відкидає пакети з імовірністю відкидання, визначеною відповідно до середньої довжини черги. Він може підтримувати довжину черги відносно стабільною, запобігаючи глобальній синхронізації. Завдяки простій процедурі розрахунку було досліджено та проаналізовано різноманітні алгоритми RED [3].

RED має численні параметри, і його параметризація для отримання задовільної продуктивності за різних обставин дуже складна [4]. Ці параметри потрібно підбирати дуже ретельно; інакше RED працює погано, що призводить до зниження пропускної здатності та збільшення рівня втрати пакетів [5].

Контроль перевантаження на основі AQM негативно впливає на помилку моделювання системи та часову затримку, викликану часом проходження (RTT) потоків TCP. Як згадувалося раніше, хост-одержувач сповіщає відповідний хост-відправник про перевантаження, коли він виявляє втрату пакета, і хост-відправник зменшує розмір вікна надсилання на основі цього. Це означає, що після надсилання пакету хост-відправник не може зменшити розмір вікна надсилання, доки не отримає сповіщення з певною затримкою RTT.

Таким чином, мережі TCP/AQM містять елемент часової затримки, такий як RTT між відправниками та одержувачами, що впливає на продуктивність системи керування [6].

1 МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

Інтернет в останні роки використовується все більш інтенсивніше. Незважаючи на те, що мережева інфраструктура регулярно оновлювалася, а здатність керувати інтенсивним трафіком значно зросла, особливо в великих мережах, перевантаження ніколи не перестають з’являтися, оскільки обсяг трафіку, що циркулює в Інтернеті зростає ще швидше з кожним роком.

Таким чином, механізми контролю перевантаження відіграють життєво важливу роль у функціонуванні комп’ютерних мереж. Активне керування чергами (AQM) – це популярний тип механізму контролю перевантажень, який реалізується на шлюзах (зокрема, маршрутизаторах), який може передбачити та уникнути перевантаження до того, як воно станеться. При правильній конфігурації AQM може ефективно зменшити затори та пом’якшити деякі проблеми, такі як глобальна синхронізація та несправедливість до швидкісного трафіку.

Проте існує ще багато проблем щодо AQM. Більшість схем AQM досить чутливі до налаштувань своїх параметрів, і ці параметри можуть значною мірою залежати від профілю мережевого трафіку, і який, ймовірно, змінюватиметься з часом. При поганій конфігурації багато AQM працюють не краще, ніж базовий Tail Drop. На даний момент не існує ефективного методу порівняння продуктивності цих алгоритмів AQM, спричиненого проблемою конфігурації параметрів.

Мета кваліфікаційної роботи полягає в тому, щоб запропонувати новий метод боротьби з перевантаженнями приймальних буферів маршрутизаторів із використанням нечіткої логіки, що в кінцевому випадку повинно привести до зменшення ймовірності виникнення заторів на маршрутизаторах.

В рамках магістерської кваліфікаційної роботи необхідно:

- провести всебічний аналіз методів запобігання перевантаженням в каналах;комп’ютерних мереж;

- проаналізувати різноманітні чінники, впливаючі на виникнення перевантаження;
- провести огляд існуючих моделей та методів управління мережевим трафіком;
- запропонувати метод управління чергами маршрутизаторів відповідно до поточного мережевого стану;
- провести імітаційне моделювання запропонованих теоретичних викладок;
- провести короткий аналіз отриманих результатів.

2 БОРОТЬБА С ПЕРЕВАНТАЖЕННЯМ В СТЕЦІ ПРОТОКОЛІВ TCP/IP

2.1 Історія TCP/IP

Стек протоколів TCP/IP складається з низки протоколів, які разом визначають спосіб обміну даними між комп'ютерами. Нащадок мережі передових дослідницьких проектів (ARPANET), створеної та фінансованої Міністерством оборони США в 1969 році, став стандартом де-факто, який сьогодні використовується в Інтернеті.

TCP/IP розгортає концепцію багаторівневості, у якій кожен рівень має окреме завдання для виконання. Ідея наявності окремих рівнів походить зі світу комп'ютерного програмування, де програми поділяються на менші функції/методи та структури/класи, які спілкуються один з одним через виклики (передачу повідомлень). Мета полягає в тому, щоб дозволити розробникам програмного забезпечення зосередитися на певному рівні і не турбуватися про інші рівні.

Прикладним програмістам, наприклад, не потрібно знати про базову інфраструктуру, і їм надається бібліотека/інтерфейс, необхідний для передачі повідомлення на рівень нижче. Такою бібліотекою є Berkley Socket API, який є набором викликів функцій C, які підтримують мережевий зв'язок. Розробники апаратного забезпечення також отримують вигоду від такого поділу рівнів, оскільки вони можуть швидко та дешево виготовити продукт, спеціально адаптований для певного рівня. Маршрутизатори та комутатори – це такі пристрої, які в основному працюють з мережевим і канальним рівнями відповідно.

Ще однією перевагою концепції багаторівневості є здатність справлятися з технологічними змінами. Новий і більш продуктивний протокол можна легко адаптувати на будь-якому з рівнів без значного впливу на інші рівні.

У 1980-х роках Міжнародна організація стандартів (ISO) почала розробляти модель мережевої системи, яка називається моделлю взаємодії відкритих систем (OSI). Ця модель мала 7 рівнів і в основному використовується як еталон у світі мереж. Ця модель ніколи не приваблювала широку аудиторію, натомість TCP/IP став широко прийнятою та розгорнутою моделлю. Різні рівні TCP/IP взаємодіють один з одним. Це можна побачити на рисунку 2.1.

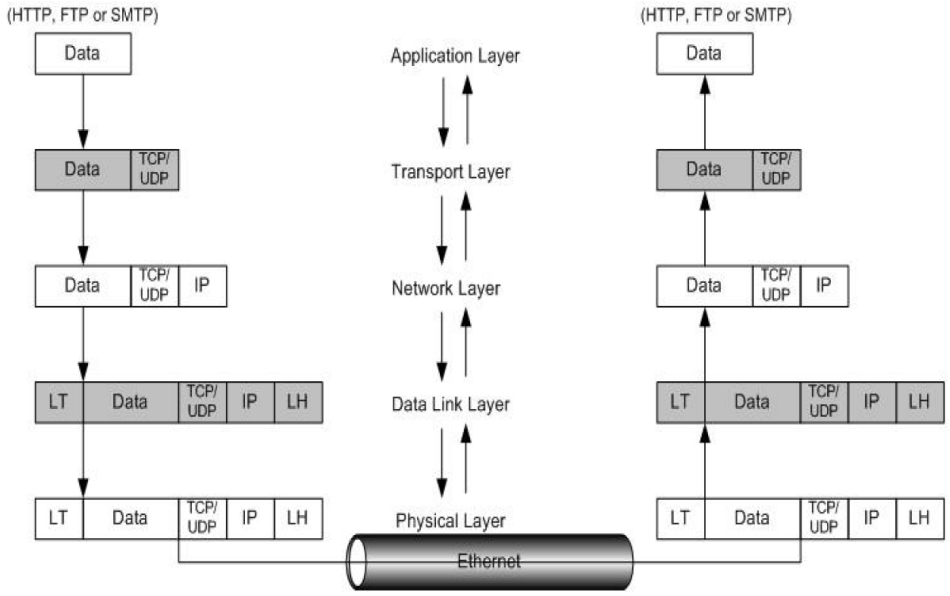

Рисунок 2.1 – Структура TCP/IP

На прикладному рівні дані починають свій шлях. На цьому рівні працюють такі популярні програми, як Інтернет (HTTP), пошта (SMTP) і протокол передачі файлів (FTP).

Після того як програма генерує дані, вона передає їх на нижчий рівень, у цьому випадку на транспортний рівень. Програмісти прикладних програм заздалегідь визначають тип транспортного протоколу, який підходить для конкретної програми.

Залежно від того, потрібен надійний транспортний протокол чи ні, програміст вибирає TCP або UDP як засіб транспортування пакету.

Коли програма передає пакет транспортному протоколу, вона витягує з нього необхідну інформацію. Найважливішим є номер порту, який ідентифікує програму-одержувач. Ця інформація заповнюється в заголовку пакету, і дані інкапсулюються в цьому заголовку. Заголовок – це не що інше, як маркер, який вказує на перехід між бітами, що належать частині даних, і тими, що належать TCP/UDP. Заголовок в купі з даними називаються сегментом на транспортному рівні, і цей сегмент передається на мережевий рівень.

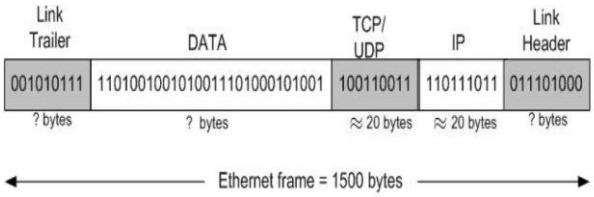

Рисунок 2.2 – Кадр

Мережевий рівень (рівень 3) відповідає за переміщення даних між мережею. Уся необхідна для цього інформація міститься в його заголовку. IP – це протокол, який працює на цьому рівні. Протокол інкапсулює сегмент з його заголовком і передає дейтаграму (сегмент + IP-заголовок) на канальний рівень.

Безпосередньо перед тим, як дані вводяться в середовище передачі, канальний рівень отримує дейтаграму від IP і доповнює її заголовком і трейлером. Завдання рівня полягає в переміщенні даних через канал зв’язку. Зазвичай канал – це пряме з’єднання між двома пристроями, які працюють на рівні 2 або 3. Протоколи, які працюють на цьому рівні, включають, зокрема, протокол «точка-точка» (PPP) і протокол інтерфейсу послідовної лінії (SLIP). Ethernet/IEEE 802.3, Token Ring/IEEE 802.5 і Fiber Distributed Data Interface (FDDI) також є протоколами Data Link. Після розміщення заголовка та трейлера кадр передається до фізичного рівня.

Фізичний рівень – фактичний фізичний носій, який використовується хостом. Це може бути провідне середовище, таке як Ethernet, Frame-relay або Fiber Optics. Це може бути безпровідний зв’язок, наприклад IEEE 802.11, IEEE 802.16 або Bluetooth. Плата мережевого інтерфейсу (NIC), встановлена на комп’ютері визначає тип носія, з яким вона може обмінюватися даними.

2.2 Протокол керування передачею (TCP)

Набір протоколів TCP/IP включає два транспортних протоколи, а саме TCP і UDP. Такі програми, як веб-перегляд (HTTP), електронна пошта (SMTP) і передача файлів (FTP), використовують TCP як основний транспортний протокол.

TCP забезпечує надійну транспортну службу, орієнтовану на з’єднання, яка гарантує доставку даних у порядку перед тим, як акуратно розірвати з’єднання. Він також має спосіб контролю обсягу трафіку, що вводиться в мережу, за допомогою процесу, відомого як контроль потоку та контроль перевантаження.

2.2.1 Протокол орієнтований на підключення

Перш ніж програма, яка використовує TCP як транспортний протокол, зможе передавати будь-які дані, вона повинна встановити з’єднання з приймальною стороною, пройшовши процес, який називається тристороннім рукостисканням. Відправник спочатку надсилає пакет SYN, чекає на пакет відповіді SYN-ACK від одержувача, а потім відповідає ACK для завершення процесу. Цей процес показано на рисунку 2.3.

Під час тристороннього рукостискання порядкові номери ініціалізуються випадковим чином, після чого кожному сегменту, яким обмінюються однорангові пристрої, призначається унікальний порядковий номер, який також вказує кількість надісланих байтів.

Після завершення тристороннього рукостискання з'єднання встановлюється і передача даних може початися. Коли всі дані надіслано, відправник і одержувач обмінюються FIN і ACK в обох напрямках, щоб завершити з’єднання.

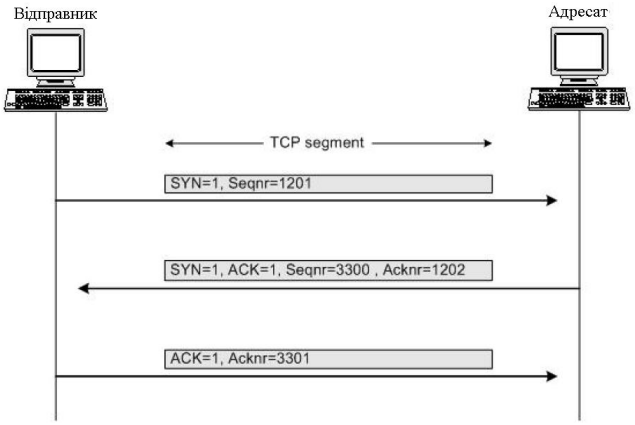

Рисунок 2.3 – Встановлення з’єднання

2.2.2 Надійна доставка

TCP гарантує доставку даних у порядку та без дублювання. Це досягається шляхом використання позитивного підтвердження та повторних передач. TCP-відправник призначає порядковий номер кожному надісланому байту та очікує підтвердження від одержувача.

У заголовку TCP є поле порядкового номера та підтвердження, яке дозволяє відправнику та одержувачу обмінюватися цією інформацією (рисунок 2.4). Він також пов’язує таймер із надісланими даними, і якщо підтвердження не отримано протягом заданого періоду часу, він повторно передає дані.

TCP має ще один механізм, який може ініціювати повторну передачу, а саме повторювані підтвердження (DACK). Коли байти надходять до адресата не за порядком, він генерує DACK, щоб повідомити відправника про ситуацію, запитуючи наступні очікувані байти за порядком.

Рисунок 2.4 – TCP заголовок

2.2.3 Контроль потоку

Контроль потоку гарантує, що відправник не перевантажить одержувача занадто великим трафіком.

Буфери виділяються під час фази тристороннього рукостискання як у відправника, так і адресата для прийняття вхідних пакетів. Щоб уникнути переповнення буфера, яке може призвести до скидання пакетів, між відправником і одержувачем обмінюються повідомленнями із зазначенням розміру буфера.

Відправник явно інформується про доступний розмір буфера в одержувачі через віконне поле заголовка TCP. Приймаюча сторона з’єднання веде підрахунок останнього байта, прочитаного (lastByteRead) приймальною програмою, тобто HTTP, FTP, SMTP тощо. Вона також веде підрахунок останнього байта, який надійшов у порядку від мережі та поміщається в буфер (lastbyteRcvd).

Буфери для вхідних пакетів (rcvBuffer) налаштовуються наступним чином:

$$lastbyteRcvd\text{-}lastByteRead \leq rcvBuffer.$$

Вікно, яке повідомляється відправнику, обчислюється на основі наступного:

$$window=rcvBuffer-(lastbyteRcvd-lastByteRead).$$

Сторона відправника підключення також має переконатися, що кількість даних, які вона надсилає, відповідає розміру вікна, тому вона підтримує змінну advertised-Wnd, яка обчислюється за формулою:

$$lastByteSent-lastByteAcked \leq advertisedWnd.$$

Крім вікна перевантаження, кількість даних, які відправник може надіслати, обмежена його ефективним вікном (effWnd):

$$effWnd=advertisedWnd*(lastByteSent-lastByteAcked).$$

2.2.4 Контроль перевантаження

Механізм контролю перевантаження в TCP відіграє важливу роль як для програми, яка запитує послугу, так і для Інтернету в цілому. Механізм забезпечує обсяг трафіку, який можна ввести в мережу, таким чином регулюючи загальну продуктивність процесів обміну даними. Перевантаження виникає, коли попит на мережеві ресурси перевищує той, який мережа може надати, що призводить до відкидання деяких пакетів. Для TCP відкинуті пакети означають повторну передачу.

Маршрутизатори на шляху з’єднання не можуть явно повідомити TCP про будь-яку перевантаженість через розділення рівнів. Проміжні маршрутизатори працюють на мережевому рівні і тому не взаємодіють з транспортним рівнем. Це зберігає багаторівневу структуру набору TCP/IP, де кожен рівень має окрему функцію. Таким чином, перевантаження виявляється через втрату пакета. Пакет вважається втраченим, якщо ACK не

отримано протягом певного періоду часу (RTO) або після отримання дублікатів ACK. Алгоритмами, які диктують контроль перевантаження, є відповідно повільний старт (SS), уникнення перевантаження (CA), швидка повторна передача та швидке відновлення [7].

Повільний запуск і уникнення перевантажень TCP не має інформації про пропускну здатність шляхів після встановлення нового з'єднання, тому повинен вдаватися до оцінки доступної пропускної здатності. Це досягається шляхом проходження процесу, відомого як повільний запуск і уникнення заторів.

У повільному старті сторона відправника з'єднання підтримує змінну під назвою «Вікно перевантаження» (CWN), яка ініціалізується на 1 максимальний розмір сегмента (MSS). CWN визначає максимальну кількість неопрацьованих пакетів, які були передані, але ще не підтверджені.

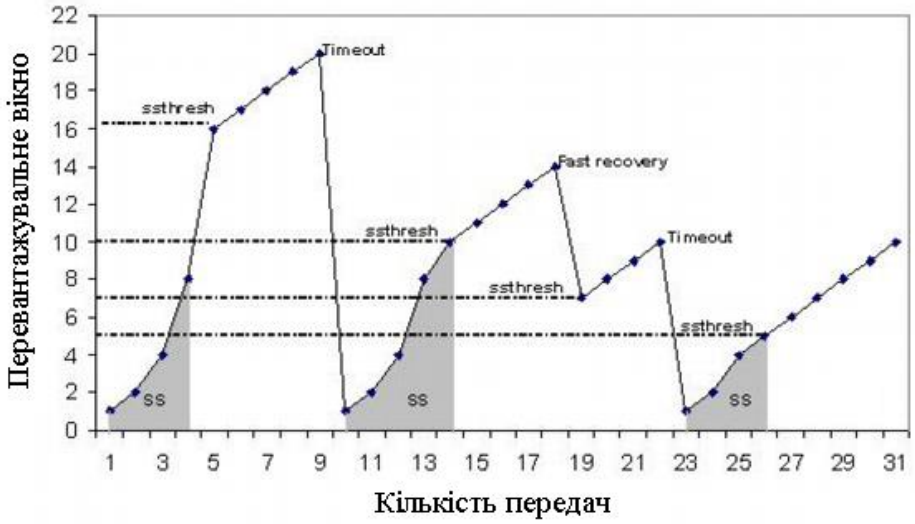

Рисунок 2.5 – Поведінка контролю перевантаження AIMD

Відправник TCP також відстежує оголошений розмір вікна одержувача (RcvWnd) і обчислює його швидкість передачі (sendWnd) наступним чином:

$$\text{SendWnd} = \min\left(\text{CWN}, \text{RcvWnd}\right).$$

Де RcvWnd – це вікно, яке оголошується отримувачем через заголовок TCP. Кожного разу, коли відправник отримує ACK, який підтверджує отримання надісланого пакету, CWN відповідно збільшується (рисунок 2.6). Таким чином, CWN збільшується експоненціально протягом фази SS.

Очевидно, що якщо це збільшення триватиме, вікно перевантаження може перевищити доступну ємність і, таким чином, переповнити буфери проміжних маршрутизаторів.

Щоб подолати цю конкретну проблему, відправник також підтримує іншу змінну, яка називається поріг повільного запуску (ssthresh). Ця змінна визначає перехід від повільного запуску до уникнення заторів. Фаза уникнення перевантаження вводиться, якщо CWN стає більшим за ssthresh. На етапі уникнення перевантажень CWN збільшується лінійно (1 сегмент) для кожного часу проходження в обидві сторони (RTT). Цей процес триває, доки не станеться перевантаження, яке буде виявлено через тайм-аут або повторні підтвердження. Виявлення за допомогою дублікатів підтвердження змусить TCP викликати алгоритми швидкої повторної передачі та швидкого відновлення.

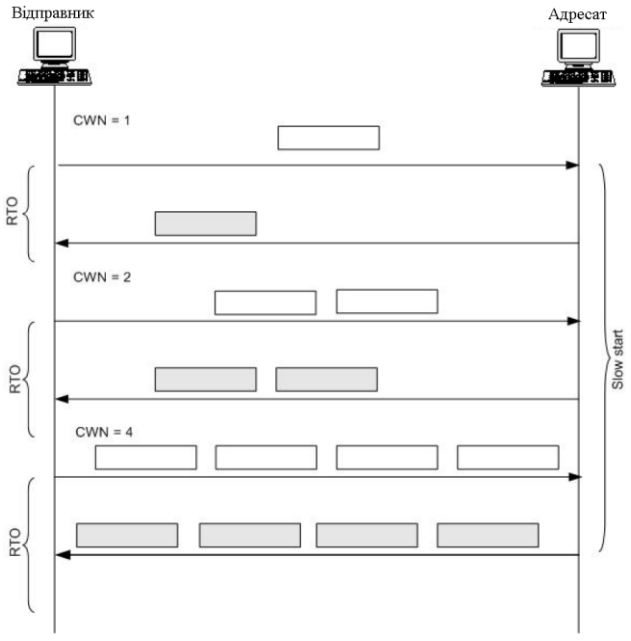

Рисунок 2.6 – Повільний старт

Швидка повторна передача та швидке відновлення Сегменти можуть надходити до приймача TCP не в порядку. Коли це відбувається, приймаючий хост не може доставити ці сегменти до приймаючої програми, і тому повинен буферизувати їх, доки не надійдуть правильні послідовні дані. Згідно з RFC2581, TCP-одержувач повинен надіслати негайний дублікат ACK, щоб повідомити відправника про отриманий непорядковий сегмент і порядковий номер, який очікується.

Це призведе до того, що відправник отримає повторювані підтвердження (DACK) для раніше надісланих даних. RFC2581 вимагає від відправника використовувати алгоритм швидкої повторної передачі після отримання 3 DACK.

Метою алгоритму швидкої повторної передачі є повторна передача втраченого сегмента одразу після 3 DACK, а не чекання часу очікування повторної передачі (RTO), який у деяких ситуаціях може бути відносно довгим.

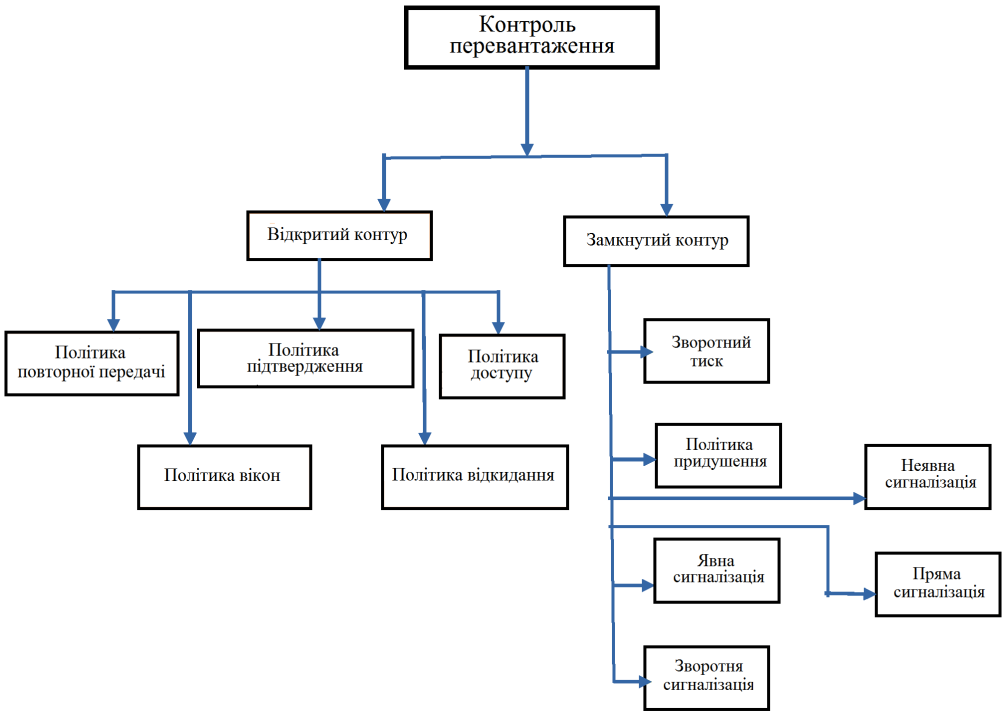

Рисунок 2.7 – Методи контролю перевантажень

Стандарт RFC2581визначає, що відправник повинен використовувати алгоритм швидкого відновлення замість повільного старту, якщо втрачений пакет повторно передається через 3 DACK, а не RTO. Швидке відновлення зменшує вікно перевантаження вдвічі, а не до 1 сегмента, як у повільному запуску. Причина цього полягає в тому, що поява 3 DACK вказує на те, що мережа здатна доставляти деякі сегменти незалежно від перевантаженості, і, отже, не потрібно проводити радикальні вимірювання, такі як зменшення вікна перевантаження до 1 сегмента.

3 КОНТРОЛЬ ПЕРЕВАНТАЖЕННЯ TCP НА ОСНОВІ АКТИВНОГО КЕРУВАННЯ ЧЕРГОЮ

3.1 Контроль перевантаження

Контроль перевантаження (або відновлення) та уникнення перевантаження виконуються на транспортному рівні кінцевих систем (тобто джерел) у контролі перевантаження Інтернету.

Перший є реактивним, оскільки контроль над перевантаженням зазвичай починає діяти після перевантаження мережі, тобто виявлення перевантаження. Останній є проактивним, оскільки запобігання перевантаженням починає діяти до того, як мережа стане перевантаженою, тобто коли очікується перевантаження. Стратегією контролю, яка використовується в поточному наскрізному контролі перевантаження TCP, особливо в TCP Reno та його варіантах, є адитивне збільшення–мультиплікативне зменшення (AIMD).

Зі збільшенням кількості користувачів і розміру Інтернету спостерігалося все більше втрат пакетів та інших погіршень продуктивності. Таким чином, нещодавно було запропоновано багато алгоритмів для ефективного контролю над перевантаженням і для того, щоб впоратися з еволюцією послуг Інтернету.

Ці підходи можна класифікувати на дві категорії: вихідні алгоритми та мережеві алгоритми. Ці підходи відносяться до підходів, які модифікують джерело та/або мережеві алгоритми для забезпечення кращого контролю перевантаження, зберігаючи при цьому парадигму поточного Інтернету та принцип проектування TCP.

Алгоритми джерела використовують лише неявну інформацію про перевантаження (наприклад, затримку) у джерелі без будь-якої допомоги з боку мережі. Вихідні алгоритми можна легко реалізувати за допомогою

поточного наскрізного контролю перевантажень TCP без будь-яких модифікацій мережевих механізмів. З іншого боку, мережеві алгоритми потрібно модифікувати не тільки для надання кращої інформації про перевантаження джерелам, але й для ефективного та швидкого виявлення та/або контролю над перевантаженням.

Мережеві алгоритми, такі як активне керування чергою (AQM), які виконуються мережевими компонентами, такими як маршрутизатори, виявляють перевантаження мережі, втрати пакетів або початкове перевантаження та інформують джерела трафіку (явно чи неявно). У відповідь вихідні алгоритми регулюють швидкість надсилання даних у мережу.

Основні питання проектування полягають у тому, на що слід звертати увагу (мережеві алгоритми) і як реагувати (вихідні алгоритми).

До тих пір, поки Інженерно-технічна група Інтернету (IETF) не запропонувала випадкове раннє виявлення (RED) для розгортання, FIFO (First-In-First-Out), TD – Tail drop (відкидання кінця черги або обрубування хвоста) був єдиним механізмом керування чергою, що використовувався в мережі.

Незважаючи на те, що це просто і легко реалізувати, TD має два добре відомі недоліки, блокування та повну чергу. Щоб подолати ці недоліки TD, RED запроваджує ймовірнісне раннє відкидання пакетів, щоб уникнути повної черги.

RED також використовує експоненціально-зважену ковзну середню (EWMA) довжини черги як індикатор перевантаження не тільки для виявлення заторів, що починаються, але й для згладжування бурхливого вхідного трафіку.

Після RED було запропоновано багато варіантів, таких як адаптивний RED і випадкове раннє маркування (REM) [8]. Ці пропозиції AQM продемонстрували серйозні проблеми з (середньою) довжиною черги як індикатором перевантаження.

Наприклад, RED може виявляти та реагувати на довгострокові моделі трафіку, використовуючи довжину черги EWMA, однак він не може виявити початкові затори, спричинені короткочасними змінами навантаження на канал.

Як наслідок, конфігурація параметрів AQM (особливо RED) була основною проблемою, оскільки RED було вперше запропоновано в 1993 році. Щоб вирішити ці проблеми, необхідно, щоб алгоритм AQM мав більш ефективний індикатор перевантаження та функцію контролю.

Щоб уникнути заторів або завчасно контролювати їх, перш ніж вони стануть проблемою, як індикатор заторів, так і функція контролю алгоритмів AQM повинні бути адаптованими до змін у транспортному середовищі, таких як кількість трафіку, коливання транспортного навантаження та характер трафіку.

Було запропоновано два адаптивні та проактивні алгоритми AQM, пропорційно-інтегрально-похідний (PID)-регулятор [9] та Pro-Active Queue Management (PAQM) [10], щоб виявити та контролювати як зародження, так і поточний стан перевантаження ефективно та завчасно.

Ціль цих алгоритмів полягає в тому, щоб проактивно контролювати перевантаження, узгодити довжину черги з бажаним рівнем і забезпечити плавну та низьку швидкість втрати пакетів для кожного потоку, щоб усунути збільшення трафіку від вибухових джерел. Для досягнення цих цілей при розробці цих алгоритмів використовувався класичний пропорційно-інтегрально-похідний (PID) метод керування зі зворотним зв’язком, щоб не лише мати передбачуване виявлення перевантаження та можливість контролю, але й досягти задовільного контролю продуктивності з точки зору динаміки довжини черги (або еквівалентну затримку), швидкість втрати пакетів або використання каналу.

Контроль перевантажень TCP на основі AQM повинен бути адаптованим до динамічно мінливої ситуації трафіку, щоб виявляти, контролювати та уникати поточних і початкових заторів проактивно.

3.2 Поточний контроль перевантаження Інтернету

У поточному використанні Інтернету домінує трафік TCP, такий як віддалений термінал (наприклад, Telnet), FTP, веб-трафік та електронна пошта (наприклад, SMTP). Хоча ці додатки досить еластичні за своєю природою, оскільки вони можуть терпіти або затримку пакетів, або втрату пакетів досить витончено, перевантаження залишається основною проблемою, яка призводить до низької продуктивності.

Основна функція контролю перевантаження в поточному Інтернеті виконується на транспортному рівні. Тут потрібно оновити лише кінцеві системи (тобто джерела), які хочуть реалізувати різні, складні чи ефективні функції.

У поточних мережах TCP/IP втрата пакета (або сегмента) TCP, що позначається тайм-аутом або потрійним дублюванням підтвердження, використовується як ознака перевантаження мережі. Коли виникає перевантаження, TCP контролює швидкість надсилання, обмежуючи розмір вікна перевантаження. Швидкість надсилання даних TCP визначається швидкістю вхідних підтверджень ACK для попередніх пакетів.

Швидкість надходження ACK, у свою чергу, визначається наявністю або відсутністю перевантажених каналів уздовж шляху між джерелом і його призначенням.

У стабільному стані швидкість надсилання джерелом відповідатиме швидкості надходження ACK. Відповідно TCP автоматично виявляє перевантаження та регулює швидкість передачі.

Маршрутизатор – мережевий компонент, який забезпечує маршрутизацію пакетів між різнорідними мережами. Інтернет-маршрутизатор використовує механізм зберігання та пересилання, у якому пакети зберігаються в буфері вихідного каналу на шляху до місця призначення та пересилаються, коли доступна пропускна здатність вихідного каналу. Для підтримки контролю перевантаження кінцевої точки (алгоритми

джерела) потрібні деякі механізми (алгоритми мережі) на компонентах мережі, тобто маршрутизаторах. Ці механізми маршрутизатора зазвичай класифікуються на два механізми: керування чергою та планування.

Керування чергою відноситься до алгоритмів, які керують довжиною черги в буфері шляхом відкидання пакетів. Планування відноситься до алгоритмів, які визначають порядок пакетів, що надсилаються наступним, і використовуються для розподілу доступної смуги пропускання між потоками. Традиційно буфером на маршрутизаторі керує планування пакетів FIFO. Якщо буфер заповнюється, пакети, що надходять, відкидаються. Таке керування чергами на основі планування FIFO називається керуванням чергами з хвостом (TD). Оскільки він простий і легкий у реалізації, TD є найбільш використовуваним алгоритмом керування чергами в Інтернет-маршрутизаторах.

Однак ця технологія має два недоліки: явища блокування та повної черги. Явище блокування може виникнути, коли TD дозволяє кільком підключенням монополізувати ємність буфера. Оскільки маршрутизатор надсилає сигнали про перевантаження джерелам через втрату пакетів лише тоді, коли буфер заповнюється, явище повної черги може зберігатися протягом тривалого часу під час керування чергою TD. Тому маршрутизатору необхідно підтримувати невелику довжину черги з достатньою ємністю буфера, щоб поглинати пакетний трафік даних. Крім того, необхідно виявити перевантаження до того, як воно стане проблемою (тобто до переповнення), щоб ефективно контролювати перевантаження та підтримувати стабільність мережі.

Одним із можливих рішень для подолання недоліків схеми TD є відкидання пакетів до заповнення черги, щоб джерело могло реагувати на перевантаження до переповнення буферів.

Алгоритм AQM відстежує та контролює трафік у межах маршрутизатора, де виникає перевантаження, і контролюється за допомогою більш точної інформації про перевантаження, ніж у джерелах. Ще одна

перевага AQM полягає в тому, що за відсутності співпраці з джерелами або якщо джерела не реагують на контроль перевантаження, AQM здатний контролювати перевантаження виключно на маршрутизаторах. Таким чином, алгоритм AQM діє як контролер доступу на маршрутизаторі для ефективного виявлення та контролю заторів. Рисунок 3.1 ілюструє ранню концепцію відкидання пакетів алгоритму AQM порівняно з традиційним керуванням чергою TD.

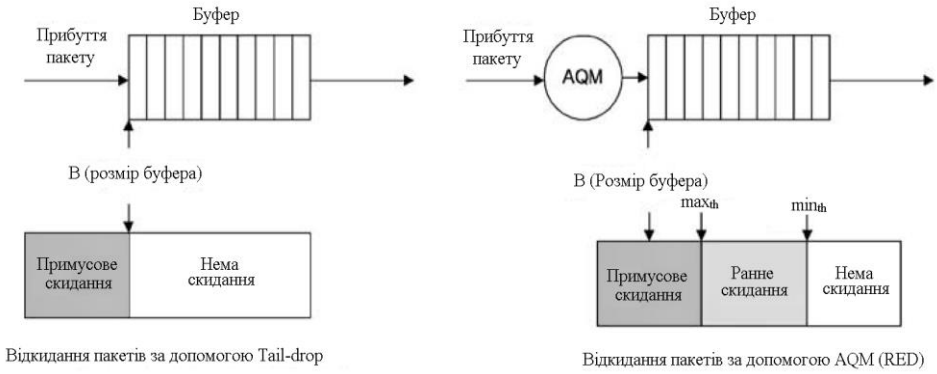

Рисунок 3.1 – Відкидання пакетів із Tail-Drop та AQM

3.3 Пасивне управління чергою

Алгоритм керування чергою – це механізм контролю перевантажень, який вирішує, чи поставити в чергу чи відкинути прибулі пакети на основі поточного стану черги або додаткових обмежень QoS. Перевантаження гарантовано виникне, якщо трафік надходить до вузлів мережі швидше та більше, ніж вузли мережі можуть обробляти протягом тривалого часу. У цей момент затримка в черзі велика, і майбутні пакети повинні бути відкинуті. Затримка в черзі пропорційна довжині черги у вузлі мережі.

До впровадження AQM найбільш використовуване керування чергами було відоме як черга з відкиданням (DT). Вона дотримується простої та інтуїтивно зрозумілої дисципліни: пакет, що надходить, допускається до черги, доки поточна довжина черги менша за фіксований максимальний поріг

довжини; інакше пакет прибуття скидається (рисунок 3.2). Хоча ця схема широко використовується в Інтернеті протягом багатьох років, цей механізм призводить до великої затримки в черзі, якщо розмір черги великий. Крім того, цей тип черги має два інших основних обмеження: тенденція штрафувати вибухові потоки і викликати глобальну синхронізацію між потоками (рисунок 3.3).

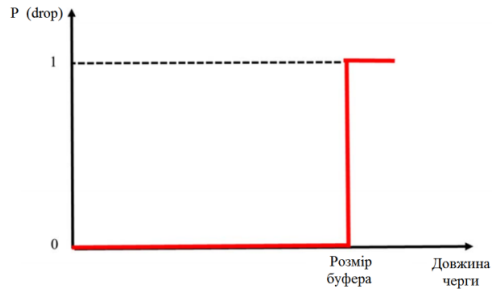

Рисунок 3.2 – Ймовірність відкидання пакету DropTail

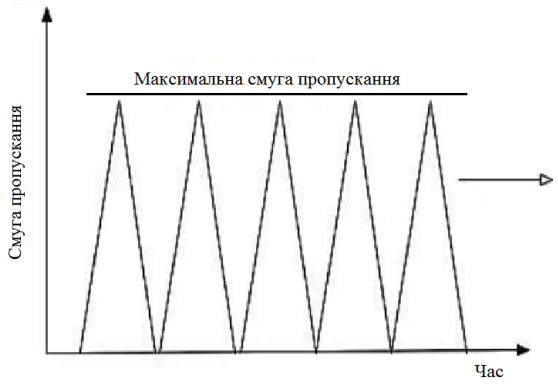

Рисунок 3.3 – Використання смуги пропускання з DT

3.4 Активне керування чергою

Схеми активного керування чергою (AQM), для порівняння, зазвичай активно керують довжиною черги в мережі на основі певних критеріїв і зворотного зв’язку з мережею, щоб підтримувати середню довжину черги на низькому рівні, що призведе до меншої затримки в черзі.

Існує кілька способів класифікації та порівняння різних схем AQM (рисунок 3.4). Залежно від механізму роботи, а саме типу використовуваного індикатора перевантаженості, схеми AQM можна класифікувати на п’ять широких категорій: на основі черги, на основі швидкості, на основі навантаження, на основі втрати пакетів або їх поєднання. Середнє (зазвичай експоненційне зважене ковзне середнє (EWMA)) або миттєві вибірки використовувалися як індикатор перевантаження.

Деякі алгоритми оновлюють EWMA після надходження кожного пакета, тоді як інші виконують оновлення через постійні заздалегідь визначені проміжки часу.

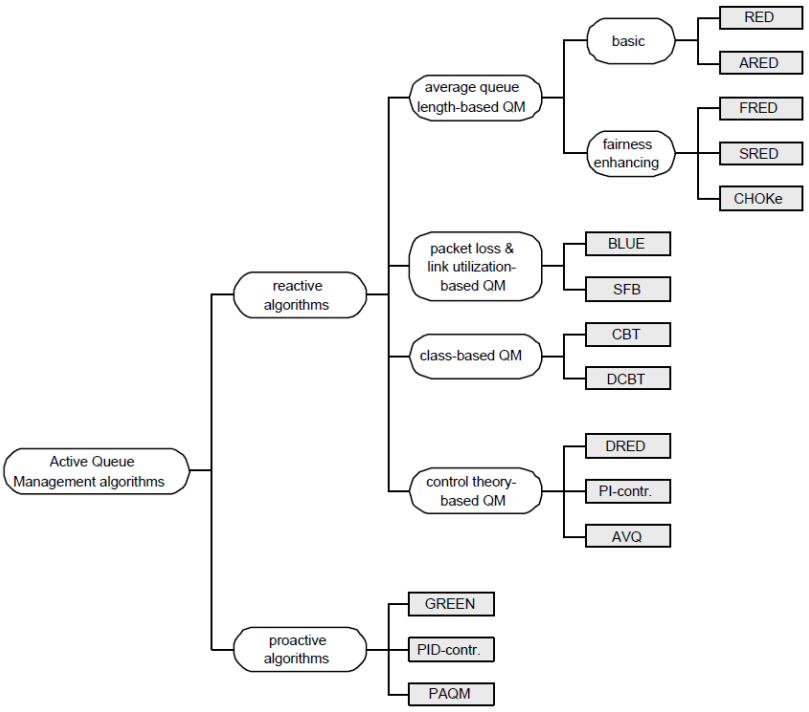

Рисунок 3.4 – Класифікація алгоритмів AQM

Для схеми, заснованої на черзі, ймовірність вилучення виводиться з миттєвої або середньої довжини черги. Основою схем цього типу є підтримання довжини черги у відносно стабільному та бажаному стані. Метою AQM на основі швидкості є підтримання швидкості надходження

пакетів у чергу на цільовому оптимальному значенні, як правило, на певному бажаному відсотку пропускної здатності каналу. За допомогою цього методу довжина черги контролюється опосередковано.

Для схеми, заснованої на навантаженні, ймовірність випадання зазвичай залежить від здатності схеми точно передбачити використання каналу (навантаження трафіку). Зазвичай це агресивне скидання або маркування пакетів даних, коли використання каналу перевищує бажаний рівень, і менш агресивне, коли воно нижче цього рівня.

Цей тип схем, як правило, менше зосереджується на довжині черги, ніж попередні типи, а більше на збалансованому співвідношенні між швидкістю надходження та швидкістю відправлення відповідних посилань. Нарешті, є деякі схеми, які використовують комбінацію цих різних індикаторів, в надії максимізувати переваги механізмів, заснованих як на довжині черги, так і на швидкості завантаження.

3.5 Родина алгоритмів RED

Дві основні функції використовуються в механізмах керування чергами на основі FIFO на маршрутизаторі:

- перша – індикатор перевантаження (для виявлення перевантаження);
- друга – функція контролю перевантаження (для уникнення та контролю перевантаження).

У RED функція контролю покращена шляхом введення імовірнісного раннього відкидання пакетів, щоб уникнути феномену повної черги. RED також покращив індикатор перевантаженості, запровадивши довжину черги EWMA не лише для виявлення заторів, що починаються, але й для згладжування навантаженого вхідного трафіку. Оскільки RED було вперше запропоновано в 1993 році, багато підходів на основі AQM, таких як стабілізований-RED (SRED), BLUE, адаптивний-RED (ARED), REM, адаптивна віртуальна черга (AVQ), динамічний-RED (DRED) і PI-контролер.

SRED [12] відкидає пакети з імовірністю, що залежить від навантаження, на основі оціненої кількості потоків і миттєвої довжини черги. SRED оцінює кількість потоків без ведення облікового запису для кожного потоку.

SRED стабілізує використання буфера на рівні, незалежному від рівня навантаження.

BLUE [13] використовує втрати пакетів і події простою каналу, а не довжину черги для контролю перевантаження. BLUE збільшує ймовірність скидання пакетів у відповідь на переповнення буфера (тобто скидання пакета) і зменшує ймовірність скидання пакетів, коли канал стає неактивним.

ARED [14] намагається підтримувати відповідні робочі параметри в RED шляхом динамічного регулювання max_p на основі спостережуваної динаміки довжини черги. ARED збільшує max_p, коли avg перевищує max_{th}, і зменшує max_p, коли avg падає нижче min_{th}.

REM [8] відокремлює показник перевантаження від показника продуктивності шляхом визначення функції ціни. Філософія REM – «швидкість відповідності та чистий буфер». Таким чином, функція ціни отримується з комбінації невідповідності довжини черги між довжиною черги EWMA та бажаним значенням і невідповідності швидкості між вхідною швидкістю та пропускною здатністю вихідного каналу. Потім пакети відкидаються (або позначаються) з імовірністю, отриманою з функції ціни.

AVQ [15] використовує модифіковану модель відра маркерів як віртуальну чергу (VQ) для регулювання використання буфера, а не довжини черги. AVQ регулює розмір і пропускну здатність каналу VQ пропорційно виміряній вхідній швидкості та відкидає пакети, коли VQ переповнюється.

DRED [16] намагається підтримувати довжину черги EWMA близькою до бажаної довжини черги Q_{ref}, щоб стабілізувати використання навколо заздалегідь визначеного рівня. DRED регулює ймовірність скидання пакетів на основі відхилення довжини черги від Q_{ref}.

PI-контролер [17] намагається підтримувати миттєву довжину черги навколо Q_{ref}. PI-контролер періодично регулює ймовірність скидання пакетів на основі комбінації як поточного відхилення довжини черги, так і суми попередніх відхилень довжини черги від Qref.

Пропозиції AQM можна класифікувати за двома основними функціями, індикатором перевантаження та функцією контролю. Таблиця 3.1 надає таксономію існуючих пропозицій AQM.

Таблиця 3.1 – Характеристики існуючих алгоритмів AQM щодо індикатора перевантаження та функцій керування

AQMs	Індикатор перевантаження	Функція контролю
1	2	3
TD	Миттєва довжина черги	Відстежує миттєву довжину черги Відкидає пакети, коли миттєва довжина черги досягає ємності буфера
RED	Довжина черги EWMA	Зберігає кілька параметрів Раніше відкидає пакети відповідно до серйозності (довгострокового) перевантаження, зазначеного довжиною черги EWMA
SRED	Миттєва довжина черги	Ведення списку Zombie для оцінки кількості потоків Відкидає пакети, використовуючи миттєву довжину черги та передбачувану кількість потоків
BLUE	Миттєва довжина черги та подія простою каналу	Імовірність переривання оновлення за допомогою індикаторів перевантаження збільшується, коли відбувається втрата пакета, і зменшується, коли з’єднання стає неактивним

Продовження таблиці 3.1

1	2	3
ARED	Довжина черги EWMA	Налаштовує max_p відповідно до трафіку. Збільшує max_p, коли довжина черги EWMA перевищує max_{th}, зменьшує max_p, коли довжина черги EWMA менше min_{th}
REM	Довжина черги EWMA та швидкість введення EWMA	Визначає функцію ціни як комбінацію довжини черги EWMA та швидкості введення Налаштовує ймовірність скидання за допомогою функції ціни
AVQ	Швидкість введення	Підтримує маркерне відро як віртуальну чергу (VQ) – оновлює пропускну здатність зв'язку VQ під час надходження кожного пакета. Відкидає пакети лише тоді, коли VQ переповнюється
DRED	Довжина черги EWMA	Підтримує бажану довжину черги (Q_{ref}) Оновлює ймовірність скидання, використовуючи відхилення довжини черги EWMA від Q_{ref}
PI-controller	Миттєва довжина черги	Підтримує бажану довжину черги (Q_{ref}) Періодично оновлює ймовірність скидання, використовуючи відхилення поточної довжини черги та суми (інтеграла) попередньої довжини черги від Q_{ref}

Одним з важливих недоліків запропонованих на даний момент алгоритмів AQM є те, що їх функції виявлення перевантаження та контролю залежать лише від поточного статусу черги або історії статусу черги (наприклад, середньої довжини черги). Таким чином, виявлення та контроль

перевантаження в цих алгоритмах реагують на поточну або минулу перевантаженість, а не проактивно на перевантаження, що починається. Наприклад, метод виявлення заторів у RED може виявляти та реагувати на довгострокові шаблони трафіку, використовуючи експоненціально-зважену ковзну середню (EWMA) довжини черги. Однак він не в змозі виявити початкові затори, спричинені короткочасними змінами навантаження. У цьому випадку неявне сповіщення про перевантаження, що повертається до кінцевих хостів через відкидання пакета, може бути неправильним керуючим сигналом і, можливо, може погіршити ситуацію перевантаження.

3.5.1 Класичний RED

Вперше представлений Флойдом С. і Джейкобсоном В. у 1993 році RED є однією з найвпливовіших, широко вивчених і успішних схем AQM. Він встановлює відносно простий, але ефективний спосіб досягнення мети AQM на основі довжини черги. Це надихнуло багато пізніших досліджень і використовувалося як основа для багатьох інших досліджень і нових схем AQM.

Класичний RED працює наступним чином: довжина черги (q) постійно контролюється; середня довжина черги (avg) обчислюється на основі вагового параметра (wq) миттєвої довжини черги та середньої довжини черги останнього обчислення (avg_0). Для обчислення використовується така функція:

Цілі RED полягають у мінімізації втрат пакетів і затримки в черзі, підтримці високого рівня використання каналу зв’язку та усуненні упереджень щодо вибухів трафіку.

Він також призначений для уникнення глобальної синхронізації, яка виникає, коли всі джерела виявляють перевантаження та одночасно знижують швидкість надсилання, що призводить до коливань використання каналу.

За допомогою RED з'єднання підтримує довжину черги EWMA:

$$Q_{avg} = (1 - w_Q) \cdot Q_{avg} + w_Q \cdot Q,$$

де Q – поточна довжина черги, а w_Q – ваговий параметр, $0 \leq w_Q \leq 1$.

Коли Q_{avg} менше мінімального порогу (th_{min}), жодні пакети не відкидаються. Коли він перевищує максимальний поріг (th_{max}), усі вхідні пакети відкидаються. Коли Q_{avg} знаходиться між ними, пакет відкидається з імовірністю p_d, яка є зростаючою функцією Q_{avg}, тобто:

$$p_d = max_p (Q_{avg} - th_{min})/(th_{max} - th_{min}),$$

де max_p є максимальним значенням p_d.

RED демонструє певні переваги перед DropTail, але він не є досконалим, головним чином через одну або декілька з наступних проблем:

- продуктивність RED дуже чутлива до налаштувань параметрів, у RED принаймні 4 параметри, а саме верхній поріг (th_{max}), нижній поріг (th_{min}), максимальна ймовірність відкидання пакетів (max_p) і ваговий коефіцієнт (w_q) повинні бути встановлені належним чином (фактично, у випадках, коли ці параметри не відповідають умовам мережі, продуктивність маршрутизатору RED може бути навіть гіршою, ніж у DropTail);

- продуктивність RED залежить від кількості конкуруючих джерел/потоків (це проблема у великомасштабній практичній мережі, оскільки інформація про джерела/потоки, швидше за все, невідома менеджеру черги; це також означає, що класичний RED не може вмістити будь-яку схему керування потоком, якщо її не модифікувати);

- продуктивність RED залежить від розміру пакета; це особливо серйозно під час роботи в пакетному режимі замість бітового;

- значні коливання черги спостерігаються під час використання RED, коли змінюється навантаження трафіку.

Лістинг 3.1 – 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
```

Незважаючи на ці труднощі з класичним RED (лістинг 3.1), його потенціал все ще дуже великий. Його було ретельно проаналізовано, розширено або модифіковано багатьма різними способами.

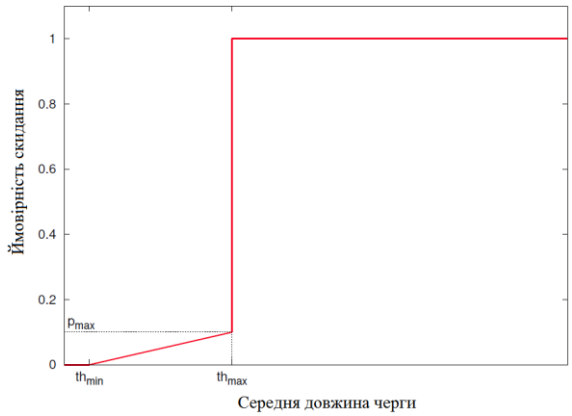

Рисунок 3.5 – Функція ймовірності відкидання пакету в RED, де th_{min} – мінімальний поріг, а th_{max} – максимальний поріг

3.5.2 Gentle-RED

Однією з проблем класичного RED є відносно низька пропускна здатність. Імовірність скидання пакетів безпосередньо стрибає до 1, коли середній розмір черги перевищує верхнє порогове значення (th_{max}). Пізніше це було визнано занадто агресивним і викликає певну нестабільність системи, пов’язану з розривом функції падіння. Імовірність відкидання пакетів, що надходять, коли середня довжина черги трохи нижче верхнього порогу (th_{max}), буде близькою до максимальної швидкості відкидання max_p, яка зазвичай становить значення менше 0,2. Однак, якщо середня довжина черги трохи збільшиться настільки, щоб перевищити верхній поріг (th_{max}), ця ймовірність відкидання для пакетів, що щойно надійшли, раптово стане 1. Це агресивне відкидання спричиняє зниження коефіцієнта використання каналу.

Що ще гірше, через спосіб обчислення середньої довжини черги та більшу частку попередньої довжини черги (оскільки w_q зазвичай є значенням, меншим за 0,2), навіть якщо загальна швидкість передачі вже достатньо сповільнилася через попередні скидання та миттєва довжина черги вже зменшується, розрахунок середньої довжини черги все ще може вказувати, що значення перевищує верхній поріг, і, отже, надмірно агресивно

вживається дія скидання. У результаті швидкість передачі знижується ще більше і далі, і врешті-решт середня довжина черги впаде нижче верхнього порогу, що дозволить новим пакетам надходити всередину черги.

З огляду на це, щадна версія RED (GRED) була запропонована [18] початковою групою дослідників незабаром після оголошення оригінального RED. У GRED ймовірність скидання пакетів лінійно зростає від max_p до 1, коли avg збільшується від th_{max} до 2 th_{max}. Імовірність падіння GRED можна описати так:

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

Хоча це здається лише незначною модифікацією класичного алгоритму RED, ефективність досить помітна (рисунок 3.6). Як наслідок, цю концепцію можна ідентифікувати майже в кожній версії пізнішої пропозиції модифікації RED.

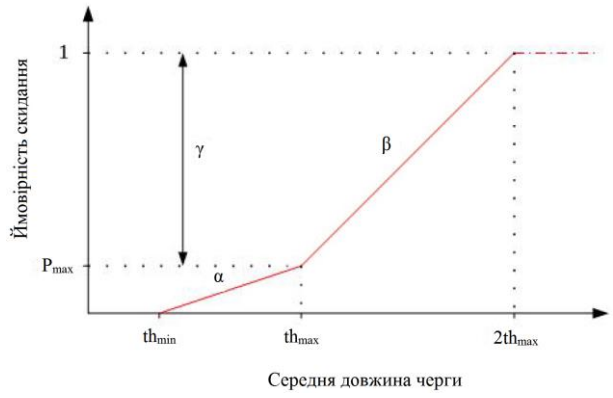

Рисунок 3.6 – Функція ймовірності скидання пакетів в Gentle RED

3.5.3 Адаптивний RED

Дуло доведено, що кількість джерел впливає на ефективність відкидання пакетів для контролю швидкості джерела. Як результат, у поєднанні з проблемою чутливості до параметрів RED, немає єдиного набору параметрів RED, який міг би добре працювати за різних сценаріїв перевантаження. Тому було запропоновано адаптивну схему RED (ARED), яка адаптує значення maxp на основі навантаження на трафік.

В принципі, ARED спостерігає за середньою довжиною черги, щоб визначити, чи є поточне налаштування параметра занадто агресивним. Шлюз, який використовує цю схему, змінює ймовірність вилучення відповідно до середньої довжини черги та визначає, наскільки агресивним є попередній параметр.

Можна помітити, що середня черга коливається навколо thmax, якщо поточне налаштування параметра занадто консервативне, і в іншому випадку коливається навколо th_{min} або навіть менше, якщо поточне налаштування параметра занадто агресивне.

ARED забезпечує певний рівень здатності до самонастроювання порівняно з класичним RED, що певним чином полегшує проблему чутливих параметрів.

Ефективність (середній розмір черги) класичного RED залежить від рівня перевантаження, а також від налаштувань параметрів RED. Версію ARED можна описати наступним псевдокодом, представленим в лістингу 3.2.

ARED використовує параметри мережі за замовчуванням, такі як час проходження за замовчуванням, щоб визначити багато параметрів (найголовніше, max_p), і ці параметри мережі можуть суттєво змінюватися з часом.

Тому ARED не може повністю вирішити проблему чутливості параметрів RED з мережевих сценаріїв. Його продуктивність також може погіршитися за деяких сильніших коливань трафіку.

Лістинг 3.2 – Псевдокод алгоритму ARED

```
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
```

3.5.4 Нелінійний NL-RED

Класичний RED (та його щадна та адаптивна модифікація) використовує лінійну функцію для обчислення ймовірності падіння між пороговими значеннями.

Було запропоновано замінити цю функцію відкидання спеціально розробленою нелінійною функцією, а решта класичного алгоритму RED залишається без змін.

Одним із таких прикладів є модифікація, запропонована Чжоу тощо [19], у якій використовується нелінійна квадратична функція для обчислення ймовірності скидання.

Його основна ідея полягає в тому, що завдяки запропонованій нелінійній функції скидання пакетів швидкість скидання пакетів є нижчою, ніж у класичного RED, при невеликому навантаженні трафіку, але більш агресивною при великому навантаженні.

Як наслідок, при невеликому навантаженні трафіку з’єднувальні потоки заохочуються відносно низькою швидкістю скидання NL-RED, щоб працювати на вищій швидкості передачі, щоб довжина черги шлюзу могла

бути у відповідному діапазоні середніх розмірів черги. які представляють вищу пропускну здатність і краще використання ресурсів. Коли навантаження велике, а середній розмір черги наближається до максимального порогового значення th_{max} – індикатора того, що розмір черги може незабаром вийти за межі бажаного діапазону, відносно вища швидкість скидання NL-RED дає більш агресивний зворотний зв'язок з'єднувальним потокам, що дозволяє довжині черги швидко повернутися до більш відповідного стану.

У своїй версії нелінійного RED, коли середня довжина черги перевищує мінімальний поріг, NL-RED використовує нелінійний квадратичну функцію для відкидання пакетів, а функцію ймовірності відкидання можна описати так:

$$p(t) = \begin{cases} 0 & avg \le th_{\min} \\ (\dfrac{avg - th_{\min}}{th_{\max} - th_{\min}})^2 P_{\max} & th_{\min} \le avg \le th_{\max} \\ 1 & avg > th_{\max} \end{cases}.$$

3.6 AQM на основі швидкості надходження

3.6.1 GREEN

Цей проактивний алгоритм керування чергою, Generalized Random Early Evasion Network (GREEN), використовує параметри потоку та знання поведінки кінцевого хоста TCP, щоб інтелектуально позначати пакети, щоб інтелектуально відкидати пакети, таким чином запобігаючи виникненню перевантажень і забезпечуючи вищий ступінь справедливості. між потоками [20]. Він визначає ймовірність відкидання пакетів на основі аналітичної модел для стаціонарної поведінки TCP.

Функцію ймовірності падіння можна представити як:

$$p_d = \left(\frac{c \cdot s \cdot N}{R \cdot RTT} \right)^2,$$

де p_d – ймовірність відкидання пакета, c – константа, яка залежить від того, як передаються пакети підтвердження, s – максимальний розмір сегмента в мережі, RTT – час проходження потоку в обидві сторони, R – швидкість передачі вихідного посилання, а N – кількість активних потоків у черзі, які мали принаймні один пакет у черзі протягом певного інтервалу часу.

Удосконалення цього алгоритму полягає в тому, що крім тих параметрів, які може визначити або оцінити маршрутизатор, немає параметрів, які потрібно налаштовувати для досягнення оптимальної продуктивності в заданому сценарії.

Крім того, як кількість потоків N, так і час RTT кожного потоку враховуються під час розрахунку ймовірностей сповіщення про перевантаження; таким чином, ймовірність позначення GREEN кольором зазвичай різна для кожного потоку, оскільки залежить від конкретних характеристик потоку.

3.6.2 SFED

Цей алгоритм, Selective Fair Early Detection (SFED) [21], намагається забезпечити справедливий розподіл смуги пропускання для різних потоків і може бути налаштований для призначення пріоритетів агрегатам потоків, як у структурі диференційованих послуг. SFED підтримує маркерне відро для кожного потоку або сукупних потоків, і швидкість, з якою маркери видаляються з цього відра, пропорційну швидкості вхідного потоку. Наповненість відра потім використовується як визначення ймовірності

скидання/позначення пакета. Він використовує висоту бакетів маркерів, щоб представити довжину вибуху трафіку, яку він може вмістити для кожного відповідного потоку. Сума висот відер пропорційна розміру буфера. Коли додається новий потік, усі висоти зменшуються, щоб розмістити нове маркерне відро, в іншому випадку висота всіх інших сегментів маркерів збільшується, коли потік стає неактивним. Функцію ймовірності скидання можна представити у вигляді:

$$p_d = \begin{cases} p_{\max}\left(\dfrac{\lambda_1 - \dfrac{x_i}{L_N}}{\lambda_1 - \lambda_2}\right) & \lambda_2 < \dfrac{x_i}{L_N} < \lambda_1 \\ 0 & \lambda_1 < \dfrac{x_i}{L_N} < 1 \\ p_{\max} + (1 - p_{\max})\left(\dfrac{\lambda_1 - \dfrac{x_i}{L_N}}{\lambda_2}\right) & 0 < \dfrac{x_i}{L_N} < \lambda_2 \end{cases},$$

де x_i – зайнятість і-го маркерного відра, N – кількість потоків або відер у системі, L_N – максимальна висота кожного відра. λ_1 і λ_2 – порогові значення коефіцієнта зайнятості.

SFED намагається досягти раннього виявлення та сповіщення про перевантаження для адаптивного джерела.

3.7 На основі комбінованих індикаторів

3.7.1 REM

RED вимірює перевантаженість за середньою довжиною черги, в результаті його показники продуктивності, такі як затримка або пропускна здатність, пов’язані з мірою перевантаження. Оскільки середній розмір черги

відображає зміну навантаження трафіку, якщо параметр не встановлено ідеально, RED важко підтримувати продуктивність у цільовій зоні незалежно від кількості користувачів.

Алгоритм Random Exponential Marking (REM), з іншого боку, спрямований на досягнення високої пропускної здатності, низького рівня втрат і низької затримки в черзі за змінних навантажень трафіку шляхом незалежного використання показника перевантаження та показника продуктивності.

Міра перевантаження представляє надлишковий попит на пропускну здатність або ресурс. Ключ REM полягає у визначенні ймовірності маркування за допомогою ціни як міри перевантаження. Ціна базується на невідповідності тарифів і черги.

Невідповідність швидкості – це вимірювання таких речей, як різниця між швидкістю введення та пропускною спроможністю каналу, тоді як невідповідність черги – це вимірювання таких речей, як різниця між розміром черги та ціллю. Для обох ціна відображає не тільки поточну перевантаженість, але й зміну перевантаженості. У черзі l ціна $price_l(t)$ оновлюється в кожному періоді t таким чином:

$$price_l(t+1) = [price_l(t) + \gamma(\alpha_l(q_l(t) - q_{ref}, l) + x_l(t) - C_l(t))],$$

де $x_l(t)$ – це сукупна вхідна швидкість у черзі l, $C_l(t)$ – доступна пропускна здатність у черзі l. Константа α_l є малим додатним числом. Його можна налаштувати окремо для кожної черги для компромісу між використанням і затримкою в черзі протягом перехідного періоду. Константа γ є малим додатним числом. Вона контролює реакцію REM на зміни в мережевих сценаріях.

Якщо цільовий розмір черги q_{ref},l не дорівнює нулю і в буфері є пакети, то $x_l(t) - C_l(t)$ є зміною розміру черги в часі. У цьому випадку його можна замінити на $q_l(t + 1) - q_l(t)$.

Таким чином, ціна оновлюється лише на основі поточного та попереднього розмірів черги, як зазначено нижче:

$$price_l(t+1) = [price_l(t) + \gamma(q_l(t+1) - (1-\alpha_l)q_l(t) - \alpha_l q_{ref}, 1)].$$

Тоді REM може використовувати наступну політику експоненціального маркування для розрахунку ймовірності маркування:

$$p_l(t) = 1 - \varphi^{-price_l(t)},$$

де $\varphi>1$ – константа. Якщо вхідний пакет не позначено жодною чергою висхідного потоку, ймовірність його маркування експоненціально зростає в поточній ціні.

3.7.2 BLUE

BLUE прагне зменшити рівень втрати пакетів і вимоги до розміру буфера більш простим способом, а не постійним налаштуванням своїх параметрів для досягнення прийнятної продуктивності за змінних умов мережі, як класичний RED. Ключова ідея BLUE полягає в тому, щоб керувати чергою безпосередньо на основі втрати пакетів і використання каналу, а не миттєвого чи середнього розміру черги.

BLUE безпосередньо використовує втрату пакетів і використання каналу як міру перевантаження для встановлення ймовірності маркування. В принципі, BLUE підтримує єдину ймовірність p, яку він використовує для позначення (або відкидання) пакетів, коли вони стоять у черзі. Якщо черга постійно відкидає пакети через переповнення буфера, ймовірність збільшується, таким чином збільшуючи швидкість, з якою він надсилає сповіщення про перевантаження. І навпаки, якщо черга стає порожньою або якщо посилання неактивне, BLUE зменшує ймовірність позначення.

Це фактично дозволяє BLUE дізнатися правильну швидкість, необхідну для надсилання сповіщення про перевантаження. Використання порогу забезпечує простір для розміщення перехідних сплесків. Псевдокод алгоритму BLUE представлено в лістингу 3.3.

Лістинг 3.3 – 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
```

Використання часу заморожування дозволяє змінам у ймовірності маркування набути чинності до того, як значення буде оновлено знову, і його також можна використовувати для контролю чутливості алгоритму при роботі зі зміною швидкості трафіку.

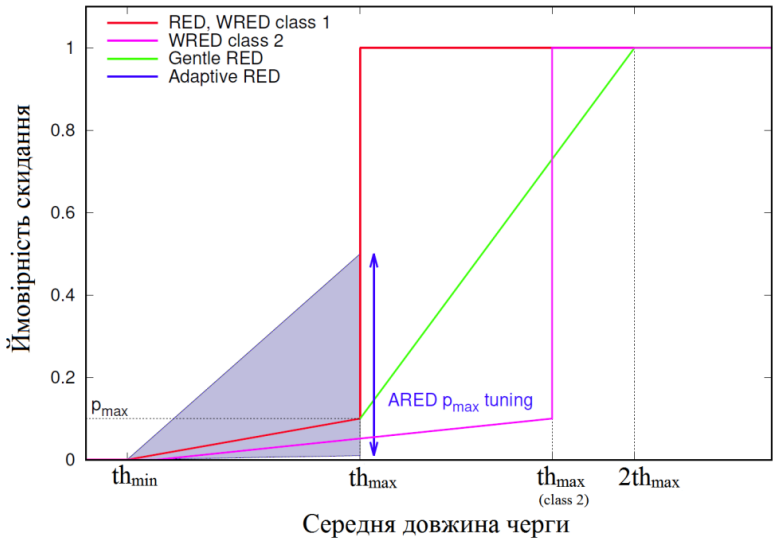

Рисунок 3.7 – Різновиди алгоритмів RED

4 РОЗРОБКА МЕТОДУ З НЕЧІТКОЮ ЛОГІКОЮ

4.1 Недоліки існуючих методів

Перевантаження на маршрутизаторі виникає, коли буфер, до якого мають бути розміщені передані пакети, переповнений. Затори в основному виникають через інтенсивний трафік. Оскільки навантаження трафіку зростає з низькою швидкістю відправлення, з часом у буфері розміщується більше пакетів, що зрештою призводить до переповнення буфера. У цей момент пакети, що надійшли, не можуть бути прийняті, що призводить до збільшення втрати пакетів, погіршення затримки та значного зниження пропускної здатності. Оскільки мережа пов’язана з маршрутизаторами, наслідки на маршрутизаторі поширюються на всю мережу.

За допомогою DT пакети відкидаються, коли пакети в черзі досягають порогового значення. Проте проблема техніки DT полягає в нездатності впоратися з раптовими перевантаженнями, які в кінцевому підсумку призводять до втрати пакетів.

Активне керування чергами (AQM) використовується для активного моніторингу буфера маршрутизатора, уникнення перевантажень і полегшення серйозних наслідків різкого трафіку.

На основі індикатора продуктивності AQM вирішує, прийняти чи скинути прибулий пакет, щоб уникнути переповнення буфера. AQM не залежить від фіксованого порогу, порівняно з DT; натомість AQM стохастично відкидає пакети залежно від навантаження на мережу.

Індикатори продуктивності використовуються з метою моніторингу та для розрахунку ймовірності падіння (D_p). Метою D_p є встановлення стохастичного механізму для відкидання пакетів, щоб уникнути глобальної синхронізації. Відповідно величина D_p залежить від характеристики результативного показника.

Метою запропонованих методів було покращити обчислення D_p для підвищення продуктивності мережі. Відповідно, були використані різні індикатори, такі як середня довжина черги (AQL), довжина черги екземпляра (Q), швидкість надходження пакетів, швидкість завантаження, оцінена затримка тощо. AQM на основі нечіткої інформації використовує вхідні змінні, еквівалентні показникам продуктивності, що використовуються в нечітких методах (рисунок 4.1). Показник продуктивності AQL – це відображення вхідних змінних Q і ΔQ [22].

Існуючі методи, які належать до обох методів (звичайних і нечітких), були зосереджені на зменшенні втрати пакетів і уникненні непотрібного відкидання пакетів.

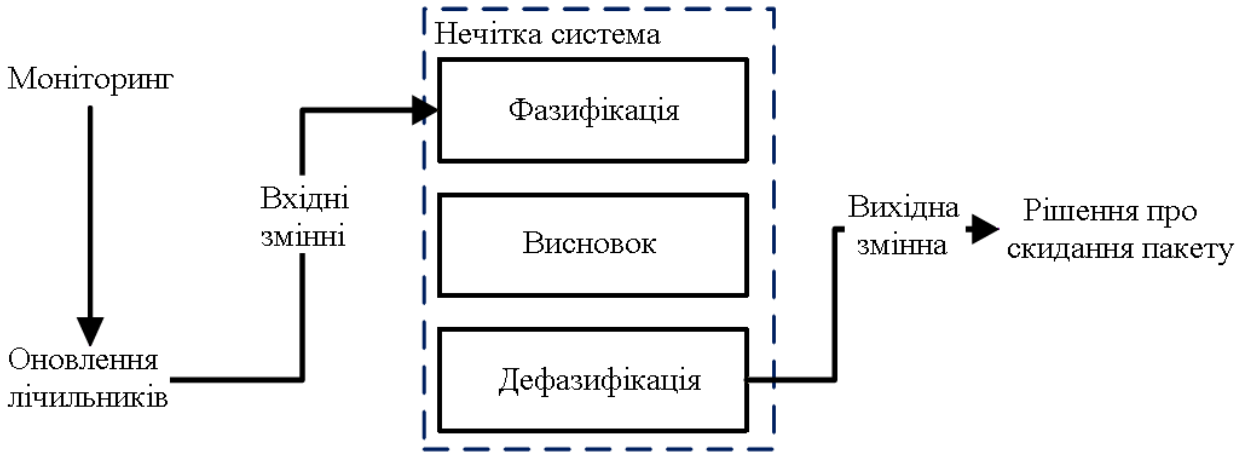

Рисунок 4.1 – AQM на основі нечіткості

Як правило, наслідки на маршрутизаторі поширюються на всю мережу. Відповідно, якість послуг і бажана продуктивність, пов’язані зі збільшенням пропускної здатності та зменшення затримок і втрат, вимагають кількох показників продуктивності, які враховують затримку, скидання та втрати пакетів в буфері маршрутизатора. Існуючі методи AQM використовують щонайбільше один або два показники ефективності, які не можуть зафіксувати бажану продуктивність. Крім того, єдиний індикатор ефективності в звичайній техніці вимагає кількох лічильників і, отже, кількох

вхідних змінних у AQM на основі нечіткості. Таким чином, існує обмеження у використовуваних показниках, вхідних змінних, які впливають на продуктивність мережі.

Алгоритм RED був запропонований на основі підтримки значення AQL і використання обчисленого значення AQL для обчислення D_p і прийняття рішення про скидання пакету. Алгоритм був побудований на основі міркувань на основі випадків із різними параметрами. Окрім передбачення перевантажень, уникнення втрат і непотрібного відкидання пакетів, RED створено для уникнення глобальної синхронізації та, наскільки це можливо, відкидання послідовних пакетів.

Існують різні параметри та порогові значення, які були визначені та використані для вирішення цих кількох цілей, як зазначено в лістингу 4.1. Параметри ініціалізуються в рядку 1. Потім AQL обчислюється в рядках 3-4. Стохастичне відкидання реалізовано в рядках 5-9. Повне видалення реалізовано в рядку 10, а відсутність – у рядку 11. Оновлення часу реалізовано в рядках 12-13.

Лістинг 4.1 – Псевдокод алгоритму RED

```
Step 1.  INITIALIZATIONS: AQL:=0, Count = −1
Step 2.  FOR-EACH Arrival-Packet(a) DO
Step 3.      IF (Q==0) THEN
                AQL ≔ (1 − q)f(cTime−iTime) ∗ AQL
Step 4.      ELSE   AQL ≔ (1 − q) ∗ AQL + q ∗ AQ
Step 5.      IF (minth ≤ AQL < maxth) THEN
Step 6.        Count ++
Step 7.        Dp′ = Dmax ∗ (AQL − minth)/ (maxth − minth)
Step 8.        Dp = Dp′/(1 − Count ∗ Dp′)
Step 9.        IF (Drop(Dp) == TRUE) THEN
                  Drop Packet (a), Count:= 0
Step 10.       ELSE IF (AQL > maxth) THEN
                  Drop Packet (a), Count:= 0
Step 11.          ELSE Count:= -1
Step 12.       IF (Q==0&&Idle:=FALSE) THEN
                  iTime =cTime, Idle:=TRUE
Step 13.         ELSE Idle := FALSE
```

Застосовані в лістингу 4.1 змінні:

- Count – лічильник для відображення останнього значення розміщення;
- AQL – середня довжина черги;
- Q – довжина черги екземпляра;
- cTime – поточний час;
- iTime – час простою;
- w – вагове значення;
- D_{max} – значення максимальної ймовірності;
- th_{min} – мінімальне порогове значення;
- th_{max} – максимальне порогове значення;
- Idle – змінна для позначення бездіяльність трафіку.

Зі зростанням розподілених і віддалених програм і спільних ресурсів, використання мережевих ресурсів зростає, що створює нові вимоги до управління чергою в буфері маршрутизатора. Таким чином, виникла потреба зменшити відкидання пакетів без наслідків. Крім того, необхідна швидка реакція на раптові затори та збільшення транспортного навантаження.

Загальна форма використання AQL в AQM використовується на рисунку 3.2.

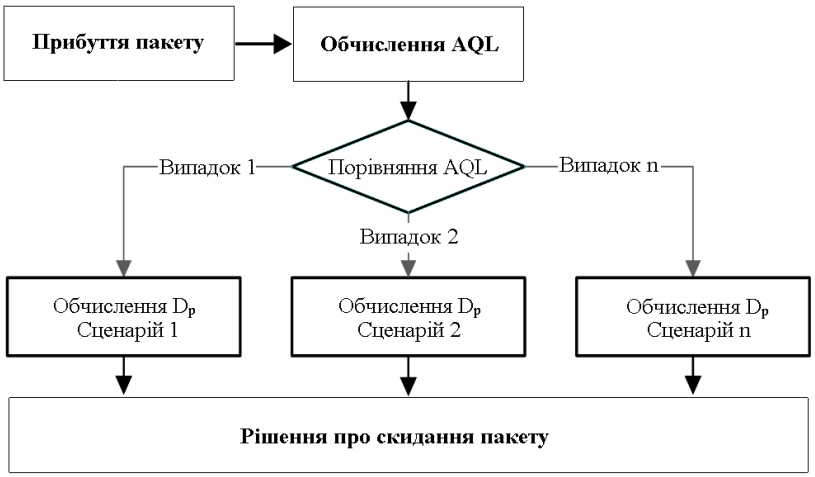

Рисунок 4.2 – Алгоритм AQM на основі AQL

Коли навантаження трафіку збільшується у черзі, D_p збільшується, і навпаки. Стабілізований AVQ і покращений AVQ (EAVQ) були створені як розширення AVQ. Випадкове експоненціальне маркування (REM) використовує довжину черги (q) і швидкість передачі як показники продуктивності. Пропорційно-інтегральний контролер (PI) використовує q і втрату пакетів як показники продуктивності. Алгоритм BLUE використовує втрату пакетів як показник продуктивності та адаптивний розрахунок D_p, які повністю відрізняються від способу реалізації RED.

Відповідно, D_p не розраховується, а адаптивно збільшується/ зменшується на основі індикатора та простою каналу. Так само багаторівневий RED (MRED) використовує втрати як показник ефективності. Ефективний RED (ERED) розширює RED і використовує AQL і q як індикатори перевантаження.

Оскільки використовуються складніші показники ефективності, такі як швидкість передачі та оцінені втрати, використовується більше параметрів. Проблема ініціалізації параметрів зростала, і виникла потреба вирішити проблеми покращення продуктивності мережі. Fuzzy-RED (FRED) використовував AQL із задачею нечіткого висновку, щоб полегшити проблему налаштування параметрів. Подібним чином Fuzzy BLUE (FB) [23] використовував розмір черги та втрату пакетів як показники продуктивності. Ці методи створили єдину вихідну змінну, якою є D_p. Випадкове раннє виявлення нечіткого контролера (FCRED) використовувало відмінності між фактичною та цільовою довжиною черги та зміну такої різниці як вхідні змінні для створення контрольного значення, яке використовується для обчислення значення D_p.

Загалом існуючі методи AQM на основі нечіткості відрізняються п’ятьма аспектами, а саме:

- вхідними змінними;
- вихідною змінною;
- функцією членства;

- нечіткими множинами;
- нечітким набором правил.

Загалом існуючі AQM на основі нечіткості використовують загальні вхідні змінні, такі як черга, AQL, затримку та втрату, щоб отримати D_p. Ці вхідні дані є індикаторами, які були відображені на основі традиційної AQM і розраховані перед тим, як їх можна було використовувати з нечіткими системами. Серед цих змінних є деякі спільні характеристики, наприклад використання черги для обчислення AQL, затримки та частоти прибуття. Використана комбінація цих змінних не охоплює всіх показників ефективності. Таким чином, вхідні змінні не є незалежними та не всеосяжними. Короткий перелік використаних показників та їхніх внутрішніх змінних узагальнено в таблиці 4.1.

Таблиця 4.1 – Підсумок використаних показників

Індикатор	Змінна, с
Q	Довжина черги
AQL	Q і попередня AQL
Q	Q і попередня Q
Швидкість прибуття	Кількість пакетів, що надійшли
Швидкість відправки	Кількість відправлених пакетів, швидкість надходження та швидкість відправлення
Навантаження	
Затримка	Q та ΔQ

Відповідно, щоб оптимізувати продуктивність AQM, вхідні змінні повинні бути проаналізовані, і їх комбінація повинна охоплювати всі бажані показники ефективності. Крім того, за винятком Fuzzy RED, існуючі нечіткі методи використовують дві вхідні змінні, яких недостатньо для покриття бажаних показників продуктивності. Далі буде проаналізовано показники продуктивності, а основні компоненти цих змінних будуть використані як

вхідні дані для нечіткої системи. Використання трапеції забезпечує більшу гнучкість у процесі та може бути легко відображено у трикутну, але трикутну не можна відобразити у трапецію. Нарешті, нещодавні дослідження зосереджені на охопленні всіх можливих комбінацій вхідного набору, кожен з окремими правилами, щоб уникнути будь-якої складності. Таким чином, у запропонованому методі буде використано трапецієподібну функцію належності та повний набір правил.

4.2 Запропонований метод AQM

Запропонований метод AQM створено для оптимізації продуктивності мережі шляхом охоплення метрик продуктивності на вимогу. Структура для побудови запропонованого методу, як показано на рисунку 4.3, складається з наступних кроків: 1) визначити вхідні змінні шляхом аналізу критеріїв втрат, затримки та пропускної здатності; 2) визначити компоненти фазифікації, якими є лінгвістичні множини та функції належності; 3) налаштувати набір правил методом проб і помилок, регулюючи функцію належності вихідних змінних; 4) дайте визначення процесу агрегації; 5) налаштувати середовище моделювання.

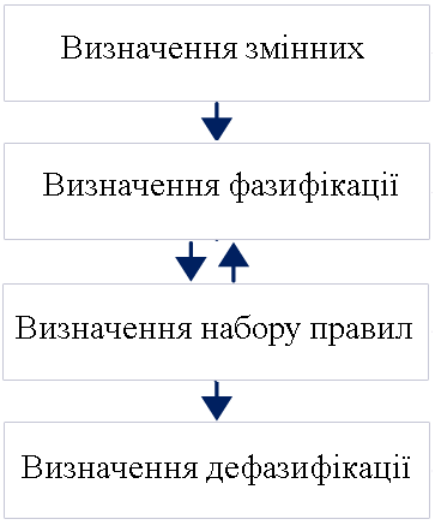

Рисунок 4.3 – Етапи розробки запропонованого методу

4.3 Вхідні змінні

Вхідні змінні отримуються на основі аналізу трьох показників ефективності; це затримка, пропускна здатність і втрати. Затримку можна оцінити на основі швидкості передачі та довжини черги. Затримка пропорційна довжині черги, оскільки прибулий пакет має чекати в черзі FIFO (першим увійшов – першим вийшов), яка зазвичай реалізована в буфері маршрутизатора.

Крім того, затримка пропорційна швидкості передачі, оскільки збільшення відправлення λ_{out} зменшує затримку, а збільшення швидкості прибуття λ_{in} збільшує затримку в залежності від довжини черги. Різницю між λ_{out} та λ_{in} можна оцінити зі зміни довжини черги з плином часу.

Відповідно, передбачувану затримку можна позначити як:

$$d_{queue} \propto q \cdot \Delta q, \tag{4.1}$$

де d_{queue} – затримка у черзі. Втрата пакетів (Packet Loss – PL) обернено пропорційна залишковій ємності (v) буфера маршрутизатора. У міру зменшення залишкової ємності, ймовірність втрати пакетів збільшується і навпаки. Відповідно, передбачувану втрату можна позначити як:

$$PL \propto 1/v. \tag{4.2}$$

Пропускну здатність можна оцінити на основі відкидання та втрати. Відкидання обернено пропорційно довжині черги, оскільки AQM відповідає за раннє відкидання пакетів у сильно завантаженій мережі. Втрати, як згадувалося, залежить від ємності, що залишилася (v). Відповідно, оцінну пропускну здатність (T) можна позначити як:

$$T \propto v / q. \tag{4.3}$$

Відповідно, з показників продуктивності витягуються три змінні: довжина черги (q*), зміна в черзі (Δq*) і ємність (v*), що залишилася. Значення цих змінних обчислюються та нормалізуються у діапазоні [0, 1]. Змінна q, яка є лічильником кількості пакетів, поділяється на ємність буфера (c). Значення Δq обчислюється як різниця між двома послідовними захопленими довжинами, поточною та попередньою, а потім нормалізується в діапазоні [0, 1]. Таким чином, якщо різниця позитивна, обчислене значення буде в діапазоні (0,5, 1], а діапазон [0, 0,5) буде для негативних різниць. Значення 0,5 означає, що різниця дорівнює нулю. Змінна v, яка є лічильником позицій, що залишилися, ділиться на ємність буфера (c). Ці змінні використовуються як вхідні змінні для запропонованого методу на основі опису, який узагальнено в таблиці 4.2.

Таблиця 4.2 – Зведення вхідних змінних

Змінна	Розрахунок	Опис
Нормована довжина черги (q*)	$q^*_t=q_t/c$	Довжина черги на основі прийнятих пакетів
Зміна черги (Δq*)	$\triangle q^*_t=(q_t-q_{t-1}+c)/(2*c)$	Різниця в довжині черги між поточним і попереднім часом
Нормована залишкова ємність (v*)	$v^*_t=(c-q_t)/c$	Максимальна довжина черги на основі пакетів, які можуть бути розміщені одночасно

4.4 Фазифікація

Значення вхідних змінних перетворюється на лінгвістичний терм зі ступенем приналежності з використанням пов'язаної нечіткої множини та функції приналежності. Відповідно до підходу розбиття простору для побудови нечіткої системи діапазон значень вхідної змінної ділиться на 2N+1 рівних областей. Значення N встановлюється рівним 1 для вхідних змінних.

Таким чином, лінгвістичний набір вхідних змінних встановлюється як низький, помірний, високий. Функція приналежності для вхідних змінних є трапецієподібною, як показано на рисунку 4.4.

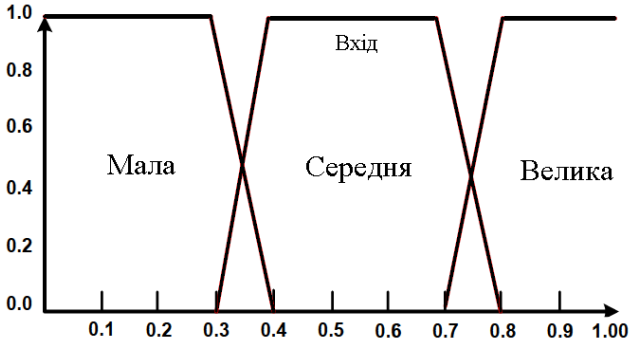

Рисунок 4.4 – Функції приналежності вхідних змінних у запропонованому методі

Кожна область у функції характеризується чотирма точками {a, b, c, d}, де a і b утворюють першу лінію, а друга лінія побудована за допомогою c і d. Значення цих балів для вхідної функції такі: низький 0, 0, 0,3, 0,4; помірний 0,3, 0,4, 0,6, 0,7; високий 0,6, 0,7, 1,0, 1,0. Відповідно, у процесі фазифікації чітке значення вхідної змінної перетворюється на термін або декілька термінів на основі діапазону значень, у якому виникає значення. Ступінь членства (μ) з кожним терміном обчислюється як:

$$\mu(x) = \begin{cases} 0 & if\ x < a, x > d \\ (x-a)/(\text{b}-\text{a}) & if\ a \le x \le b \\ 1 & if\ b \le x \le c \\ (d-x)/(d-c) & if\ c \le x \le d \end{cases}.$$

Відповідно, для значення кожної вхідної змінної витягується один або більше лінгвістичних термінів. На основі використовуваної функції членства та межі регіонів отримані терміни пов’язані зі ступенем членства.

4.5 Вихідна змінна

Вихідна змінна D_p пов'язана з функцією приналежності з рівними областями, подібними до вхідних змінних. Значення N встановлено рівним 2 для вихідної змінної, і, отже, лінгвістичний набір для виходу {нуль, мала, середня, велика, екстремальна}. Базуючись методом проб і помилок, межі приналежності D_p модифіковано, як показано на рисунку 4.5. Значення граничних точок для вихідних функцій модифіковано відповідно до бажаного результату таким чином: нуль 0, 0, 0,005, 0,01; низький 0,005, 0,01, 0,4, 0,5; помірний 0,4, 0,5, 0,6, 0,7; високий 0,6, 0,7, 0,8, 0,9; екстремальний 0,8, 0,9, 1,0, 1,0. У функції приналежності результату змінна використовується в процесі дефазифікації, в якому лінгвістичний термін перетворюється на чітке значення.

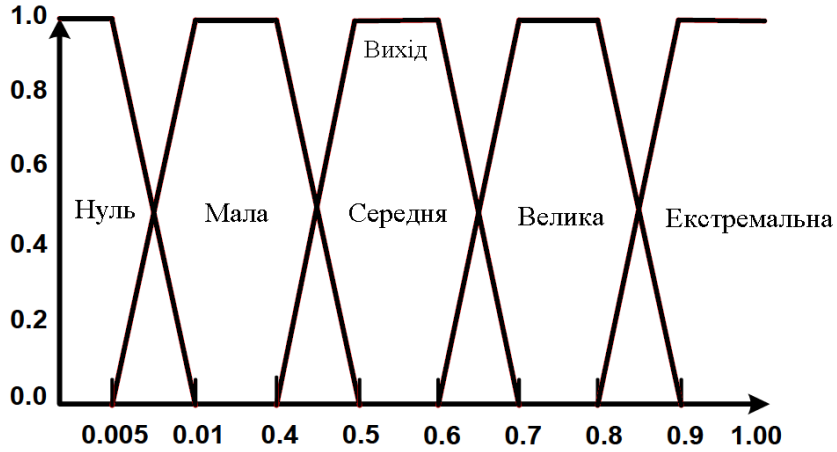

Рисунок 4.5 – Функції приналежності для вихідної змінної

4.6 Набір правил

Метод проб і помилок ототожнює набір правил із досвідом у цій галузі разом із тими правилами, які можна зробити з [24]. Нечіткі правила, які будуються як повний набір, означають, що кожна можлива комбінація термінів у вхідних змінних пов’язана з одним правилом.

Відповідно, для трьох вхідних змінних із трьома можливими термінами в перехресній матриці генерується 27 правил, як наведено в таблиці 4.3.

Як наведено в таблиці 4.3, правила з набору правил подано таким чином: якщо Q мала і ΔQ мала, а v велика, то D_p середня. Решта правил витягуються з перехресної матриці аналогічно.

На етапі оцінки правил на основі термінів вихідних даних застосовується правило і ступінь достовірності A пов'язана з частиною правила THEN, яка виводиться як вихід. Ступінь достовірності для кожного із вхідних даних визначається за допомогою оператора I.

Відповідно, мінімальне значення ступенів належності вхідних даних вибирається як ступінь достовірності вихідного члена.

Таблиця 4.3 – Перехресна матриця набору правил

Q	Мала			Середня			Велика		
ΔQ / v	Мала	Середня	Велика	Мала	Середня	Велика	Мала	Середня	Велика
Велика	Нуль	Нуль	Нуль	Мала	Мала	Велика	Екстремальна	Екстремальна	Екстремальна
Середня	Нуль	Нуль	Нуль	Мала	Середня	Велика	Екстремальна	Екстремальна	Екстремальна
Мала	Нуль	Нуль	Мала	Середня	Середня	Велика	Екстремальна	Екстремальна	Екстремальна

4.7 Агрегація та дефазифікація

На етапі агрегування схожі вихідні терміни агрегуються. Таким чином, на основі виходу оцінки правил, кілька правил можуть бути оцінені в подібний термін, кожне з яких зі ступенем достовірності. Ці терміни з їх ступенями достовірності об’єднуються в один термін і один ступінь. Наприклад, враховуючи, що оцінка правила створила два виходу терміна мала з двома значеннями довіри, результатом кроку агрегування буде термін мала з одним значенням довіри.

Загалом, агрегування реалізовано за ступенями достовірності частини THEN- правил з ідентичними термінами.

На етапі дефазифікації умови D_p з їх сукупним ступенем достовірності перетворюються на чітке значення. Для цього використовується функція центру тяжіння (Center of Gravity-CoG), яка формує усереднення за функцією приналежності [25].

Метод центра тяжіння розраховує дефазифіковане значення як центр ваги під кривою функції належності. Це досягається шляхом множення кожного можливого значення на його відповідний ступінь приналежності і сумування отриманих результатів, а потім ділення цієї суми на суму всіх ступенів приналежності.

Результатом буде сума всіх результатів правил виводу (w_i,z_i) усереднена сумою всієї вихідної чисельності (z_i). Підсумковий розрахунок в може бути представлений формулою:

$$\text{fuzzy\_result} = \frac{\sum_{i=1}^{N} w_i z_i}{\sum_{i=1}^{N} z_i}.$$

4.8 Середовище моделювання

Моделювання реалізовано в інтегрованому середовищі розробки (IDE) NetBeans і Java Development Kit (1.6). Змодельована мережа складається з вхідних каналів, маршрутизатора та вихідних каналів. Маршрутизатор пов'язаний з буфером ємністю 20 пакетів. Черга з дискретним часом використовується для моделювання потоку трафіку в мережі, що моделюється.

Відповідно, швидкість прибуття та відбуття пакетів залежить від значень ймовірності. Значення перераховуються для створення різного трафікового навантаження на змодельовану мережу. Швидкості надходження вибрано такими, що мають значення [0,3, 0,5 і 0,95], а відправлення – значення 0,5.

Введемо наступні позначення:

- v – ємність сховища маршрутизатора (пак);
- λ– швидкість прибуття (пак/с);
- μ – швидкість відправлення (пак/с).

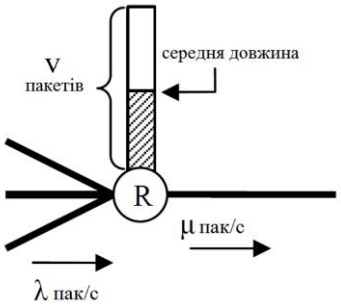

Рисунок 4.6 – Модель маршрутизатора

Створено три сценарії з такими ймовірностями прибуття та відправлення, легкий трафік, помірний та інтенсивний трафік. Процес надходження пакетів моделюється як процес Бернуллі. Компоненти моделювання наведено на рисунку 4.7.

Рисунок 4.7 – Компоненти моделювання

4.9 Експериментальні результати

Пропоновані та порівнювані методи оцінюються при трьох навантаженнях трафіку: легкий трафік, помірний і інтенсивний трафік. Порівняння реалізується на основі затримки, швидкості відкидання та втрати

пакетів, крім того, повторно передані пакети можуть дати краще представлення при порівнянні методів. Відповідно, це використовувалося в якості ще одного критерію для порівняння методів.

При легкому трафіку, як показано на рисунку 4.8, пропонований та порівнювані методи працювали однаково з розумною затримкою. Не було ні втрати пакетів, ні відкидання, оскільки черга ніколи не переповнюється таким трафіком, і відкидання пакетів не потрібне.

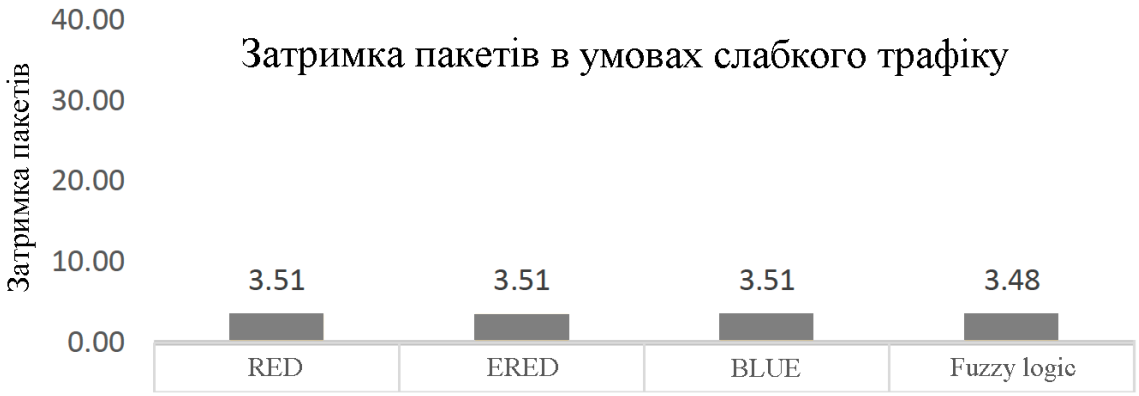

Рисунок 4.8 – Затримка пакетів в умовах слабкого трафіку

В умовах помірного трафіку пропонований метод показав меньшу затримку в порівнянні з RED та ERED і трохи гірше, ніж BLUE, як показано на рисунку 4.9.

Тим не менш, BLUE не показав кращу продуктивність в цілому, оскільки BLUE досяг цієї затримки в черзі через непотрібне відкидання пакетів, як показано на рисунку 4.10. Подібно до слабкого трафіку, не було втрати пакетів через використання будь-якого з порівнюваних методів.

Відповідно, пропонований метод перевершив порівнювані методи, відкидаючи відповідну кількість пакетів для запобігання втрат і збереження затримки. У той же час RED та ERED відкидали трохи менше пакетів та жертвували затримкою. BLUE відкидає майже вдвічі більше пакетів у порівнянні з пропонованим методом і забезпечує трохи кращу затримку.

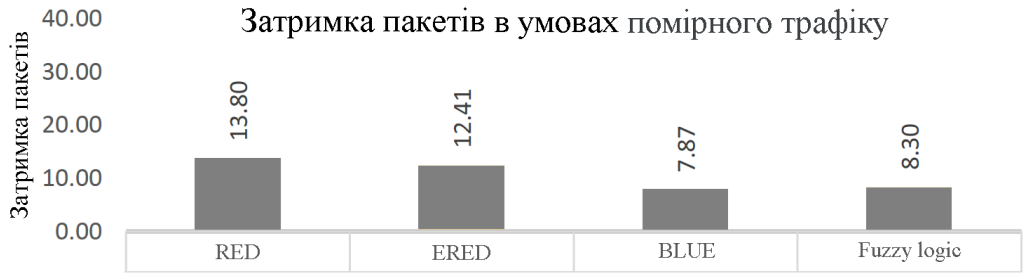

Рисунок 4.9 – Затримка пакетів в умовах помірного трафіку

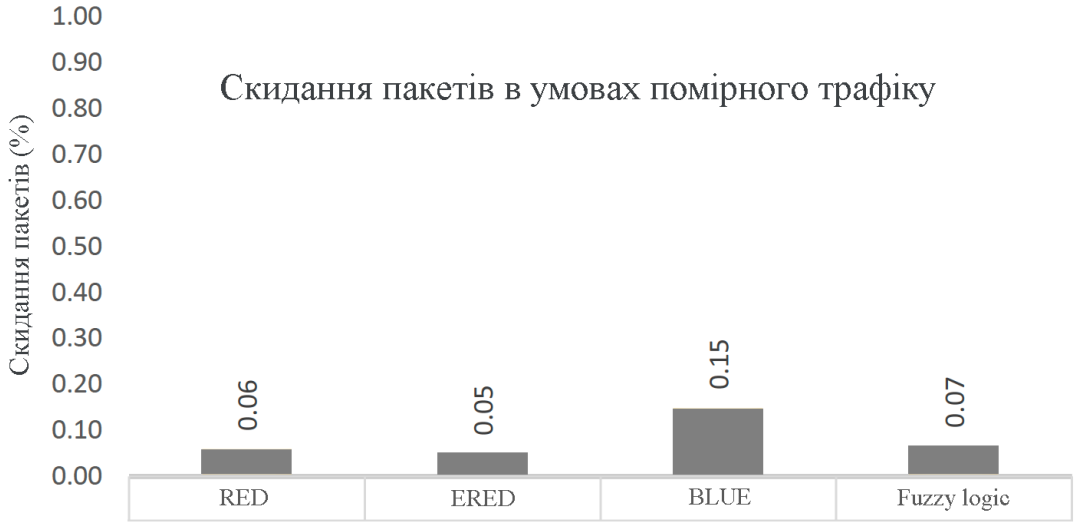

Рисунок 4.10 – Скидання пакетів в умовах помірного трафіку

В умовах інтенсивного трафіку, що є основною проблемою методів AQM, результати близькі до результатів за умов помірного трафіку, але з чітко помітними деталями. Запропонований метод дав меншу затримку порівняно з RED та ERED та гірше, ніж BLUE, як показано на рисунку 4.11. ERED дає дуже велику затримку, якій неможливо протистояти в деяких часто використовуваних програмах, таких як відеоконференції. BLUE дає розумну затримку, але не дає кращої продуктивності в цілому, тому що BLUE досягає цієї затримки в черзі внаслідок непотрібного відкидання пакетів, як показано на рисунку 4.12. ERED втрачає значний обсяг даних, як показано на рисунку 4.13. Використовуючи повторну передачу як критерій, ясно, що запропонований метод краще, ніж BLUE.

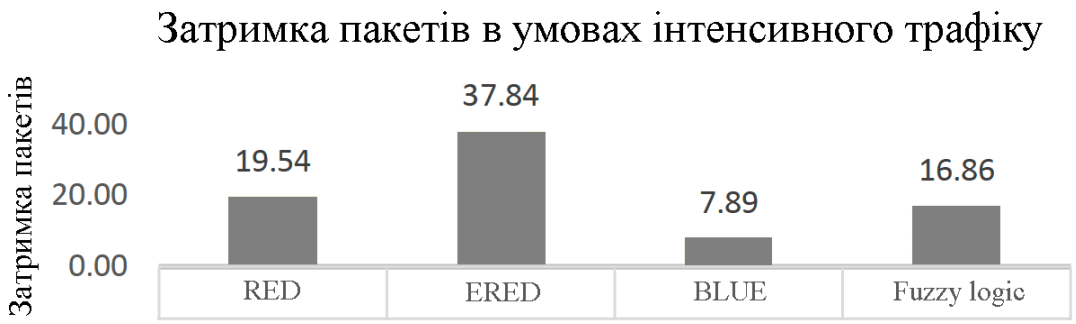

Рисунок 4.11 – Затримка пакетів в умовах інтенсивного трафіку

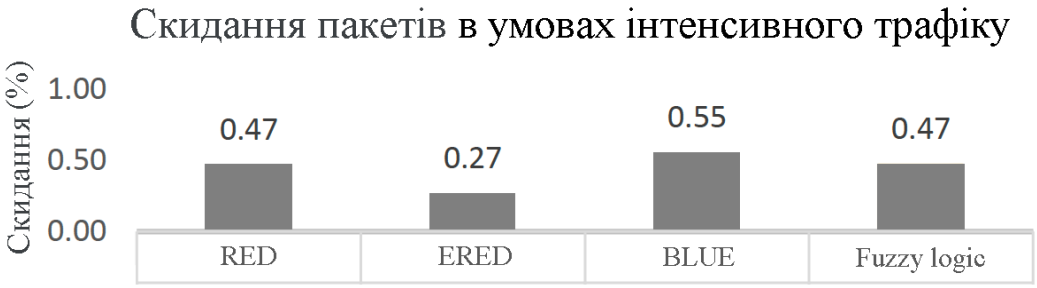

Рисунок 4.12 – Скидання пакетів в умовах інтенсивного трафіку

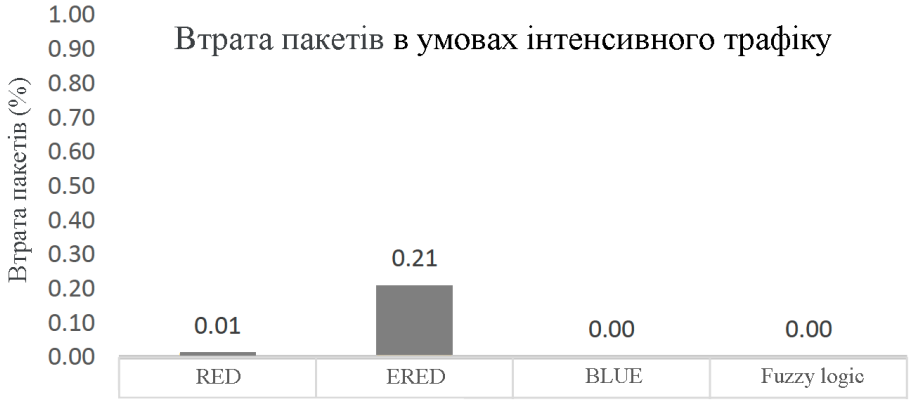

Рисунок 4.13 – Втрата пакетів в умовах інтенсивного трафіку

Відповідно, як зазначено в результатах, запропонований метод зменшує втрату пакетів, відкидає пакети лише тоді, коли це необхідно, щоб уникнути високої швидкості відкидання, і відкидає пакети якомога раніше, щоб уникнути високої затримки.

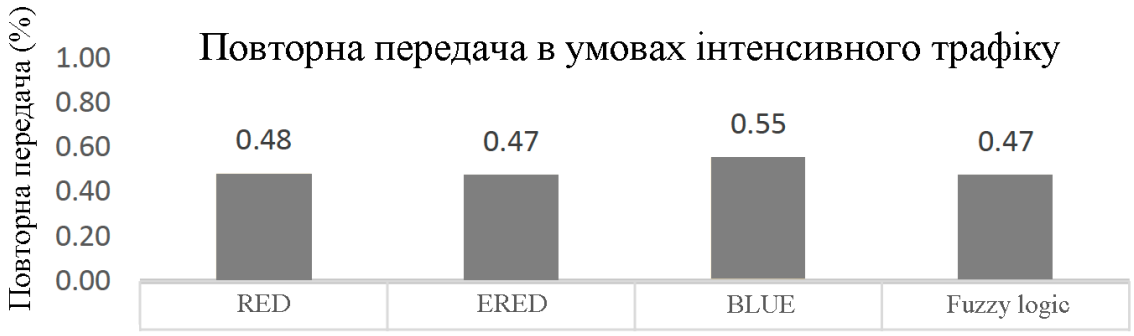

Рисунок 4.14 – Повторна передача в умовах інтенсивного трафіку

Порівняно з іншими методами запропонований метод краще, ніж RED, оскільки він дає кращі результати, прогнозуючи навантаження на ранній стадії та відкидаючи пакети, щоб уникнути затримки. Крім того, запропонований метод краще, ніж ERED, тому що він уникає втрат, відкидаючи пакети тільки тоді, коли це потрібно, на основі індикаторів, що використовуються. Нарешті, запропонований метод краще, ніж BLUE, тому що він уникає високої швидкості відкидання, зменшує втрати, і уникає хибного навантаження, яке неможливо уникнути в BLUE. Високий рівень відкидання вказує на те, що BLUE чутливий до помилкового перевантаження, що призводить до великої кількості втрачених пакетів.

ВИСНОВКИ

Зі збільшенням кількості користувачів і обсягів переданого трафіку спостерігається більше втрат пакетів та інших погіршень продуктивності комп'ютерних мереж через виникнення перевантаження в маршрутизаторах.

Перевантаження на маршрутизаторі виникає, коли буфер, до якого мають бути розміщені передані пакети, переповнений. Як правило, наслідки на маршрутизаторі поширюються на всю мережу. Відповідно, якість послуг і бажана продуктивність, пов'язані зі збільшенням пропускної здатності та зменшення затримок і втрат, вимагають впровадження новітніх методів боротьби з перевантаженнями.

В кваліфікаційній роботі запропоновано метод активного управління чергою маршрутизатора на основі нечіткої логіки для оптимізації продуктивності AQM відповідно до загальновживаних показників мережевої ефективності.

Для досягнення поставлених цілей було реалізовано кілька кроків, починаючи з аналізу показників ефективності та визначення вхідних змінних. Нечітка система реалізована на основі функцій приналежності, повного набору правил і функції дефазифікації. Отримані під час проведення імітаційного моделювання результати показують, що запропонований метод зменшує кількість втрачених пакетів, прогнозуючи навантаження та відкидаючи пакети для уникнення великих затримок. Пропонований метод також виявляє застійні явища на ранній стадії і розпізнає помилкові втрати пакетів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, Recommendations on Queue Management and Congestion Avoidance in the Internet, document RFC 2309, Apr. 1998.

2. S. Floyd and V. Jacobson, ‘‘Random early detection gateways for congestion avoidance,’’ IEEE/ACM Trans. Netw., vol. 1, no. 4, pp. 397–413, Aug. 1993.

3. X. Chen, S. Wong, and C. K. Tse, ‘‘Adding randomness to mod- eling Internet TCP-RED systems with interactive gateways,’’ IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 4, pp. 300–304, Apr. 2010.

4. W. Chen, Y. Li, and S. Yang, ‘‘An average queue weight parameterization in a network supporting TCP flows with RED,’’ in Proc. IEEE Int. Conf. Netw., Sens., Control (ICNSC), Apr. 2007, pp. 590–595.

5. S. Woo and K. Kim, ‘‘Tight upper bound for stability of TCP/RED systems in AQM routers,’’ IEEE Commun. Lett., vol. 14, no. 7, pp. 682–684, Jul. 2010.

6. L. Khoshnevisan, F. R. Salmasi, and V. Shah-Mansouri, ‘‘Robust queue management for TCP-based large round trip time networks with wireless access link,’’ in Proc. IEEE Wireless Commun. Netw. Conf. (WCNC), Mar. 2015, pp. 1309–1313.

7. Choon Chan Mun and Ramjee Ramachandran. Tcp/ip performance over 3g wire- less links with rate and delay variation. Proceedings of the 8th annual interna- tional conference on Mobile computing and networking, Sep 2002.

8. D.E. Lapsley and S.H. Low, Random early marking for Internet congestion control, in: Proc. Of GLOBECOM’99, December 1999, pp. 66–74, http://www.ee.mu.oz.au/staff/slow/.

9. S. Ryu and C. Rump, Design of an adaptive queue management for

supporting TCP congestion con- trol, Journal of Communications and Network (2004) to appear.

10. S. Ryu and C. Rump, Control-theoretic design of a pro-active queue management for Internet conges- tion control, IEICE Transactions on Communications (2003) submitted.

11. M. Christiansen, K. Jaffey, D. Ott and D. Smith, Tuning RED for Web traffic, IEEE/ACM Transactions on Networking 9(3) (2001) 249–264.

12. T.J. Ott, T.V. Lakshman and L. Wong, SRED: Stabilized RED, in: Proc. of INFOCOM'99, March 1999, pp. 1346–1355.

13. W. Feng, K.G. Shin, D.D. Kandlur and D. Saha, The BLUE active queue management algorithms, IEEE/ACM Transactions on Networking 10(4) (2002) 513–528.

14. W. Feng, D.D. Kandlar, D. Saha and K.G. Shin, A self-configuring RED gateway, in: Proc. of INFO- COM'99, March 1999, pp. 1320–1328.

15. S. Kunniyur and R. Srikant, Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management, Technical Report, UIUC (February 2001).

16. J. Aweya, M. Ouellette and D.Y. Montuno, A control theoretic approach to active queue management, Computer Networks 36(2/3) (2001) 203–235.

17. C.V. Hollot, V. Misra, D. Towsley and W. Gong, On designing improved controllers for AQM routers supporting TCP flows, in: Proc. of INFOCOM'2001, April 2001, pp. 1726–1734.

18. Sally Floyd. Recommendation on using the "gentle" variant of RED. http://www.icir.org/floyd/red/gentle.html, March 2000.

19. Kaiyu Zhou, Kwan L. Yeung, and Victor O.K. Li. Nonlinear RED: A simple yet efficient active queue management scheme. Computer Networks, 50:3784–3794, 2006.

20. Wu-chun Feng, Apu Kapadia, and Sunil Thulasidasan. GREEN: pro-active queue management over a best-effort network. In Global Telecommunications Conference, IEEE, volume 2, pages 1774–1778. IEEE, 2002.

21. Abhinav Kamra, Sundeep Kapila, Varun Khurana, Vikas Yadav, Rajeev Shorey, Huzur Saran, and Sandeep Juneja. SFED: a rate control based active queue management discipline. IBM India Research Laboratory Research Report, 2000.

22. Lin, D., R. Morris. Dynamics of Random Early Detection. – In: Proceeding of the ACM SIGCOMM'97 Conference on Applications Technologies, Architectures, and Protocols for Computer Communication, 1997, pp. 127-137.

23. Yaghmaee, M.H., H. Amin Toosi. A Fuzzy Based Active Queue Management Algorithm. – Simulation Series, Vol. 35, 2003, No 4, pp. 458-464.

24. Abualhaj, M.M., A.A. Abu-Shareha, M. M. Al-Tahrawi. FLRED: An Efficient Fuzzy Logic Based Network Congestion Control Method. – Neural Computing and Applications, Vol. 30, 2018, No 3, pp. 925-935.

25. Negnevitsky M., A. Intelligence. A Guide to Intelligent Systems. – In: Artificial Intelligence. 2nd Edition. Pearson Education, 2005.

26. Пилипенко А.О., Чередниченко В.В., Янковський О.А.. Алгоритми AQM для боротьби з перевантаженням//Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Чотирнадцята міжнародна науково-технічна конференція. Баку – Харків – Жиліна. 2024 р. Том 1. С. 114.

27. Вірко А.О., Пилипенко А.О., Янковський О.А. Методи буферизації пакетів в маршрутизаторах IP-мереж//Збірник тез доповідей дванадцятої міжнародної науково-технічної конференції «Проблеми інформатизації». Баку, Харків, Бельсько-Бяла. 2024. С. 76.