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 адаптації протоколу TCP до поточного мережевого стану
(тема)

Виконав:

студент II курсу, групи КСМм-23-1

Козін М.В.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

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

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

Допускається до захисту

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

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 Комп’ютерні системи та мережі
(повна назва)

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

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Козіну Максиму Владиславовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи адаптації протоколу TCP до поточного мережевого стану

затверджена наказом по університету від “ 22 ” листопада 2024 р. № 1237 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) Стек протоколів TCP/IP

3) Методи та алгоритми версій протоколу TCP

4) 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 в протоколі TCP

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) 17 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 перевантаженням	29.11.24–04.12.24	
	в протоколі TCP		
3	Розробка моделі управління мережним	05.12.24 –16.12.24	
	трафіком		
4	Вибір програмних засобів моделювання	17.12.24 –18.12.24	
5	Тестування запропонованого метода	19.12.24–25.12.24	
6	Оформлення пояснювальної записки	26.12.24–-05.01.25	

Дата видачі завдання 25 листопада 2024 р.

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 92 с., 30 рис., 1 табл., 1 дод., 28 джерел.

RTT, TCP, UDP, КАНАЛ ЗВ’ЯЗКУ, ПЕРЕВАНТАЖЕННЯ, ПІДТВЕРДЖЕННЯ, ПОВІДОМЛЕННЯ, ПРОПУСКНА СПРОМОЖНІСТЬ, СКИДАННЯ ПАКЕТІВ, ТАЙМ-АУТ, ТОПОЛОГІЯ, ТРАНСПОРТНИЙ РІВЕНЬ.

Метою кваліфікаційної роботи є вдосконалення механізмів протоколу TCP шляхом застосування різноманітних методів запобігання перевантаженню каналів комп’ютерних мереж.

В кваліфікаційній роботі пропонується метод контролю перевантажень для повільного старту, який зменшує ефект експоненціального зростання ковзного вікна шляхом проектування нових обмежень розміру cwnd у фазі повільного старту, що, у свою чергу, зменшує швидкість втрати пакетів у високошвидкісних мережах. NS-2 використовується для моделювання експериментів із вдосконаленим TCP і найсучаснішими механізмами контролю перевантаження. Результати показують, що продуктивність покращеного TCP перевищує продуктивність сучасних механізмів контролю перевантаження.

ABSTRACT

Master’s thesis: 92 pages, 30 figures, 1 tables, 1 appendices, 28 sources.

ACKNOWLEDGMENT, BANDTHROUGH, COMMUNICATION CHANNEL, MESSAGE, OVERLOAD, PACKET DROP, RTT, TCP, TIMEOUT, TOPOLOGY, TRANSPORT LAYER, UDP.

The purpose of the qualification work is to improve the mechanisms of the TSR protocol by applying various methods of preventing overloading of computer network channels.

The qualification paper proposes a slow-start congestion control method that reduces the exponential growth effect of the sliding window by designing new cwnd size constraints in the slow-start phase, which in turn reduces the packet loss rate in high-speed networks. NS-2 is used for simulation experiments with advanced TCP and state-of-the-art congestion control mechanisms. The results show that the performance of the improved TCP exceeds that of the state-of-the-art congestion control mechanisms.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 8
ВСТУП 9
1 МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ 11
2 КОРОТКИЙ ОГЛЯД СТАНУ ПРОБЛЕМИ 12
2.1 Поява TCP 12
2.2 Багаторівнева структура 13
2.2.1 Еталонна модель OSI 13
2.2.2 Еталонна модель TCP 17
2.2.3 Протокол UDP 20
2.2.4 Протокол TCP 22
2.2.5 Контроль заторів 27
3 ЕВОЛЮЦІЯ ПРОТОКОЛУ TCP 33
3.1 Варіанти протоколу TCP 33
3.2 Різновиди TCP 34
3.2.1 TCP Tahoe 34
3.2.2 TCP Reno 37
3.2.3 TCP Vegas 39
3.2.4 Високошвидкісний TCP (HSTCP) 42
3.2.5 BIC TCP 43
3.2.6 TCP Cubic 44
3.2.7 Compound TCP 46
4 МЕХАНІЗМ КЕРУВАННЯ ПЕРЕВАНТАЖЕННЯМ TCP НА ОСНОВІ ВТРАТ 49
4.1 Необхідність вдосконалення TCP 49
4.2 Фаза повільного старту TCP 53
4.3 Метрики продуктивності 56

4.3.1 Справедливість протоколу 56
4.3.2 ЧАС Сходності 57
4.3.3 Корисна продуктивність 58
4.3.4 Швидкість втрати пакетів 58
4.4 Розробка методу 58
4.4.1 Метод M-SS 59
4.4.2 Тестування і оцінка продуктивності 60
4.4.3 Налаштування імітаційного моделювання 60
4.5 Розробка методу M-SS 65
4.6 Оцінка пропускної здатності та затримки 67
4.7 Результати моделювання 74
ВИСНОВКИ 79
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 80
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 83

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

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

AIMD – адитивне збільшення/мультиплікативне зменшення (англ., Additive increase multiplicative decrease)

CWA – дія вікна перевантаження (англ., Congestion window action)

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

DACK – підтвердження повторного TCP-пакету (англ., Duplicate TCP packet acknowledgment)

DUPACK – дублікат підтвердження (англ., Duplicate Acknowledgement)

FTP – протокол передачі файлів (англ., File Transfer Protocol)

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

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

PACK – часткове підтвердження (англ., Partial acknowledgment)

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

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

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

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

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

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

ВСТУП

Сьогодні більшість трафіку в Інтернеті передається за протоколом керування передачею (TCP). TCP – це віконний надійний протокол передачі даних, що забезпечує передачу даних між двома кінцевими хостами з'єднання. Оригінальний TCP офіційно предствлено в [1]. Він має простий механізм керування потоком ковзного вікна. Основною стратегією TCP є надсилання пакетів у мережу без резервування та подальша реакція на спостережувані події.

Протягом багатьох років мережевий трафік зростав, і Інтернет-додатки еволюціонували від стандартної функції пошуку документів до мультимедійних послуг. Швидке зростання Інтернету та збільшення попиту на трафік призвели до серйозної проблеми під назвою колапс перевантаження [2].

Перевантаження в Інтернеті виникає, коли сукупний попит на ресурси перевищує доступну пропускну здатність мережі. Ця проблема призводить до неприйнятно тривалого часу відгуку, особливо для програм реального часу. Коли пакет стикається з перевантаженням, існує велика ймовірність того, що пакет буде відкинуто, а відкинутий пакет витратить дорогоцінну смугу пропускання мережі на шляху від свого відправника до точки перевантаження. Контроль перевантаження, таким чином, необхідний для запобігання колапсу перевантаження в мережі. Без контролю перевантаження вузол-відправник може бути зайнятий передачею пакетів, які пізніше можуть бути відкинуті. Таким чином, одним із ключів до успіху Інтернету є використання механізмів контролю перевантаження.

Найбільш суттєвим є механізм контролю перевантаження, запропонований Якобсоном у 1988 році [3]. Цей механізм називається TCP Tahoe. Він включає три алгоритми; а саме; повільний старт, уникнення перевантаження та швидка повторна передача. У 1990 році була розроблена

нова версія TCP під назвою TCP Reno, додавши до Tahoe алгоритм швидкого відновлення [4]. TCP Reno можна розглядати як реактивну схему контролю перевантаження, яка використовує втрату пакетів як індикатор перевантаження.

Щоб перевірити доступну пропускну здатність уздовж наскрізного шляху, вікно перевантаження TCP збільшується доти, доки не буде виявлено втрату пакета, після чого вікно перевантаження зменшується вдвічі, і виконується алгоритм лінійного збільшення, доки не буде виявлено подальшу втрату пакета.

TCP Reno може періодично генерувати втрату пакетів сам по собі і не може ефективно відновити багато втрат пакетів з вікна даних. Крім того, стратегія адитивного збільшення мультиплікативного зменшення (AIMD) TCP Reno призводить до періодичних коливань в аспектах розміру вікна перевантаження (cwnd), затримки в обидві сторони та довжини черги вузького місця. Коливання можуть викликати хаотичну поведінку в мережі, тим самим негативно впливаючи на загальну продуктивність мережі.

1 МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

Протокол керування передачею (TCP) відповідає за надійну та безпечну передачу даних у цих мережах середньої інфраструктури. TCP є надійним і безпечним завдяки механізму контролю перевантажень, який відповідає за виявлення перевантажень у мережі та реагування на них.

Експоненціальне зростання вікна перевантаження у фазі повільного старту протоколу TCP спричиняє втрати пакетів, і потоки TCP не розподіляють доступну пропускну здатність мережевого каналу справедливо. Основна мета кваліфікаційної роботи полягає в тому, щоб підвищити продуктивність протоколу TCP для високошвидкісних мереж з метою досягнення кращої продуктивності.

Таким чином, в ході кваліфікаційної роботи необхідно виконати наступні завдання:

- провести огляд сучасних літературних джерел, пов’язаних з проблемами запобігання перевантаженню в комп’ютерних мережах;
- розглянути існуючі моделі та методі управління інформаційними потоками;
- представити метод адаптації протоколу TCP до поточного мережевого стану;
- провести імітаційне моделювання для підтвердження запропонованих в кваліфікаційній роботі теоретичних викладок;
- провести аналіз отриманих результатів.

2 КОРОТКИЙ ОГЛЯД СТАНУ ПРОБЛЕМИ

2.1 Поява TCP

Наскрізна доставка даних була революціонізована Серфом і Каном у 1974 році у їхній епохальній статті, де детально описано протокол TCP і викладено концепції, пов’язані з інкапсуляцією, дейтаграмами та функціями шлюзу. Завдяки переданню відповідальності за виправлення помилок під час зв’язку від процесора інтерфейсних повідомлень (IMP) до головної машини мережа агентства передових дослідницьких проектів (ARPANET) стала центром усіх комунікаційних розробок. Потім у 1977 році була успішно продемонстрована мережа, що складається з ARPANET, пакетного радіозв’язку та пакетного супутникового зв’язку, що поклало початок новій епосі сучасного зв’язку.

Після цього протокол TCP/IP було розділено на два окремі протоколи: TCP і IP. Завдяки цьому маршрутизація розділених дейтаграм оброблялася IP, тоді як TCP виконував такі завдання, як сегментація, повторна збірка та виявлення помилок.

Новим міжмережевим протоколом став TCP/IP, який пізніше був включений у комп’ютерні мережеві системи, такі як UNIX. До 1981 року протокол TCP/IP був формалізований, і його включення як мережевого програмного забезпечення в комп’ютерні операційні системи дало початок передачі даних на настільних комп’ютерах. Зрештою оригінальний протокол ARPANET було скасовано, і TCP/IP став стандартом для різних мереж та Інтернету.

Протокол використовується та постійно розвивається більше трьох десятиліть. Те, що починалося як експериментальна комунікаційна технологія, тепер стало основою найбільш використовуваної та складної мережі у світі.

2.2 Багаторівнева структура

Мережі та Інтернет є надзвичайно складними системами, оскільки багато компонентів працюють разом, щоб забезпечити платформу для наскрізного зв’язку. Програми, програмне забезпечення, протоколи, операційні системи, комутатори, маршрутизатори, носії на рівні зв’язку тощо мають працювати разом, щоб бездоганно забезпечувати з’єднання.

Враховуючи таку величезну складність, мережева архітектура була організована пошарово. Це узгоджується з основними принципами та філософією проектування архітектури Інтернету, яка коротко стверджує, що складність потрібно контролювати, якщо хтось сподівається ефективно масштабувати складну проблему. Таким чином, функції комунікаційних систем були розділені на рівні абстракції, що призвело до стандартизації передачі даних. Функціональні можливості кожного рівня залежать від послуг, які пропонує нижчий рівень.

2.2.1 Еталонна модель OSI

Існують дві переважаючі моделі, які використовуються як основа багаторівневої структури комунікації. Міжнародна організація зі стандартизації (ISO), заснована в 1947 році і відповідальна за угоди щодо міжнародних стандартів, запровадила модель взаємозв’язку відкритих систем (OSI) наприкінці 1970-х років [5].

OSI дозволяє зрозуміти та спроектувати архітектуру мережі, яка є гнучкою, надійною та сумісною, дозволяючи кінцевим системам спілкуватися незалежно від їх базової архітектури.

Друга модель, відома як еталонна модель TCP, була розроблена до моделі OSI. Модель TCP має інтерактивні модулі, кожен з яких забезпечує певну функціональність у спілкуванні, хоча кожен модуль не обов’язково взаємозалежний. Однак у моделі OSI функції, що належать до певних рівнів,

залежать від роботи інших рівнів, тоді як у моделі TCP, оскільки вона використовує відносно незалежні протоколи як різні рівні, модулі можна змішувати та підбирати відповідно до потреб системи. Тим не менш, як моделі OSI, так і моделі TCP забезпечують механізми, за допомогою яких складні гетерогенні середовища можуть бути розумно сегментовані.

Рисунок 2.1 – Еталонна модель OSI

Фізичний рівень (рівень 1) головним чином відповідає за координацію передачі бітових потоків через фізичне середовище між кінцевими вузлами. Він визначає правила, яким дотримуються середовища передачі стосовно переміщення бітів даних (нулів і одиниць) у ланцюзі.

Протоколи на цьому рівні залежать від каналу зв’язку, тобто він покладається на фактичні деталі середовища передачі (тобто волоконно-оптична оптика, мідний провід з витою парою тощо).

На цьому рівні виконуються наступні функції:

- визначаються характеристики інтерфейсів між пристроями та середовищем передачі;
- біти даних кодуються в електричні або оптичні сигнали;
- визначається швидкість передачі;

- між пристроями встановлюється з’єднання або точка-точка, або багатоточка;

- визначається напрямок передачі між пристроями (симплекс, напівдуплекс, повний дуплекс).

Основним завданням канального рівня є забезпечення того, щоб необроблені дані, отримані на фізичному рівні, перетворювалися в схему, яка виглядає безпомилковою для верхніх рівнів моделі OSI.

Це робиться шляхом маскування справжніх помилок і досягається розбиттям даних відправника на кадри даних із подальшою передачею кадрів послідовно.

Цей рівень також додає заголовок до кадру, щоб визначити фізичні адреси вузлів відправника та приймача. Основними обов'язками канального рівня є:

- розподіл потоку даних на кадри керованого розміру;

- фізична адресація кадрів для включення адрес відправника та одержувача;

- встановлення механізму контролю потоку, який запобігатиме перевантаженню приймача;

- підвищення надійності передачі даних шляхом впровадження механізму виявлення та повторної передачі втрачених або пошкоджених даних; і керування доступом до каналів, де два чи більше пристроїв підключено до одного каналу.

Мережевий рівень відповідає за переміщення пакетів мережевого рівня від джерела до пункту призначення, відомих як дейтаграми, можливо, через різні мережі. Канальний рівень відповідає за транспортування всередині мережі, тоді як мережевий рівень гарантує, що кожен пакет досягне свого кінцевого пункту призначення, навіть якщо він проходить через кілька мереж. Таким чином, на цьому рівні до пакетів, що надходять з верхніх рівнів, додається заголовок, що містить логічні адреси відправника/одержувача.

Завершивши це, дейтаграма починає процес маршрутизації до пункту призначення шляхом визначення наступного вузла мережі, до якого має бути надіслано повідомлення, щоб знайти найкращий можливий маршрут (залежно від реалізованого протоколу маршрутизації).

Мережевий рівень містить протокол IP і численні протоколи маршрутизації, однак цей рівень просто називають рівнем IP.

Транспортний рівень в основному займається питаннями наскрізного з'єднання, такими як визначення процедур, яких необхідно дотримуватися для введення та виведення даних у різні мережі. Попередній мережевий рівень відповідає за наскрізну доставку окремих пакетів без усвідомлення зв'язку між пакетами, а транспортний рівень, з іншого боку, гарантує, що все повідомлення прибуде до місця призначення неушкодженим і в правильному вигляді та послідовності. Тобто він відповідає як за контроль помилок, так і за контроль потоку всього процесу транспортування.

Цей рівень:

- додає заголовок з адресою точки обслуговування (адресою порту), що дозволяє мережевому вузлу передати все повідомлення до правильного процесу на комп'ютері призначення;
- сегментує повідомлення на сегменти, які можна передавати, і додає порядковий номер (пізніше використовує цю інформацію для повторного складання повідомлення);
- підтримує механізм контролю з'єднання;
- підтримує наскрізний контроль потоку;
- забезпечує доставку без помилок (пошкодження, втрата або дублювання) за допомогою повторних передач, коли це необхідно.

Рівень сеансу є контролером мережевого діалогу, оскільки він відповідає за ініціювання, підтримку та припинення взаємодії між комунікуючими системами. Управління та структурування сеансів, під час яких учасники сеансу виконують такі дії, як вхід в обладнання, передача файлів і виконання перевірок безпеки, все виконується на цьому рівні.

Наприкінці процесу зв’язку сеанс також припиняється за допомогою механізмів на цьому рівні, однак, якщо відбувається передчасне завершення сеансу, вбудовані резервні засоби можуть відновити сеанс. Таким чином, сеансовий рівень дозволяє двом системам вступати в діалог.

Рівень презентації в основному відповідає за переміщення бітів по комунікаційній інфраструктурі – рівень представлення пов'язаний із синтаксисом і семантикою інформації, якою обмінюються пристрої зв'язку. Він форматує дані для представлення користувачеві, щоб різні інтерфейси на різних комунікаційних пристроях і програми на комп’ютерах не хвилювалися про формати даних, тобто він займається відображенням, форматуванням і редагуванням введених і виведених даних користувача.

Спеціальні обов’язки рівня представлення включають:

- переклад даних таким чином, що залежний від відправника формат даних змінюється на загальний формат, який на кінці прийому змінюється на формат, що залежить від адресата;
- шифрування даних, щоб конфіденційна інформація могла передаватись між системами;
- стиснення даних, коли кількість бітів, що містяться в інформації, зменшується.

Прикладний рівень – це найвищий рівень (рівень 7) еталонної моделі OSI. На цьому рівні представлені різноманітні протоколи, які зазвичай використовуються кінцевими користувачами, полегшуючи доступ до мережі через такі служби, як електронна пошта, передача файлів, доступ до бази даних тощо.

2.2.2 Еталонна модель TCP

Модель OSI використовувалася як базовий механізм для пояснення концепцій мережі протягом тривалого часу; однак в основі моделі OSI лежить еталонна модель, яка використовується в проекті ARPANET.

Канальний рівень в основному пов'язаний зі зв'язком між хостами та лініями передачі. Ранні матеріали, що стосуються цієї моделі, містили мало інформації, пов’язаної з цим рівнем, однак вимоги до цього рівня призвели до вибору технології мережі з комутацією пакетів, заснованої на концепціях рівня без з’єднання, що працює в кількох мережах.

Таким чином, цей рівень описує, що повинні виконувати зв’язки, такі як послідовні лінії або Ethernet, щоб відповідати вимогам наступного рівня, Інтернет-рівня.

У еталонній моделі TCP/IP рівень Інтернету близько відповідає мережевому рівню в моделі OSI. Цей рівень має вирішальне значення для архітектури мережі, оскільки він дозволяє мережевим хостам ініціювати зв’язок, вводячи пакети в будь-яку мережу, полегшуючи їх подорож до пункту призначення, можливо, різними мережевими маршрутами. Прибувши до пункту призначення, вищі рівні в цій архітектурі перегруповують та зберають пакети. Тобто рівень Інтернету відповідає за доставку IP-пакетів у будь-яке місце, куди вони призначені.

Транспортний рівень, як і в моделі OSI, розроблений для того, щоб дозволити вузлам джерела та вузла призначення продовжувати з’єднання. Для цього на цьому рівні визначено два протоколи: TCP і UDP.

TCP – це протокол, орієнтований на з’єднання, який забезпечує безпомилкову передачу даних, тоді як UDP – це протокол без з’єднання, який може використовуватися програмами, які не вимагають перевірок і противаг, які зазвичай зустрічаються в з’єднаннях типу TCP.

Прикладний рівень у моделі TCP/IP є найвищим рівнем, і крім того, що він містить протоколи вищого рівня, необхідні користувачеві мережі, він також включає функції сеансу та презентації, які існують на різних рівнях у Модель OSI.

Варто об’єднати функції всіх трьох рівнів в один, оскільки сеансовий і презентаційний рівні зазвичай мало корисні для більшості програм. На прикладному рівні є численні протоколи, які зазвичай використовуються в

мережах, зокрема протокол терміналів (TELNET), протокол передачі файлів (FTP), простий протокол передачі пошти (SMTP), протокол передачі гіпертексту (HTTP), DNS тощо.

Функціональні можливості на рівнях еталонних моделей OSI і TCP, як показано на рисунку 2.2, приблизно однакові. Рівні, аж до транспортного, надають незалежні від кінця до мережі транспортні послуги для зв'язку, тоді як рівні вище транспортного рівня орієнтовані на застосування для користувачів. Однак між двома моделями є деякі ключові відмінності.

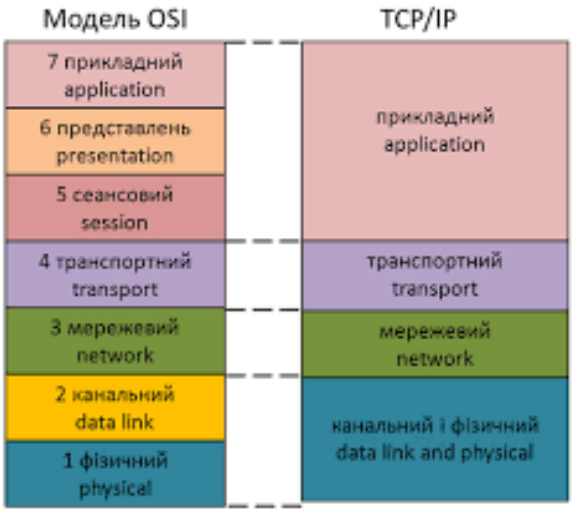

Рисунок 2.2 – Порівняння моделі ISO/OSI та архітектури TCP/IP

По-перше, модель OSI робить дуже чітке розмежування між послугами, інтерфейсами та протоколами, на відміну від моделі TCP. Тут сервіси визначають семантику рівня, інтерфейси пов'язані з інформуванням процесів над рівнем про те, які параметри стосуються взаємодії між рівнями, а протоколи пов'язані з тим, як рівень виконує інструкції в межах рівня для досягнення бажаного. цілі. Розділивши ці три завдання, OSI може добре вписуватися в сучасні концепції об'єктно-орієнтованого програмування. Однак у моделі TCP ці межі між службами, інтерфейсами та протоколами розмиті. По-друге, модель OSI є загальною і не упереджена до будь-якого конкретного маршрутизованого протоколу, такого як TCP/IP.

Це в основному через те, що сама модель була окреслена задовго до розробки будь-якого конкретного протоколу. Це було не так для моделі TCP, модель була першою, а потім прийняті протоколи. З точки зору дизайну, такі протоколи, як TCP/IP, ідеально вписуються в еталонну модель TCP.

По-третє, у відношенні зв'язку без з'єднання та зв'язку, орієнтованого на з'єднання, модель OSI підтримує обидва на мережевому рівні, але на транспортному рівні підтримує лише зв'язок, орієнтований на з'єднання.

Проте модель TCP підтримує протокол без встановлення з'єднання лише на мережевому рівні, але обидва на транспортному рівні – це важливо, оскільки служби транспортного рівня видимі для кінцевого користувача, тому це дає користувачам моделі TCP вибір.

Нарешті, відсутність відмінності між фізичним і канальним рівнем у моделі TCP викликає занепокоєння. Комунікаційні характеристики фізичного рівня мідних проводів, волоконно-оптичної мережі або безпровідного зв'язку та канального рівня, що стосуються початкових/кінцевих кадрів і бажаних ступенів надійності, є різними наборами характеристик, які розрізняються лише в моделі OSI.

2.2.3 Протокол UDP

Інтернет-модель має як TCP, так і UDP на своєму транспортному рівні, і з двох протоколів UDP є простішим протоколом. Це ненадійний протокол без встановлення з'єднання, який забезпечує механізм для додатків надсилати інкапсульовані IP-дейтаграми без необхідності встановлення з'єднання між комунікуючими вузлами.

UDP не має механізму керування потоком і не має функцій, пов'язаних із підтвердженням отриманих пакетів.

Перевірка помилок не є основною функцією: якщо вона виявляє помилку, пакет скидається мовчки. UDP можна описати як простий транспортний протокол без надмірностей.

Протокол транспортного рівня повинен забезпечувати механізм для з'єднання процесів між комунікуючими вузлами. Процеси з'єднання передбачають встановлення з'єднання між відправником і одержувачем, сегментацію потоку пакетів на транспортабельні одиниці, їх нумерацію, а потім відправку одну за одною від відправника до одержувача. На приймальному кінці транспортний рівень чекає, поки не прибудуть усі різні одиниці, загальні для процесу, перевіряє їх, а потім передає безпомилкові сутності процесу як потік.

Однак UDP не виконує жодної зі згаданих тут дій – він просто отримує потік даних для процесу та доставляє його ненадійно. Отже, UDP є найпростішим із транспортних протоколів, який забезпечує зв'язок між процесами замість повного процесу між хостами.

Незважаючи на недоліки UDP, це широко використовуваний протокол зв'язку завдяки деяким його невід'ємним перевагам. Це простий протокол, який має лише 8 байт накладних витрат на сегмент. Якщо відправнику потрібно надіслати інформацію, і надійність не є проблемою, UDP є протоколом вибору. UDP негайно пакує дані в сегмент UDP і швидко передає їх на мережевий рівень. Крім того, взаємодія між відправником і одержувачем зведена до мінімуму, тобто на початку процесу немає попередніх дій. UDP не встановлює та не підтримує стан з'єднання, який включає такі сутності, як буфери прийому та надсилання, механізми контролю перевантаження, а також параметри підтвердження та порядкового номера.

Структура сегмента UDP спочатку була визначена в RFC 768. Вона має заголовок із 8 байтів фіксованого розміру, за яким слідує корисне навантаження.

Як показано на рисунку 2.3, вузли джерела та призначення ідентифікуються портами, і коли пакет надходить, корисне навантаження передається процесу, приєднаному до порту призначення. Потім порт доставляє вбудований сегмент у правильну програму. Поле порту джерела

копіюється до вхідного сегмента, щоб через нього можна було надіслати відповідь до потрібного джерела зв'язку. Поле довжини UDP може мати мінімум 8 байт (розмір заголовка) або до 65515 байт. Поле контрольної суми заголовка – це необов'язкове поле, яке можна використовувати для додання додаткової надійності інформації заголовка або самим даним.

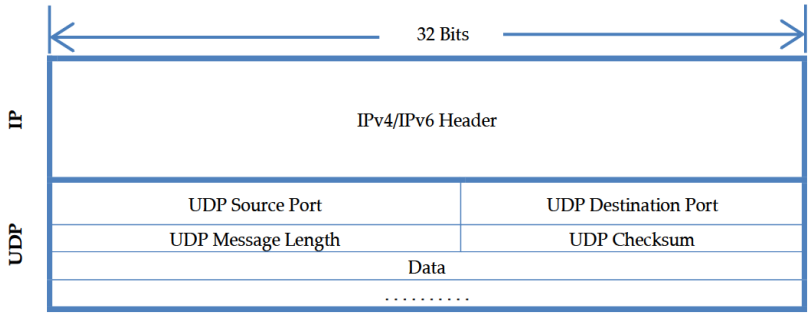

Рисунок 2.3 – Структура UDP пакету

Протокол UDP використовується в ситуаціях зв'язку, коли засоби виправлення помилок не потрібні, або коли існує потреба в одному короткому обміні повідомленнями між програмами. Однак у більшості каналів зв'язку гарантується надійна передача даних, де недопустима втрата навіть одного біта.

2.2.4 Протокол TCP

Протокол TCP, на відміну від UDP, є протоколом, орієнтованим на підключення, який надійно передає дані між джерелом і одержувачем, перетинаючи ненадійні міжмережі. У 1981 році Міністерство оборони США стандартизувало протокол, до цього було дев'ять попередніх доповнень ARPA TCP. Важливість протоколу TCP як основи для сучасної комунікаційної інфраструктури підкреслюється кількістю наступних RFC, написаних після RFC793, кожен з яких пропонує вдосконалення та виправлення помилок і невідповідностей.

Однією з його ключових особливостей є конструкція, яка оптимізована для точної та надійної доставки, а не для швидкої та своєчасної доставки. Отже, передача даних TCP може піддаватися відносно тривалим затримкам передачі через виправлення, пов’язані з помилками передачі. Таким чином, TCP не підходить для додатків передачі даних у реальному часі, таких як Voice-over IP (VoIP).

Крім того, усі з’єднання TCP є повнодуплексними, тобто дані передаються в обох напрямках під час процесу зв’язку. TCP також є протоколом «точка-точка», що означає, що в процесі зв’язку завжди є лише дві кінцеві точки (з цієї причини протокол не підходить для програм, які використовують механізми багатоадресної та широкомовної передачі). Використання сокетів відіграє важливу роль під час спілкування за допомогою протоколу TCP. Сокети, які є кінцевими точками на вузлах відправлення та приймання, отримують номери сокетів, які можуть використовуватися декількома програмами для цілей підключення.

Під час встановлення сокетів для кожної точки кінцевого вузла створюються номери сокетів, які складаються з IP-адреси та 16-бітного номера локального порту. Контроль потоку та перевантаження також поширений у передачі даних TCP. Тут швидкість передавання керується одержувачем, щоб забезпечити надійне отримання даних відправником (це робиться шляхом постійного натяка одержувача відправникові, скільки даних він може отримати за певний період).

При передачі за протоколом TCP кожен байт у процесі зв'язку має свій порядковий номер. Ці 32-розрядні номери передаються разом із пакетами в обох напрямках передачі та використовуються для забезпечення того, щоб усі дані в транзиті надходили до місця призначення без змін (інакше стартаються процедури повторної передачі).

Коли йдеться про передачу даних між відправником і одержувачем, використовуються сегменти TCP. Вони складаються з фіксованого 20-байтового заголовка, за яким зазвичай слідують деякі додаткові параметри.

Як правило, розмір усього сегмента буде залежати від обсягу даних, який він може нести, який обмежується двома параметрами: кожен сегмент має вміщатися в 65535 байт корисного навантаження (включно з заголовком) і повинен відповідати розміру максимальної одиниці передачі (MTU). Пакет TCP, як показано на рисунку 2.4, має заголовок IP і заголовок TCP.

У заголовку TCP перші два поля є адресами портів вузлів джерела та призначення, подібно до структури пакетів UDP. Ці дві адреси є кінцевими точками каналу зв'язку, який необхідно встановити. Далі йде послідовність і номери підтвердження, де перше пов’язане з напрямком передачі даних, а друге використовується у зворотному напрямку. Обидва ці числа стосуються позиції фактичного октету в повному потоці повідомлень, ідентифікуючи або позицію першого октету відносно початку потоку даних, або у зворотному потоці даних. Далі, поле параметрів – це поле змінної довжини, яке може вміщувати заголовки, якщо вони мають змінну довжину.

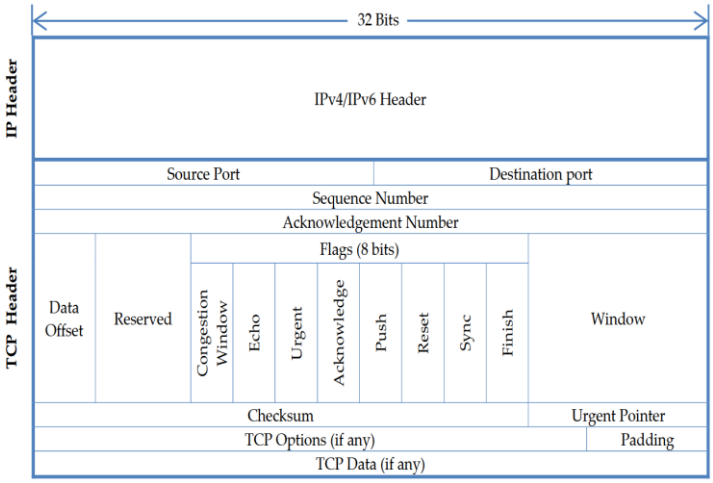

Рисунок 2.4 – Структура пакету TCP

Далі йдуть вісім 1-бітових прапорців. Вікно перевантаження – це прапорець, який встановлює відправник, щоб вказати, що він отримав сегмент TCP і відповів механізмом контролю перевантаження, тоді як прапор відлуння відіграє подвійну роль, залежно від значення прапора SYN. Обидва

ці прапорці використовують явне сповіщення про перевантаження (ECN) у процесі контролю перевантаження в мережі. Значення термінового поля визначає, чи використовується терміновий покажчик, а поле підтвердження вказує, чи це поле є значущим. Дані, передані до програми, активують поле push, вказуючи, що після надходження нові дані не потребують буферизації, а можуть надходити безпосередньо до програми. Після цього прапор скидання може бути активований, якщо з'єднання між відправником і одержувачем потрібно повторно активувати, допомагаючи біту SYN відновити з'єднання. Кінцевий прапорець (finish) використовується для вказівки на те, що більше даних передавати не потрібно.

У з'єднаннях TCP об'єм даних, який можна надіслати до отримання підтвердження, контролюється за допомогою ковзного вікна змінного розміру. Поле розміру вікна керує цим за допомогою параметра масштабу вікна, щоб дозволити відправникам і одержувачам узгодити розмір вікна. Далі поле контрольної суми додає додаткову надійність як заголовку, так і даним, тоді як необов'язкове поле можна використовувати для надання додаткових можливостей, якщо цього вимагає звичайний заголовок. У кожному надісланому пакеті є позначка часу, яка повторюється одержувачем. Опція вибіркового підтвердження (SACK) дозволяє одержувачу повідомляти відправнику про діапазон порядкових номерів, які він. SACK дозволяє відправнику визначити, які дані вже має одержувач, і що потрібно повторно передати [6]. Фактична передача даних відбувається лише після узгодження параметрів сеансу.

На початку процесу передачі даних TCP виконується процедура тристороннього обміну повідомленнями, відома як тристороннє рукостискання. Як показано на рисунку 2.5, метою цієї процедури є синхронізація кінців з'єднання шляхом узгодження параметрів, які будуть використовуватися під час сеансу TCP. Механізм рукостискання також гарантує, що обидві сторони готові до процесу передачі, і що всі передачі будуть лише після успішного встановлення сеансу.

Фактичний сеанс встановлюється, коли вузол-відправник надсилає сегмент із прапорцем SYN разом із запропонованою першою послідовністю у відповідному полі заголовка TCP (syn=X). Отримавши цю інформацію, одержувач бере до уваги те, що було надіслано, а потім повертає сегмент із встановленими прапорцями SYN і ACK із власними значеннями для зворотного напрямку як для порядкового номера (syn=Y), так і поле підтвердження. Це надсилається разом із syn=X+1 і Ack=X+1 для підтвердження отримання початкових значень із вхідного напрямку.

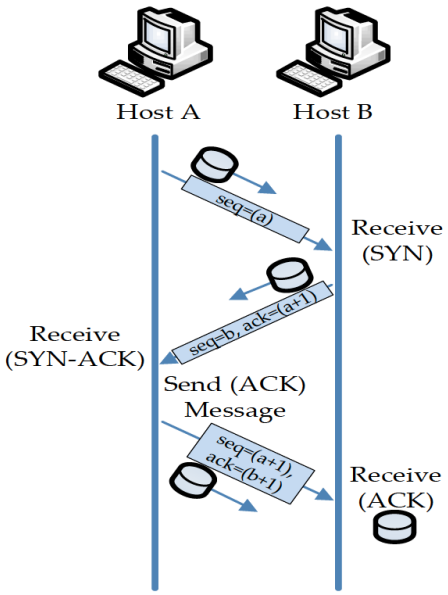

Рисунок 2.5 – Встановлеення TCP-з’єднання

Коли відправник отримує зворотну інформацію, він бере до відома значення Y і надсилає назад підтвердження зі значенням Y+1. Після успішного виконання тристороннього рукостискання обидві сторони в каналі зв’язку налаштовані та можуть почати незалежно надсилати дані одна одній.

Наприкінці процесу зв’язку сеанс між відправником і одержувачем має бути припинено. Це може бути ініційоване будь-яким із вузлів, що спілкуються, шляхом надсилання сегмента TCP із бітом FIN, що вказує на те, що більше немає даних для передачі (рисунок 2.6).

Отримавши біт FIN, інший кінець вимикає канал зв’язку в одному напрямку для будь-яких нових даних. Однак дані можуть продовжувати надходити у протилежному напрямку. Цей зворотний напрямок також можна вимкнути подібним чином, коли FIN-пакет надсилається з іншого вузла до його відповідника. Таким чином, для завершення сеансу необхідно надіслати чотири сегменти TCP.

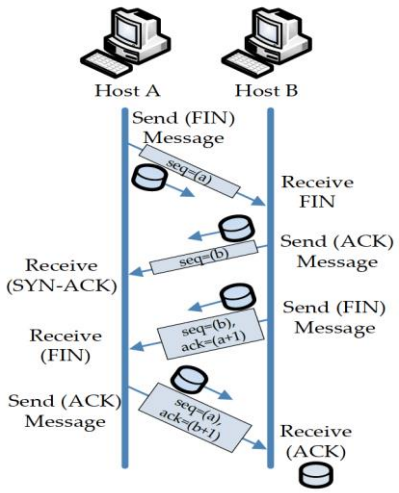

Рисунок 2.6 – Розрив TCP-з’єднання

2.2.5 Контроль заторів

Однією з поширених особливостей протоколу TCP є його здатність контролювати перевантаження. Під час передачі даних, коли даних більше, ніж може обробляти мережа, накопичується перевантаження, створюючи мережеві черги, що призводить до погіршення продуктивності та втрати пакетів [7]. По суті, це пов’язано з тим, що вихідний вузол беззастережно надсилає пакети в мережу, в результаті чого одержувач реагує на подію. На цьому етапі мережевий рівень інформує транспортний рівень про ситуацію, який реагує, знижуючи швидкість передачі. Роль, яку TCP відіграє під час передачі даних, є критично важливою, оскільки це ключовий протокол для контролю перевантажень у переповнених мережах.

Щоб контролювати перевантаження в мережі, було розроблено низку алгоритмів і процедур, деякі з яких вбудовані в TCP та інші пов’язані протоколи.

RFC 5681 описує чотири взаємопов’язані алгоритми TCP: повільний старт, уникнення перевантаження, швидка повторна передача та швидке відновлення.

Коли ці чотири алгоритми використовуються разом, вони відомі разом як алгоритм адитивного збільшення/мультиплікативного зменшення (AMID). Далі представлені деталі чотирьох ключових алгоритмів, які використовуються для контролю перевантажень мережі.

Алгоритм повільного старту використовується для контролю обсягу перевантажених даних, які надсилаються в мережу. Він був розроблений насамперед для контролю агресивної поведінки TCP при старту, коли відправник швидко переповнював канал зв’язку до максимуму, що призводило до втрати даних і переповнення буфера, в результаті чого була низька продуктивність мережі.

Під час першого циклу відправлення лише один пакет вставляється в мережу відправником, а на пункті призначення підтверджується його отримання, два пакети надсилаються в наступному раунді, а потім чотири, кожен після підтвердження від приймача.

Повільний старт працює добре, незалежно від пропускної здатності каналу зв’язку та пов’язаного часу проходження туди й назад [8]. Це можливо, оскільки його механізм використовує ACK, щоб узгодити швидкість передачі відправника зі швидкістю зв’язку. Алгоритми повільного старту можуть досить агресивно передавати дані, що негативно впливає на продуктивність мережі. Ця поведінка контролюється за допомогою порогового параметра ssthresh, який працює разом із розміром іншої змінної, відомої як cwnd. Під час передачі, коли значення cwnd перевищує ssthresh, вузол-відправник стартає алгоритм уникнення перевантаження (рисунок 2.7).

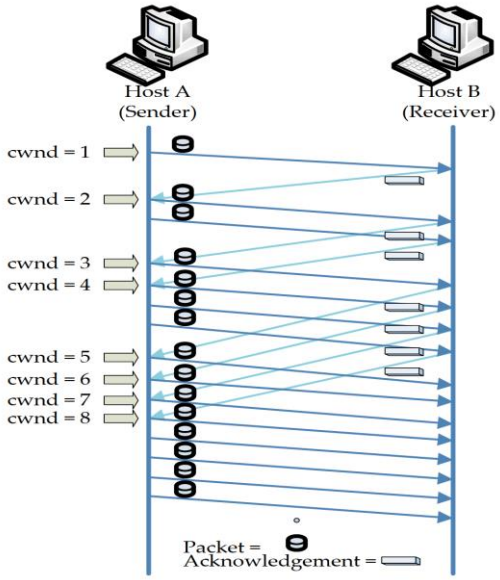

Рисунок 2.7 – Робота алгоритму повільного старту

Алгоритм уникнення перевантажень є основним механізмом контролю над перевантаженнями в мережі. Алгоритм підтримує стабільний стан передачі даних, вводячи нові потоки даних у мережу зі швидкістю, еквівалентною швидкості отримання ACK [8].

Виходячи з передумови, що будь-який канал зв’язку може прийняти трохи більше даних, алгоритм уникнення перевантажень використовує будь-яку додаткову смугу пропускання на шляху для підтримки стабільного потоку даних.

Коли алгоритм працює, відправник може динамічно адаптуватися до будь-яких випадкових змін у стані мережевого шляху. Під час уникнення перевантажень значення cwnd збільшується на один сегмент за час проходження в обидві сторони після отримання кожного нового ACK для надісланих даних.

Цей інкрементальний метод дозволяє відправнику обережно шукати будь-яку доступну смугу пропускання, яку можна використати, залишаючись справедливим до будь-яких інших сеансів TCP, які використовуються тим самим мережним каналом.

Якщо сегмент даних втрачається під час сеансу передачі TCP-відправника, цей сегмент потрібно передати повторно. Швидка повторна передача – це спеціальний механізм, який керує цим і намагається скоротити час, який відправник повинен чекати, перш ніж він зможе повторно передати втрачений сегмент.

Основою, на якій швидка повторна передача вирішує, на скільки затримати передачу перед повторним надсиланням втрачених сегментів, є повторне підтвердження (DUPACK). Отримавши три DUPACK від одержувача (із зазначенням втрачених сегментів), відправник негайно повторно передає те, що виглядає як відсутні сегменти, не чекаючи періоду очікування.

Кожного разу, коли викликається алгоритм швидкої повторної передачі, надсилається один сегмент даних, і відразу після виконання повторної передачі відповідальність за подальшу передачу передається алгоритму швидкого відновлення. У ситуації, коли в мережі спостерігається помірне перевантаження, алгоритм швидкого відновлення підвищує швидкість передачі даних, особливо для великих вікон.

Алгоритм зменшує вікно перевантаження до 1 кожного разу, коли виявляється перевантаження мережі, і полегшує проблему, уникаючи фази повільного старту. Це відбувається, коли відправник TCP отримує DUPACK, який інформує про втрату додаткових пакетів і вказує на те, що дані все ще передаються між взаємодіючими вузлами, і що немає необхідності різко зменшувати потік за допомогою повільного алгоритм старту.

Алгоритм швидкого відновлення є значним вдосконаленням TCP, яке було реалізовано з моменту випуску, відомого як TCP Reno. Алгоритм зазвичай використовується в поєднанні з алгоритмом швидкої повторної передачі.

Механізм контролю перевантаження покращує мережеву продуктивність протоколу TCP. Версія, відома як TCP Tahoe, використовує разом алгоритми повільного старту, уникнення перевантажень і швидкої

повторної передачі, щоб максимізувати передачу даних [9]. Tahoe працює за принципом збереження пакетів, відповідно до якого йому не потрібно вводити додаткові пакети в потік даних, якщо він працює з максимальною доступною пропускною здатністю.

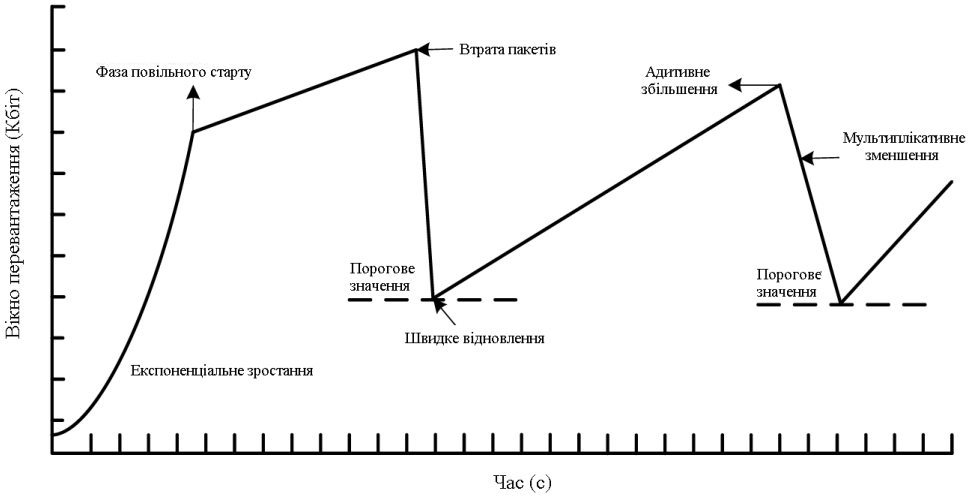

Рисунок 2.8 – «Зуб пилки», створений алгоритмом перевантаження TCP

TCP Reno працює за тими самими основними принципами, що й Tahoe, однак спосіб виявлення втрачених пакетів є більш складним, гарантуючи, що конвеєр даних не спустошується щоразу, коли відбувається втрата пакетів через передачу [10]. Пізніше було виявлено, що відправнику TCP часто доводилося чекати закінчення таймера, щоб відновити втрату кількох пакетів – це призводить до непотрібних затримок, і спровокувало прийняття SACK, який виправляє цю поведінку [11].

Однак на той час багато хостів TCP не підтримували його використання, що призвело до випуску модифікованого Reno. TCP New-Reno (спадкоємець TCP Reno) має навіть кращу здатність до втрати пакетів, оскільки він має здатність виявляти численні втрати пакетів. Таким чином, щодо модифікацій TCP, як TCP Reno з SACK, так і TCP-New Reno є вірогідними рішеннями для вирішення тієї самої проблеми контролю перевантаження. Заголовок TCP містить кілька фрагментів інформації,

життєво важливих для встановлення та використання каналу зв'язку. Що стосується проблем продуктивності TCP, то під час ініціалізації з’єднання параметр максимального розміру сегмента отримання використовується для інформування адресата про максимальний розмір сегмента.

Цей параметр встановлює розмір як максимального розміру сегмента прийому, так і розміру вікна TCP. Таким чином, цей параметр дозволить сегментам проходити без необхідності фрагментації, таким чином покращуючи продуктивність мережі.

Опція масштабу вікна в заголовку TCP також відіграє важливу роль у підвищенні продуктивності мережі, оскільки вона вирішує проблему максимального розміру вікна, регулюючи його відповідно до вимог передачі. Це дозволяє відправнику TCP ефективно змінювати розмір вікна, щоб утримувати більше даних залежно від пропускної здатності мережі та параметрів затримки мережі. Ці два параметри необхідно узгодити на початку фактичного сеансу TCP, щоб весь сеанс мав найбільший можливий розмір пакету (уникаючи фрагментації) і щоб розмір вікна відповідав атрибутам пропускної здатності/затримки шляху зв’язку.

3 ЕВОЛЮЦІЯ ПРОТОКОЛУ TCP

3.1 Варіанти протоколу TCP

Інтернет-трафік в основному складається з невеликих пакетів даних. Ці пакети містять інформацію про походження та призначення даних. Пакети створюються та збираються за допомогою протоколу керування передачею та надсилаються через Інтернет за допомогою Інтернет-протоколу.

Спочатку TCP був розроблений для провідних каналів, і провідні канали мають менші шанси на високу затримку та пошкодження даних через зовнішні впливи. Перевантаження є основною причиною втрати пакетів у провідних каналах зв'язку. Спочатку з'явилися варіанти під назвами Tahoe, Reno, New Reno і SACK та багато інших. TCP – це надійний наскрізний протокол, орієнтований на з'єднання. TCP забезпечує надійність, стартаючи таймер кожного разу, коли він надсилає сегмент. Якщо він не отримує підтвердження від одержувача протягом «тайм-ауту», то він повторно передає сегмент.

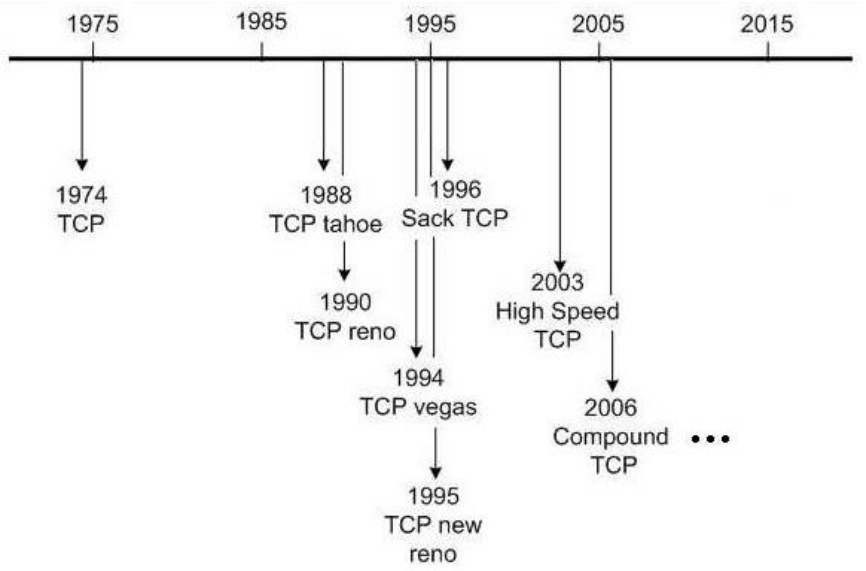

Рисунок 3.1 – Еволюція варіантів TCP

3.2 Різновиди TCP

3.2.1 TCP Tahoe

TCP Tahoe, один із варіантів TCP, був запропонований Ван Джекобсоном. Він додав деякі нові вдосконалення щодо завершення TCP на ранній стадії. Це покращення полягає в уникненні перевантаження, повільному старту та швидкій повторній передачі.

Щоб уникнути збою перевантаження, TCP дотримується принципу пакетної розмови [12], який підтверджує доставку переданого пакету за допомогою підтвердження. Для кожного пакета, надісланого в мережу джерелом, очікується, що ACK буде передано назад від пункту призначення. Джерело контролює швидкість надсилання пакетів за допомогою вікна перевантаження cwnd, яка визначає кількість пакетів, які джерело може надсилати.

Пункт призначення також повідомляє джерелу кількість даних, які він бажає буферизувати для з’єднання, що називається об’явленим вікном (rwnd).

Використовуючи ці дві змінні, джерело може передавати дані до максимального розміру вікна перевантаження або оголошеного вікна. Передача даних між джерелом і одержувачем залежить від порівняльних мінімальних значень cwnd або rwnd. На рисунку 3.2 показано типову поведінку контролю перевантаження TCP.

Контроль перевантаження TCP складається з чотирьох основних компонентів: повільний старт і уникнення перевантаження, швидка повторна передача та швидке відновлення.

Повільний старт і алгоритми уникнення перевантажень контролюють передачі. Поріг повільного старту (ssthresh) використовується для визначення того, який алгоритм, повільний старт або уникнення перевантаження використовуються TCP для керування передачею даних. Якщо величина

cwnd менша за ssthresh, то використовується алгоритм повільного старту, а якщо cwnd більше або дорівнює ssthresh, то використовується алгоритм уникнення перевантажень, як зазначено у формулі 3.1.

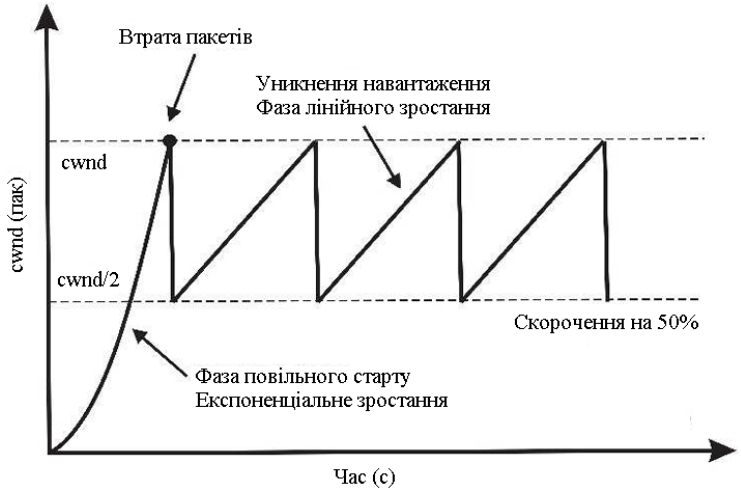

Рисунок 3.2 – Динаміка контролю перевантаження TCP Tahoe

$$\begin{cases} \text{cwnd} < \text{ssthresh} \quad \text{SlowStartAlgorithm} \\ \text{cwnd} \leq \text{ssthresh} \quad \text{CongestionAvoidanceAlgorithm} \end{cases}. \tag{3.1}$$

Під час початкового етапу підключення алгоритм повільного старту експоненціально збільшує вікно перевантаження, щоб знайти невідомий стан рівноваги мережі. Але, з іншого боку, алгоритм уникнення перевантаження контролює зростання вікна перевантаження, оскільки джерело вже досягло рівноважного стану мережі.

Після отримання трьох дублікатів ACK алгоритм швидкої повторної передачі повторно передає скинутий пакет, не чекаючи закінчення таймера повторної передачі.

Після швидкого алгоритму повторної передачі алгоритм швидкого відновлення продовжує працювати, щоб підтримувати ту саму кількість пакетів перед входом у швидке відновлення. Коли джерело отримує ACK про втрачені дані, алгоритм швидкого відновлення припиняє роботу.

Алгоритм повільного старту використовується для перевірки доступної смуги пропускання поточного мережевого шляху, що змінюється в часі. Джерело збільшує свій cwnd на одиницю при кожному ACK, що подвоює його під час отримання ACK для всіх пакетів. Формула 3.2 позначає зміну cwnd під час повільного старту після отримання ACK. Місце призначення, що використовує іншу схему ACK, впливає на швидкість нарощування алгоритму повільного старту на джерелі. Коли розмір cwnd стає більшим за ssthresh, джерело виходить із фази повільного старту та переходить у фазу уникнення перевантаження. ssthresh – це розрахункова консервативна міра доступної пропускної здатності каналу в мережевому шляху.

$$\text{cwnd} \leftarrow \text{cwnd} + 1 \quad \text{if cwnd} < \text{ssthresh}. \tag{3.2}$$

Заповнення мережного каналу у фазі повільного старту є важливою концепцією продуктивності. У фазі запобігання перевантаженню джерело зазвичай збільшує своє вікно навантаження на (1/cwnd) для кожного вхідного ACK. Це змушує джерело поступово збільшувати свій cwnd тільки на один пакет за кожен RTT, оскільки джерело вже досягло рівноважного стану мережі.

При виявленні втрат після отримання трьох дублікатів ACK джерело зменшує своє cwnd вдвічі. Рівняння 3.3 і 3.4 показують еволюцію cwnd під час запобігання навантаженню при отриманні ACK і при виявленні втрати відповідно.

Джерело використовує втрату пакетів як ознаку перевантаження мережі. За відсутності перевантаження мережі джерело відповідно до AIMD збільшує своє cwnd адитивно, у той час як за наявності перевантаження мережі, отримавши три дублікати ACK, джерело знижує своє cwnd на половину (1/2) свого поточного cwnd, щоб зменшити навантаження на мережному шляху. Рівняння 3.5 показує загальний Стандартний TCP (α=1) і (β=1/2).

$$\text{cwnd} = \text{cwnd} + \frac{1}{\text{cwnd}} \quad \text{if cwnd} \geq \text{ssthresh}. \tag{3.3}$$

$$\text{cwnd} \leftarrow \frac{1}{\text{cwnd}} \ . \tag{3.4}$$

$$\text{AIDM:} \begin{cases} \text{ACK} \quad \leftrightarrow \quad \text{cwnd} \leftarrow \text{cwnd} + \dfrac{\alpha}{\text{cwnd}}. \\ \text{Loss} \quad \leftrightarrow \quad \text{cwnd} \leftarrow (1-\beta) \cdot \text{cwnd} \end{cases} \tag{3.5}$$

Проблема Tahoe полягає в тому, що для виявлення втрати пакета потрібен повний інтервал часу очікування. Фактично, у більшості реалізацій це займає навіть більше часу. Крім того, оскільки він не надсилає негайні ACK, він надсилає сукупні підтвердження, тому він дотримується підходу «повернутися назад». Таким чином, кожного разу, коли пакет втрачається, він очікує тайм-ауту, і ковзне вікно скидається до 1. TCP Tahoe погано справляється з скиданням кількох пакетів в одному вікні даних.

3.2.2 TCP Reno

Reno визначається як TCP, що містить алгоритми повільного старту, швидкої повторної передачі, швидкого відновлення та уникнення перевантажень. Цей Reno зберігає основні принципи Tahoe, такі як повільний старт і таймер повторної передачі. Однак він додає деяку інтелектуальну інформацію, щоб втрачені пакети виявлялися раніше, а вікно передачі не скидалося щоразу, коли втрачається пакет.

Reno вимагає, щоб було отримано негайне підтвердження кожного разу, коли отримано сегмент. Логіка цього полягає в тому, що кожного разу, коли отримується дублікат підтвердження, його дублікат підтвердження міг бути отриманий, якщо очікуваний наступний сегмент у послідовності був затриманий у мережі, а сегменти досягли його не в порядку або пакет було втрачено.

Якщо отримуються кілька повторюваних підтверджень, це означає, що минуло достатньо часу, і навіть якщо сегмент пройшов довший шлях, він уже повинен був потрапити до одержувача. Є дуже висока ймовірність того, що він був втрачений. Тож Reno пропонує алгоритм під назвою «Швидка повторна передача».

Основна проблема в TCP Reno полягає в тому, що механізм швидкої повторної передачі передбачає, що втрачається лише один сегмент, якщо втрачається більше одного сегмента, це призводить до зниження продуктивності за наявності множинних втрат пакетів. TCP New Reno [13] та TCP SACK [14] вирішили цю проблему. Виснаження ACK – ще одна проблема TCP Reno, яка виникає через неоднозначність дублюючих ACK. Отже, TCP Reno кращий за TCP Tahoe тільки в разі втрати одного пакета, але не набагато краще, якщо губиться кілька пакетів. Більшість нових механізмів керування навантаженням TCP засновані на TCP Reno. Рівняння 3.6 і 3.7 представляють значення cwnd у фазах повільного старту та швидкого відновлення. На рисунку 3.3 показана типове поведінка зростання вікна навантаження TCP Reno.

$$\text{ACK:}\begin{cases} \text{cwnd} \leftarrow \text{cwnd} + 1 \\ \text{ssthresh} \leftarrow \frac{1}{2} \cdot \text{cwnd} \end{cases}. \tag{3.6}$$

$$\text{ACK:}\begin{cases} \text{ssthresh}_n \leftarrow \frac{1}{2} \cdot \text{cwnd}_n \\ \text{cwnd}_{n+1} \leftarrow \text{ssthresh}_n \end{cases}. \tag{3.7}$$

Reno дуже добре працює коли втрати пакетів невеликі. Але коли є кілька втрат пакетів в одному вікні, тоді RENO не працює надто добре, і його продуктивність майже така ж, як у Tahoe за умов високої втрати пакетів. Причина в тому, що він може виявити втрату лише одного пакета. Якщо

відкидається кілька пакетів, перша інформація про втрату пакета надходить, коли отримуються повторювані ACK. Але інформація про втрачений другий пакет надійде лише після того, як ACK для повторно переданого першого сегмента досягне відправника після одного RTT.

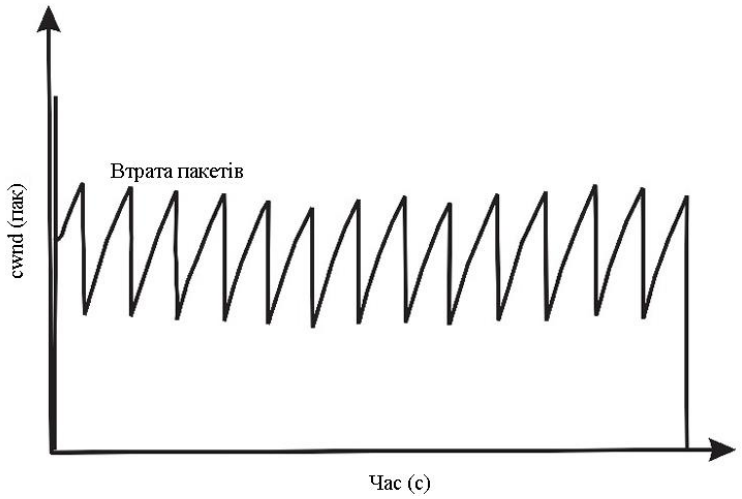

Рисунок 3.3 – Поведінка зростання вікна перевантаження TCP Reno

3.2.3 TCP Vegas

TCP Vegas – це модифікована версія TCP Reno. Він базується на тому факті, що профілактичні заходи для подолання заторів набагато ефективніші, ніж реактивні. Він намагався обійти проблему грубих тайм-аутів, запропонувавши алгоритм, який перевіряє тайм-аути за дуже ефективним графіком.

Крім того, він долає проблему потреби достатньої кількості дублікатів підтвердження для виявлення втрати пакета, а також пропонує модифікований алгоритм повільного старту, який запобігає перевантаженню мережі. Це не залежить лише від втрати пакетів, як ознаки перевантаження. Він виявляє перевантаження до того, як відбудеться втрата пакетів. Однак він все ще зберігає інший механізм Reno і Tahoe, і втрату пакета все ще можна виявити за грубим тайм-аутом інших механізмів.

Три основні зміни, застосовувані Vegas:

$$Cwnd(t+t_A) = \begin{cases} cwnd(t)+1 & \text{if} \quad diff < \alpha/base\_RTT \\ cwnd(t) & \text{if} \quad \alpha/base\_RTT < diff < \beta/base\_RTT, \\ cwnd(t)-1 & \text{if} \quad \beta/base\_RTT < diff \end{cases} \qquad (3.8)$$

$$diff = cwnd(t)/base\_RTT - cwnd/RTT.$$

Новий механізм ретрансляції Vegas поширюється на механізм ретрансляції Reno.

Він відстежує, коли було надіслано кожен сегмент, а також обчислює оцінку RTT, відстежуючи, скільки часу потрібно, щоб отримати підтвердження.

Кожного разу, коли отримується дублікат підтвердження, він перевіряє, чи (час передачі поточного сегмента часу)> оцінки RTT; якщо це так, то він негайно повторно передає сегмент, не чекаючи 3 дублікатів підтвердження або грубого тайм-ауту.

Таким чином, він усуває проблему, з якою стикається Reno, пов’язану з нездатністю виявити втрачені пакети, коли мале вікно й не отримується достатньо дублікатів Ack.

Щоб перехопити будь-які інші сегменти, які могли бути втрачені до повторної передачі, коли отримано підтвердження без дублікатів, якщо воно є першим або другим після нового підтвердження, він знову перевіряє значення часу очікування, і якщо час сегмента перевищує значення тайм-ауту, тоді він повторно передає сегмент, не чекаючи повторного підтвердження.

Таким чином Vegas може виявити численні втрати пакетів. Крім того, він зменшує своє вікно, лише якщо повторно переданий сегмент було надіслано після останнього зменшення. Таким чином, він також долає недоліки Reno щодо багаторазового зменшення вікна перевантаження, коли

втрачається кілька пакетів. Він не використовує втрату сегмента, щоб сигналізувати про перевантаження. Він визначає перевантаження шляхом зменшення швидкості надсилання порівняно з очікуваною швидкістю в результаті накопичення великих черг у маршрутизаторах. Таким чином, щоразу, коли обчислена швидкість надто далека від очікуваної, він збільшує передачу, щоб використати доступну смугу пропускання, щоразу, коли обчислена швидкість наближається занадто близько до очікуваного значення, він зменшує свою передачу, щоб запобігти перенасиченню смуги пропускання.

Таким чином, Vegas досить ефективно бореться з перевантаженням і не витрачає пропускну здатність, передаючи дані на надто високій швидкості та створюючи перевантаження, а потім скорочуючи, як це роблять інші алгоритми.

TCP Vegas відрізняється від інших алгоритмів фазою повільного старту. Причина цієї модифікації полягає в тому, що під час першого старту підключення воно не має уявлення про доступну пропускну здатність, і можливо, що під час експоненціального збільшення пропускна здатність збільшується на велику кількість і, таким чином, спричиняє перевантаження.

З цією метою Vegas експоненціально збільшує лише кожен другий RTT, між цим він обчислює фактичне значення до очікуваного, і коли різниця перевищує певний поріг, він виходить із повільного старту та переходить у фазу уникнення перевантаження.

TCP Vegas має дивовижну властивість стабілізації швидкості в стабільному стані, що може значно підвищити загальну пропускну здатність потоку TCP.

На жаль, пізніші дослідження виявили низку проблем, у тому числі недооцінку доступних мережевих ресурсів у деяких середовищах (наприклад, у випадку багатошляхової маршрутизації) та упередженість до нових потоків (тобто новачки отримують більшу частку) через доситьнеточні оцінки RTTmin.

3.2.4 Високошвидкісний TCP (HSTCP)

Оскільки модифікована функція відповіді високошвидкісного TCP діє лише з великими вікнами перевантаження, високошвидкісний TCP не змінює поведінку TCP у середовищах із сильним перевантаженням і, отже, не створює нових небезпек колапсу заторів.

Високошвидкісний TCP покращує продуктивність TCP у середовищах із високою пропускною здатністю. Високошвидкісний TCP (HSTCP) заслуговує на особливу увагу, оскільки він розроблений для роботи з подібною функцією відповіді, як стандартний TCP, але масштабований, щоб задовольнити вимоги вищого рівня безпеки.

Високошвидкісний TCP (HSTCP) є модифікацією запропонованою Флойдом, щоб швидше отримати доступну пропускну здатність (і швидше досягти повного використання каналу) з високою пропускною здатністю мережі.

Цільовими мережевими середовищами для HSTCP є мережі з низьким рівнем втрати пакетів, тому HSTCP пропонує швидше збільшення вікна перевантаження порівняно з TCP. Модифікована функція відповіді високошвидкісного TCP діє лише з більшим вікном перевантаження. Він не змінює поведінку TCP у середовищах із сильним перевантаженням і, отже, не створює нових небезпек збою перевантаження.

HighSpeed TCP використовує модифіковані параметри AIMD, де коефіцієнт лінійного збільшення становить f_α / cwnd, а мультиплікативний коефіцієнт зменшення $g_\beta(\text{cwnd})$ коригуються опуклою функцією для поточного розміру cwnd. Якщо розмір cwnd менший або дорівнює 38, HighSpeed TCP використовує подібні коефіцієнти збільшення та зменшення, як стандартний TCP. Коли cwnd перевищує граничне значення, функція підвищує коефіцієнт збільшення та зменшує коефіцієнт зменшення пропорційно розміру cwnd. На рисунку 3.4 показано зростання cwnd для цього протоколу.

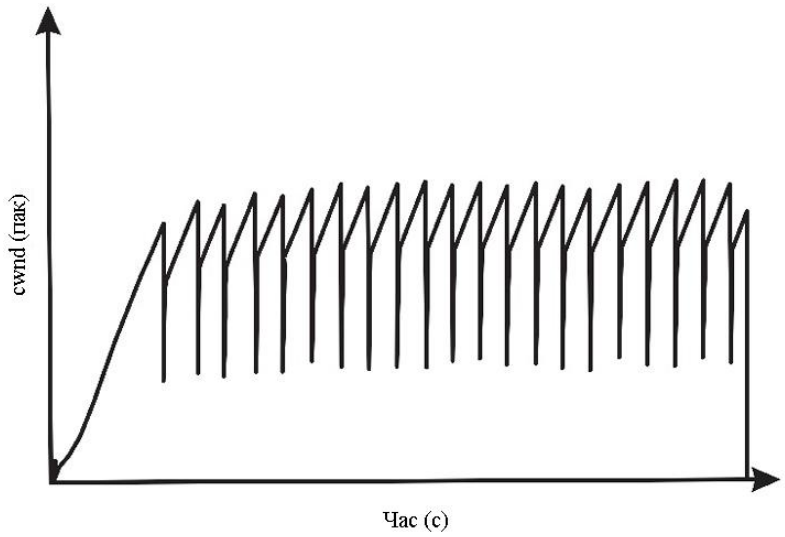

Рисунок 3.4 – Поведінка вікна перевантаження високошвидкісного TCP

Рівняння 3.9 та 3.10 позначають значення вікна перевантаження для кожної події ACK і втрати відповідно.

$$\text{ACK:} \quad \text{cwnd} \leftarrow \text{cwnd} + \frac{f_{\alpha}(\text{cwnd})}{\text{cwnd}}. \tag{3.9}$$

$$\text{Loss:} \quad \text{cwnd} \leftarrow g_{\beta}(\text{cwnd}) \cdot \text{cwnd}. \tag{3.10}$$

3.2.5 BIC TCP

BIC (Binary Increase Congestion Control Algorithm) TCP [15] використовує дві політики керування розміром вікна, які називаються адитивним збільшенням і бінарним збільшенням пошуку, щоб максимізувати cwnd. У разі втрати пакета BIC зменшує cwnd шляхом мультиплікативного зменшення (β) як показано в рівнянні 3.11.

Попередній розмір cwnd для зменшення встановлено (W_{max}), а після зменшення є (W_{min}). Оскільки втрата пакетів сталася при (W_{max}), розмір cwnd, який мережа наразі може обробляти без втрат, має бути дещо між цими

двома числами, тому BIC виконує бінарний пошук за допомогою параметрів (W_{max}) та (W_{min}), перейшовши до середньої точки між цими двома параметрами. Якщо відстань між середньою точкою (W_{max}+W_{min})/2 і нинішнього мінімуму (W_{min}) більше максимального приросту (S_{max}), BIC збільшує поточний розмір вікна на (S_{max}), таким чином cwnd=cwnd+S_{max} і це називається лінійним збільшенням. Якщо BIC не отримує втрати пакетів з оновленим розміром вікна, цей розмір вікна стає новим (W_{min}), а якщо втрачається пакет, оновлений розмір вікна стає новим (W_{max}). Цей процес триває доти, доки крок вікна не стане меншим за мінімальний крок (W_{min}) і у цей момент вікно встановлюється на поточний максимум (W_{max}). Функція зростання вікна BIC TCP пояснюється графічно на рисунку 3.5.

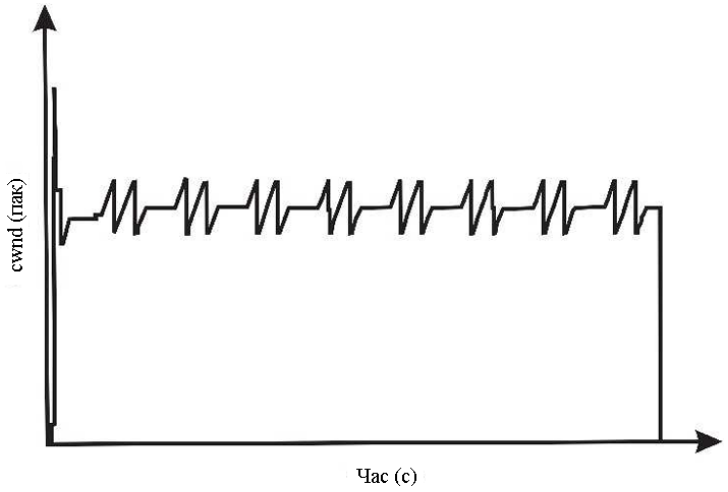

Рисунок 3.5 – Поведінка вікна перевантаження BIC TCP

$$\text{Loss:} \quad \text{cwnd} \leftarrow (1\text{-}\beta) \cdot \text{cwnd}\,. \tag{3.11}$$

3.2.6 TCP Cubic

TCP CUBIC [16] прийняв новий алгоритм повільного старту, так званий HyStart [17], який запобігає тривалим втратам пакетів, знаходячи безпечну точку виходу під час повільного старту і, таким чином, покращує

пропускну здатність TCP CUBIC при роботі в мережах з великою протяжністю та високою пропускною здатністю. Після зменшення розміру вікна через втрату TCP CUBIC встановлює (W_{max}) як розмір вікна, де сталася втрата. Потім він зменшує cwnd на постійний коефіцієнт зменшення (β) і входить у фазу уникнення перевантажень та починає збільшувати розмір вікна за допомогою увігнутої кубічної функції, доки розмір вікна не стане (W_{max}). Вікно зростає дуже швидко після зменшення вікна, але в міру наближення до (W_{max}), уповільнює свій ріст, навколо (W_{max}), приріст вікна стає майже нульовим.

Рівняння 3.12 показує функцію зростання TCP CUBIC, де (C) – параметр TCP CUBIC, (t) – час, що минув від останнього зменшення вікна, а (K) – період часу, потрібний функції для збільшення (W) до (W_{max}) (коли більше не відбувається подій з втратами. (K) розраховується за допомогою функції, наведеної у формулі 3.13. TCP CUBIC встановлює (W(t+RTT)) як потенційне цільове значення cwnd. Зростання cwnd цього протоколу показано на рисунку 3.6.

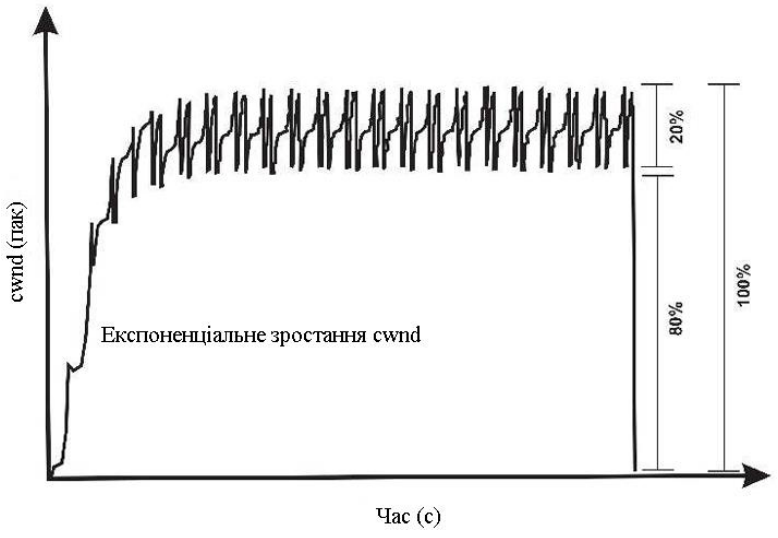

Рисунок 3.6 – Поведінка вікна перевантаження TCP CUBIC

$$W(t) = C(t-K)^3 + W_{max}. \quad (3.12)$$

$$K = \sqrt[3]{\frac{W_{max} \cdot \beta}{C}}. \quad (3.13)$$

3.2.7 Compound TCP

Compound TCP (CTCP) [18], розроблений Microsoft для операційної системи Vista і призначений для мереж з високою затримкою, використовує компонент, що масштабується на основі затримки TCP Vegas у стандартному алгоритмі запобігання перевантаженню TCP Reno.

Нова змінна стану, яка називається вікном затримки cwnd, введена в поточний блок керування TCP (TCB) для керування компонентом на основі затримки CTCP.

$$cwnd \leftarrow \begin{cases} cwnd + \frac{1}{win} & \text{після отримання ACK} \\ \frac{1}{2} \cdot cwnd & \text{після винекнення втрати} \end{cases}. \quad (3.14)$$

$$win = \begin{cases} cwnd+0 & \text{повільний старт} \\ cwnd+dwnd & \text{уникнення заторів} \end{cases}. \quad (3.15)$$

$$ACK: \quad win_{t+1} \leftarrow win_t + \alpha \cdot win_t^k. \quad (3.16)$$

$$Loss: \quad win_{t-1} \leftarrow win_t - \beta \cdot win_t. \quad (3.17)$$

$$dwnd_{t+1} \leftarrow \begin{cases} if \;\; Diff < \gamma \quad dwnd_t + \max((\alpha \cdot (win_t)^k - 1, 0) \\ \quad if \;\; Diff \geq \gamma \quad \max((dwnd_t - \zeta Diff, 0) \\ \text{після втрати} \;\; \max((win_t(1-\beta) - \frac{cwnd}{2}), 0) \end{cases}. \quad (3.18)$$

При старту нового з'єднання цей протокол використовує повільну поведінку старту звичайного TCP, збільшуючи cwnd способом, аналогічним TCP New Reno, як виражено в рівнянні 3.14, і встановлює значення dwnd на 0. Коли з'єднання перемикається на фазу запобігання навантаженню, включається компонент на основі затримки. Таким чином, CTCP підтримує два вікна одночасно, звичайний cwnd на основі застарілого алгоритму TCP AIMD та вікно затримки dwnd на основі компонента на основі затримки TCP Vegas.

Швидкість надсилання CTCP визначається (win) шляхом підсумовування цих двох вікон, як показано в рівнянні 3.15. Вікно затримки dwnd визначається за допомогою механізму затримки черги Vegas. При старту нового з'єднання цей протокол оцінює та вимірює (baseRTT), також відомий як (minRTT), та експоненційно згладжений час кругового оберту (sRTT). Він також оцінює кількість відкладених пакетів у з'єднанні як (Diff), що дорівнює:

$$\left(\frac{\text{win}}{\text{baseRTT}} - \frac{\text{win}}{\text{sRTT}}\right) \cdot \text{baseRTT}\,. \qquad (3.19)$$

Він позначає кількість даних, які було введено в мережу в останньому раунді, але не проходять через мережу в цьому раунді, тобто кількість даних, накопичених у маршрутизаторі з вузьким місцем. Якщо кількість пакетів у черзі (Diff) перевищує порогове значення, можна виявити ранню ознаку перевантаження (γ). Якщо (Diff<γ), мережевий шлях вважається недостатньо використаним, інакше мережевий шлях визначається як перевантажений.

За відсутності перевантаження вікно CTCP збільшується відповідно до рівняння 3.16 і якщо є втрати, вікно мультиплікативно зменшується відповідно до рівняння 3.17. Загальний CTCP відповідає поведінці, визначеній у цих двох рівняннях. Наприкінці кожного RTT обчислюється (Diff). На основі значення цього інтервалу (Diff) і глобального постійного

порогу (γ), що дорівнює 30 пакетам у цьому протоколі, (dwnd) обчислюється як у формулі 3.18, тут ζ – це параметр, який визначає, як швидко компонент на основі затримки повинен зменшувати це вікно, коли було раннє виявлення перевантаження. Якщо значення (Diff) невелике, dwnd збільшується дуже швидко, щоб максимально використати канал. Якщо воно велике, це означає, що мережевий шлях перевантажений і dwnd зменшується. На рисунку 3.7 показаний графік cwnd TCP Compound.

CTCP може ефективно використовувати мережевий ресурс і досягати високого рівня використання каналу. Теоретично CTCP може дуже швидко отримати вільну пропускну здатність мережі, використовуючи правило швидкого збільшення в компоненті на основі затримки, наприклад. мультиплікативне збільшення.

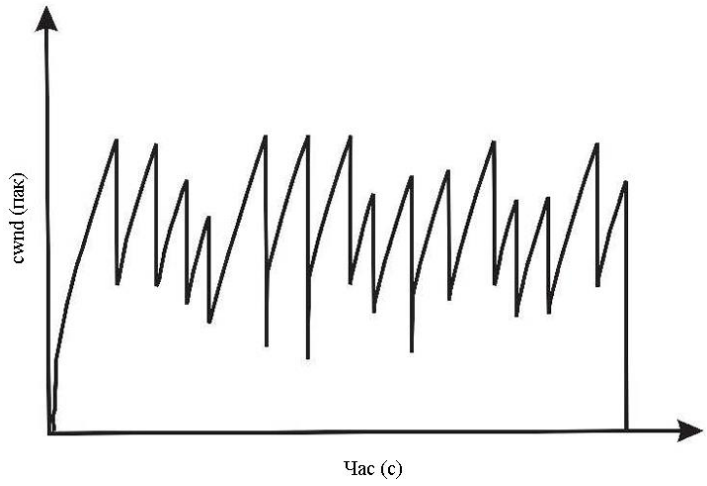

Рисунок 3.7 – Поведінка вікна перевантаження CTCP

CTCP має подібну або навіть покращену справедливість RTT порівняно зі звичайним TCP. Це пов’язано з компонентом на основі затримки, який використовується в алгоритмі уникнення перевантажень CTCP.

4 МЕХАНІЗМ КЕРУВАННЯ ПЕРЕВАНТАЖЕННЯМ TCP НА ОСНОВІ ВТРАТ

4.1 Необхідність вдосконалення TCP

TCP Compound і TCP Fusion є типовими механізмами контролю перевантаження TCP в операційних системах Microsoft Windows і Sun Solaris, тоді як TCP CUBIC є типовим механізмом контролю перевантаження в операційних системах Linux, Android і Free-BSD. Крім того, в даний час близько 50% Інтернет-трафіку контролюється TCP CUBIC замість механізму контролю перевантаження стандартного TCP Reno.

Таким чином, у центрі уваги кваліфікаційної роботи є підвищення продуктивності та ефективності TCP CUBIC для захищених мереж зв’язку з високою пропускною здатністю.

Перевантаження виникає, коли в мережевих маршрутизаторах занадто великий трафік даних, і кілька користувачів змагаються за доступ до тих самих мережевих ресурсів. Як було зазначено раніше, механізм контролю перевантаження складається з чотирьох фаз: повільний старт, уникнення перевантаження, швидкої повторної передачі та швидкого відновлення.

Перші дві фази механізму контролю перевантаження (повільний старт і уникнення перевантаження) відповідають за виявлення перевантаження, тоді як інші два компоненти (швидка повторна передача та швидке відновлення) реагують, щоб подолати перевантаження. В роботі пропонується метод, пов’язаний з фазою повільного старту. TCP дотримується принципу пакетної передачі, який підтверджує доставку переданого пакету за допомогою підтвердження (ACK). Цей принцип передачі пакетів зображено на рисунку 4.1, де показано, що вихідний вузол розбиває повідомлення даних на багато пакетів і надсилає до вузла призначення через мережу.

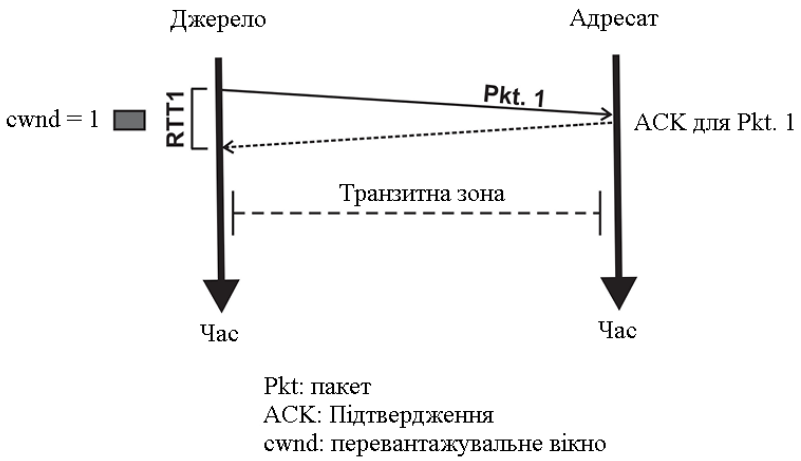

Рисунок 4.1 – Пакетний принцип передачі

Хост-джерело контролює швидкість передачі даних за допомогою вбудованої змінної Congestion Window (cwnd), яка визначає максимальну кількість пакетів даних, які вузол-джерело може відправити. Передача даних в Інтернеті використовує подібну концепцію cwnd. Раніше Інтернет-трафік контролювався стандартним TCP, який також відомий як TCP Reno, однак, Інтернет-трафік тепер контролюється декількома механізмами контролю перевантаження, такими як TCP Compound і TCP CUBIC.

Лише від 3,31% до 14,4% веб-серверів Інтернету все ще використовують TCP Reno, тоді як від 14,5% до 25,66% використовують TCP Compound і 46,92% використовують TCP CUBIC. Таким чином, більшість TCP-потоків в Інтернеті тепер контролюються TCP CUBIC замість TCP Reno [15]. Тепер TCP CUBIC також є механізмом контролю навантаження за замовчуванням в операційних системах Android і Free-BSD. У 2008 році було вдосконалено модуль повільного старту TCP CUBIC і запропоновано гібридний старт (HyStart), який покращив продуктивність TCP CUBIC у фазі повільного старту.

Під час початкового етапу з’єднання модулі повільного старту TCP TCP Reno, TCP Compound і TCP CUBIC використовуються для визначення змінної в часі невідомої доступної пропускної здатності мережевого шляху.

Після оцінки доступної пропускної здатності зв’язку фаза повільного старту TCP починає передачу з одного пакету, тобто cwnd=1, і після отримання кожного успішного ACK розмір cwnd збільшується на 1 додатковий пакет, як описано в рівнянні 4.1, що подвоює розмір cwnd наприкінці кожного часу проходження туди й назад (RTT), як описано в рівнянні 4.2. Рисунок 4.2 схематично ілюструє експоненціальне зростання розміру cwnd під час фази повільного старту. Через експоненціальне зростання cwnd розмір cwnd стає подвоєним наприкінці кожного RTT, таким чином, у мережах на великій відстані з високою пропускною здатністю, коли RTT дуже довгі, це може спричинити дуже великий розмір cwnd навіть на самому початковому етапі підключення у фазі повільного старту.

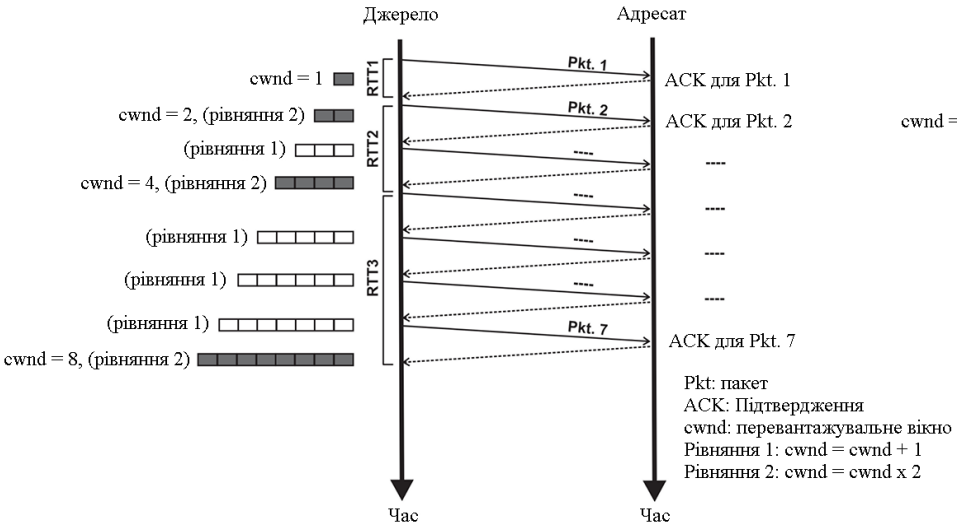

Рисунок 4.2 – Експоненціальне зростання вікна перевантаження під час фази повільного старту

Таким чином, експоненціальне зростання розміру cwnd на великій відстані з високою пропускною здатністю викликає перевантаження в мережі, що, у свою чергу, спричиняє втрату великої кількості пакетів, що також відомо як пакетні втрати пакетів. TCP CUBIC також страждає від пакетних втрат пакетів у фазі повільного старту [19].

$$\text{ACK}: \ \text{cwndnew} = \text{cwndPrevious} + 1. \tag{4.1}$$

$$\text{RTT}: \ \text{cwndnew} = \ \text{cwndPrevious} \cdot 2. \tag{4.2}$$

Поріг повільного початку (ssthresh) – це міра доступної пропускної здатності в поточному мережевому шляху для перемикання з'єднання на наступну фазу, яка відома як умовна змінна. Механізми контролю перевантаження використовують різні рівняння та формули для обчислення змінної ssthresh, яка, у свою чергу, впливає на точку виходу для фази повільного старту.

Точка виходу називається точкою завершення, коли розмір cwnd стає рівним або більшим за ssthresh, з'єднання виходить із фази повільного старту та переходить у фазу уникнення перевантаження.

TCP CUBIC збільшує розмір cwnd експоненціально, а не лінійно, тому TCP CUBIC дуже швидко займає доступну пропускну здатність. Таким чином, експоненціальне зростання cwnd TCP CUBIC на етапі уникнення перевантажень також викликає перевантаження в мережі, що, у свою чергу, викликає високу швидкість втрати пакетів. Рівняння 4.3 показує різницю зростання cwnd у фазі уникнення перевантажень TCP Reno, TCP Compound і TCP CUBIC:

$$\text{ACK}: \begin{cases} \text{cwnd} = \text{cwnd} + \dfrac{1}{\text{cwnd}} \quad \text{Лінійна} \quad \text{Reno} \\ \text{cwnd} = \text{cwnd} + \dfrac{1}{\text{cwnd}} \quad \text{Лінійна} \quad \text{Compound}. \\ \text{cwnd} = \text{cwnd} + 1 \quad \text{Експоненційна} \quad \text{CUBIC} \end{cases} \tag{4.3}$$

TCP використовує змінну ssthresh, щоб визначити, який модуль: повільний старт чи уникнення перевантаження слід використовувати для зв'язку. Якщо величина cwnd менша за ssthresh, тоді для зв'язку

використовується модуль повільного старту, а якщо cwnd більше або дорівнює ssthresh, то використовується модуль уникнення перевантажень, як зазначено в рівнянні 4.4.

$$\text{Congestion Control}: \begin{cases} \text{cwnd} < \text{ssthresh} & \text{Slow Start} \\ \text{cwnd} \geq \text{ssthresh} & \text{Cong. Avoidance} \end{cases}. \qquad (4.4)$$

4.2 Фаза повільного старту TCP

Механізм контролю перевантаження складається з фаз повільного старту, уникнення перевантаження, швидкої повторної передачі та швидкого відновлення, які також називаються модулями [20]. Модулі повільного старту та запобігання перевантаженням контролюють передачу даних, тоді як модулі швидкої повторної передачі та швидкого відновлення повторно передають втрачені дані. Фаза повільного старту TCP є першою фазою механізмів контролю перевантаження TCP.

Після завершення процесу тристороннього рукостискання TCP викидає додаткові пакети, які не дозволені погодженим розміром вікна (cwnd). Це не було великою проблемою в невеликих мережах, однак у міру того, як мережі зростали, а потім збільшувалася кількість підключених хостів, ці великі спалахи виявилися причиною проблем. У вузьких місцях мережі почали виникати перевантаження, дані накопичувалися швидше, ніж їх можна було переслати чи отримати. Тому був введений модуль для запобігання миттєвих сплесків. З включенням повільного старту було введено дві нові змінні: поріг повільного старту (ssthresh) і cwnd.

На цьому етапі модулі повільного старту використовують різні методи для оцінки доступної пропускної здатності каналу та збільшення розміру cwnd до обмеження ssthresh. Під час початку передачі cwnd встановлюється на 1 MSS (максимальний розмір сегмента), а ssthresh встановлюється на довільний розмір, залежно від механізму контролю перевантаження

операційної системи, що використовується. Обсяг даних, який відправник може надіслати, визначається min[cwnd, wnd], а оскільки cwnd=1 під час старту, дозволяється лише один пакет. cwnd тоді збільшиться на 1 MSS для кожного отриманого ACK. Це експоненціальне зростання триватиме до тих пір, поки не буде виявлено втрату або cwnd стане рівним ssthresh, коли це стається, алгоритм уникнення перевантаження починає роботу.

На початку передачі доступна пропускна здатність мережі невідома. Отже, метою модулів повільного старту є приблизно оцінити доступну пропускну здатність каналу. Експоненціальне зростання стандартного модуля повільного старту є швидким способом заповнення мережі. Оскільки стандартний повільний старт не може оцінити доступну пропускну здатність каналу, доки не станеться втрата пакетів, стандартний модуль повільного старту перевищує розмір cwnd, щоб використовувати доступну пропускну здатність каналу, що призводить до значного збільшення RTT і пакетних втрат пакетів.

Проблема оцінки доступної пропускної здатності каналу стандартного повільного старту була вирішена підходом Vegas, для оцінки доступної пропускної здатності без втрат пакетів. Він експоненціально збільшує розмір cwnd альтернативно (не в кінці кожного RTT). Однак підхід Vegas передчасно припиняє фазу повільного старту та переходить у фазу уникнення перевантажень, поки BDP мережі високий. Пізніше проблему Вегаса було вирішено за допомогою підходу, який уникає джерела передчасного припинення фази повільного старту. Підхід Хоу покращив продуктивність повільного старту TCP, встановивши краще початкове значення ssthresh як оціночне значення Bandwidth-Delay Product (BDP), яке вимірюється за допомогою методу пар пакетів.

Інший перехресний трафік може перешкоджати правильній оцінці доступної пропускної здатності каналу за допомогою методу пар пакетів, оскільки кілька потоків отримують однакову оцінку доступної пропускної здатності каналу.

Пізніше було запропоновано Additive Start [21] для оцінки доступної пропускної здатності каналу за допомогою техніки TCP Westwood під назвою «Оцінка прийнятної швидкості» (ERE – Eligible Rate Estimation).

Використовуючи ERE, Additive Start може багаторазово скидати ssthresh до більш відповідного значення. Адитивний старт повільніший за стандартний повільний старт і може перевищити розмір cwnd, оскільки кілька потоків обчислюють одне й те саме значення ERE. У 2003 році було запропоновано оцінку доступної пропускної здатності каналу за допомогою методів інтервалу пакетів і інтервалів ACK. Paced Starts включає механізм оцінки пропускної здатності в стандартний повільний старт. Однак пізніше це вимірювання розриву виявилося складним для мереж на великі відстані з високою пропускною здатністю.

Paced Start [22], який використовує техніку ланцюжків пакетів для оцінки доступної пропускної здатності каналу, уникає джерела TCP від передчасного завершення фази повільного старту (передчасне перемикання з фази повільного старту на уникнення перевантаження).

Отже, у всіх методах із повільним стартом розмір cwnd експоненціально зростає, що спричиняє втрати пакетів. У [23] було вперше представлено метод лінійного та стабільного зростання cwnd. Використовуючи лінійне та стабільне зростання cwnd, втрати пакетів зменшуються.

Галоп-Вегас ефективніший, ніж підхід Вегаса. Gallop Vegas також підходить для міжміських мереж із високою пропускною здатністю. Пізніше було запропоновано кілька методів повільного старту, які використовували інформацію зворотного зв’язку маршрутизатора для кращої оцінки доступної пропускної здатності каналу.

Наприклад, Quick Start використовує явний зворотний зв’язок маршрутизатора (ERF – Explicit Router Feedback), a Additive Limited Slow Start використовує технологію простої схеми сповіщення ресурсів Інтернету (SIRENS) для оцінки доступної смуги пропускання.

Сьогодні найбільш часто використовувані операційні системи, такі як Microsoft Windows, Linux, Solaris і Android, все ще використовують експоненціальне зростання cwnd під час фази повільного старту. Продуктивність TCP страждає від метода повільного старту TCP у мережах на великі відстані з високою пропускною здатністю.

Однак продуктивність TCP під час фази повільного старту можна підвищити шляхом інтелектуального налаштування ssthresh і розумного налаштування розміру cwnd. Будучи точкою перемикання між повільним стартом і уникненням перевантаження, ssthresh має вирішальне значення для продуктивності TCP. Якщо параметр ssthresh встановлено надто низьким, TCP передчасно перемикається з повільного старту на уникнення перевантаження, що може спричинити дуже тривалий час для досягнення відповідного розміру вікна перевантаження. Однак високий ssthresh може призвести до втрати кількох пакетів і, що більш серйозно, може спричинити тайм-аути TCP.

Пропонується розширений метод повільного старту з використанням ідеї Альтернативного повільного старту та практичної реалізації HyStart. Розширений метод повільного старту змінює розмір cwnd у фазі повільного старту HyStart.

4.3 Метрики продуктивності

4.3.1 Справедливість протоколу

Справедливість протоколу являє собою співвідношення частки смуги пропускання каналу між двома потоками TCP, налаштованими з тим самим механізмом управління перевантаженням (TCP CUBIC).

Під час аналізу справедливості протоколу обидва потоки повинні бути налаштовані з тим самим механізмом управління перевантаженням. Справедливість визначається як рівність частки смуги пропускання каналу

між конкуруючими потоками одного і того ж механізму управління навантаженням (TCP CUBIC) у мережі. Справедливість протоколу обчислюється за допомогою формули індексу справедливості Джайна [24], яка визначена у рівнянні 4.5. Для заданого набору пропускних здібностей (x_1, x_2, x_3, x_4, x_5...x_n,) ця формула обчислює індекс справедливості:

$$f\left(x_1, x_2, x_3, x_4, x_5 \ldots x_n\right) = \frac{\left(\sum_1^n x_i\right)^2}{n \cdot \sum_1^n x_i^2}. \tag{4.5}$$

Індекс справедливості – це значення від 0 до 1, де значення 1 показує найбільш рівний розподіл доступної пропускної спроможності каналу між конкуруючими потоками в мережі, а 0 показує несправедливий розподіл доступної пропускної спроможності каналу.

4.3.2 ЧАС Сходності

Час збіжності методу управління перевантаженням – це час, необхідний потоку 2 щоб одержати 80% пропускної спроможності потоку 1. Час збіжності методів управління навантаженням розраховується з використанням рівнянь 4.6 та 4.7. Де C_{f1} являє розмір cwnd потоку 1, C_{f2} представляє розмір cwnd потоку 2, T_1 – початковий час передачі потоку 1, який встановлено на 1 секунду, тоді як T_2 – початковий час передачі потоку 2, а T_3 – час у секундах, коли розмір cwnd C_{f2} досягає 80% обсягу cwnd C_{f1} вперше. Для розрахунку часу збіжності T_{sim} є загальним часом моделювання. Віднімаючи початковий час потоку 2 з T_3, обчислюється час збіжності T [25].

$$T_3 = \begin{cases} \text{for} \quad (i = 1; i \le T_{sim}; i++) \\ \text{if} \quad C_{f2} \ge \dfrac{80 \cdot C_{f1}}{100} \end{cases}. \tag{4.6}$$

$$T = T_3 - T_2 . \quad (4.7)$$

4.3.3 Корисна продуктивність

Корисна продуктивність – це міра обсягу переданих даних (реальний обсяг даних без втрачених пакетів). Визначення корисної продуктивності еквівалентне визначенню ефективної пропускної спроможності. Корисна продуктивність – це пропускна здатність прикладного рівня, виміряна на вузлі джерела даних TCP. В усіх симуляціях корисна продуктивність розраховується з допомогою рівняння 4.8. Корисна продуктивність представлена у вигляді стовпчастих діаграм, представлених у [26].

$$\text{Goodput} = \left(\frac{\text{SentData -RetransmittedData}}{\text{TransferTime}} \right). \quad (4.8)$$

4.3.4 Швидкість втрати пакетів

Втрата пакетів – це нездатність більш ніж одного переданого пакета досягти зазначеного пункту призначення. Швидкість втрати пакетів залежить від рівня навантаження мережі. Швидкість втрати пакетів розраховується за кількістю пакетів, втрачених за одиницю часу.

4.4 Розробка методу

TCP CUBIC використовує нижню граничну межу та верхню граничну межу для розміру cwnd у фазі повільного старту, який визначає мінімальний та максимальний діапазон розміру cwnd у фазі повільного старту.

Однак TCP CUBIC збільшує розмір cwnd на 1 для кожного вхідного ACK замість часу проходження туди й назад і дуже швидко займає всю доступну пропускну здатність каналу. Цей тип зростання називається

експоненціальним зростанням cwnd. Таким чином, це експоненціальне зростання cwnd механізму TCP CUBIC на фазі уникнення перевантажень викликає перевантаження в мережі, що, у свою чергу, спричиняє високу швидкість втрати пакетів під час зв’язку.

4.4.1 Метод M-SS

M-SS – метод повільного старту шляхом зміни нижньої межі розміру cwnd у фазі повільного старту TCP. Нижня і верхня граничні межі розміру cwnd у фазі повільного старту впливають на його експоненційне зростання (як у фазах повільного старту, так і запобігання перевантаженню) і точку перемикання з'єднання з фази повільного старту на фазу запобігання перевантаженню, що, у свою чергу, впливає на швидкість втрати пакетів за допомогою зниження втрат пакетів при пакетному старті.

TCP CUBIC використовує гібридний старт (HyStart) як свій модуль повільного старту за замовчуванням. HyStart встановлює нижній та верхній граничні межі для зростання розміру cwnd у фазі повільного старту. Ідея M-SS полягає в граничних межах розміру cwnd HyStart. Таким чином, швидкість втрати пакетів можна знизити, використовуючи нову граничну межу для розміру cwnd у фазі повільного старту. Таким чином, TCP-з'єднання може перемикатися з фази повільного старту на фазу запобігання навантаженню без втрати великої кількості пакетів.

M-SS виявляє точку виходу управління перевантаженням для cwnd, щоб переключити з'єднання з фази повільного старту на фазу запобігання перевантаженню. Точка виходу менша від умовної змінної, значення якої обчислюється під час виконання під час зв'язку. Втрата пакетів відбудеться, якщо розмір cwnd більший за умовну змінну. M-SS збільшує нижню межу розміру cwnd у фазі повільного старту, що збільшує можливість наборів значень розміру cwnd. Використовуючи цю ідею, ефект експоненційного зростання cwnd зменшується, що, своєю чергою, знижує втрати пакетів.

4.4.2 Тестування і оцінка продуктивності

Тестування та оцінка продуктивності M-SS є заключною фазою розробки методу. Перший крок цієї фази складається з конфігурації та налаштування моделювання.

На другому кроці описуються кілька метрик продуктивності, що найбільш широко використовуються. Для оцінки продуктивності пропонованих методів використовуються показники втрати пакетів, справедливості протоколу, часу збіжності та продуктивності корисної пропускної спроможності.

4.4.3 Налаштування імітаційного моделювання

Оцінка M-SS на основі моделювання виконується за допомогою Network Simulator-3 (NS-3) версії 3.35. У всіх експериментах із моделюванням використовується набір тестів Hamilton (www.hamilton.ie), який є найпоширенішим і широко використовуваним тестом NS-2 для аналізу ефективності механізмів контролю перевантаження. Налаштування моделювання включає параметри моделювання, топологію мережі, потоки TCP, спільне використання смуги пропускання між потоками TCP і модель трафіку.

У таблиці 4.1 наведено повний набір параметрів моделювання, які використовуються в усіх експериментах моделювання для мереж з великими та малими RTT. Використовуючи значення різних параметрів із таблиці, експерименти моделювання виконуються для кожного механізму контролю перевантаження TCP. Для отримання точних результатів експерименти моделювання для кожної конфігурації параметрів повторюються тричі. Час початку потоку 1 становить одну секунду, тоді як потік 2 починається через дві секунди для симуляції справедливості та через 150 секунд для моделювання часу конвергенції.

Усі TCP-з’єднання підключаються за допомогою агентів протоколу передачі файлів (FTP), і симуляції виконуються протягом 600 і 300 секунд. Той самий процес моделювання повторюється з тим самим часом початку та закінчення, щоб встановити легітимність оцінювання. Для коротких мережевих RTT, RTT потоку 1 є фіксованим, що дорівнює 50 мс, а RTT потоку 2 змінюється на 2 мс, 4 мс, 6 мс, 8 мс, 10 мс, 12 мс, 14 мс і 16 мс. Це пояснюється тим, що мережі від 2 мс до 16 мс вважаються мережами на короткі відстані, розгорнутими, наприклад, всередині будівлі.

Для великих RTT фіксовано значення RTT потоку 1, яке дорівнює 100 мс, а діапазон RTT потоку 2 становить 50 мс, 70 мс, 90 мс, 110 мс, 130 мс, 150 мс, 170 мс і 190 мс. Це пояснюється тим, що мережі від 50 мс до 200 мс вважаються мережами великої відстані, розгорнутими в межах міста чи країни.

Таблиця 4.1 – Параметри моделювання

Параметр моделювання	Значення параметрів
Протокол TCP	TCP Linux
Методи	CUBIC, BIC, TCP, Reno, Highspeed
Пропускна здатність вузького місця	50-250 Мб/с
Пропускна здатність каналів	100-500 Мб/с
Сценарії мережі	малий RTT, великий RTT
Розмір BDP-Q	0.01 to 2.0
Фоновий трафік	відсутній
Час моделювання	600 с
Повторення	3 рази

Топологія мережі гантелей з шести вузлів використовується в усіх експериментах моделювання. Топологія гантельової мережі, показана на рисунку 4.3, з’єднує групу вузлів-відправників S1 і S2 з одним

маршрутизатором R1, R1 послідовно з'єднаний з іншим маршрутизатором R2, який, у свою чергу, з'єднаний з іншою групою вузлів-одержувачів D1 і D2. Пряма лінія показує зв'язок між вузлами відправника та адресата зі швидкістю зв'язку 100 Мбіт/с, 200 Мбіт/с, 300 Мбіт/с, 400 Мбіт/с і 500 Мбіт/с.

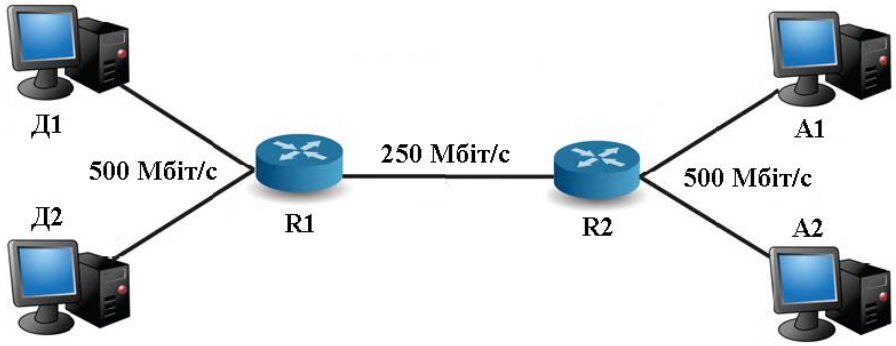

Рисунок 4.3 – Топологія експериментальної мережі

Пропускна здатність цього зв'язку може варіюватися від 100 Мбіт/с до 500 Мбіт/с для різних конфігурацій і різних сценаріїв мережі. Пропускна здатність послідовного з'єднання між двома маршрутизаторами також змінюється від 50 Мбіт/с до 250 Мбіт/с для різних експериментів.

Для всіх експериментів з моделюванням розмір буферної черги обох маршрутизаторів встановлено на [0,01, 0,02, 0,05, 0,1, 0,2, 0,4, 0,5, 1,0, 1,5, 2,0] відсотка продукту затримки пропускної здатності (BDP) за допомогою алгоритму DropTail. Буферна черга маршрутизатора використовується для тимчасового зберігання пакетів даних. Якщо розмір черги буфера дуже малий, швидкість втрати пакетів збільшиться, а якщо розмір черги буфера дуже великий, буде важко перевірити поведінку механізмів контролю перевантажень.

Затримка з'єднання між вузлами джерела та призначення та затримка вузьких місць між маршрутизаторами змінюються залежно від мережевого сценарію. Діапазон від 2 мс до 16 мс і від 50 мс до 190 мс використовується

для коротких RTT і довгих RTT відповідно. Таким чином, у всіх експериментах використовуються дуже точні та помірні діапазони розміру черги та затримки зв’язку.

Вихідний вузол S1 потоків TCP надсилає сегменти даних до кінцевого вузла D1, використовуючи cwnd як логічне відро. Сегменти даних проходять через смугу пропускання між маршрутизаторами R1 і R2 і досягають кінцевого вузла D1.

Після успішного отримання сегмента даних кінцевий вузол D1 надсилає ACK вихідному вузлу S1. Сигнал ACK від вузла призначення D1 також проходить через смугу пропускання між маршрутизаторами R1 і R2. Цей обмін даними між вихідним вузлом S1 і кінцевим вузлом D1 створює потік даних між джерелом і призначенням. Оскільки це перший потік TCP на каналі зв’язку, він називається TCP-потоком 1. Вихідний вузол S1 налаштовано за допомогою механізму контролю перевантаження TCP CUBIC під час конфігурації моделювання, таким чином, TCP-потік 1 також називається TCP CUBIC-потоком 1. Таким чином, якщо в смузі пропускання каналу немає інших потоків, то TCP CUBIC-потік 1 може повністю використовувати пропускну здатність каналу.

Подібним чином, якщо вихідний вузол S2 також надсилає сегменти даних до кінцевого вузла D2 за допомогою cwnd і тієї самої смуги пропускання між маршрутизаторами R1 і R2. Ця друга розмова також створює назву нового потоку TCP як TCP-потік 2. Як обговорювалося раніше, обидва потоки TCP CUBIC спільно використовують спільну смугу пропускання для передачі.

Як обговорювалося раніше, коли два потоки TCP CUBIC спільно використовують спільну пропускну здатність каналу і якщо пропускна здатність обох потоків TCP CUBIC дорівнює (або майже дорівнює) один одному, або іншими словами, якщо справедливість протоколу обох filows дорівнює (або майже дорівнює) 1, це означає, що обидва потоки TCP CUBIC спільно використовують спільну пропускну здатність каналу справедливо.

Однак, якщо обидва потоки TCP CUBIC не розподіляють смугу пропускання рівномірно один з одним навіть на один момент, то справедливість протоколу дорівнюватиме 0, як показано на рисунку 4.5.

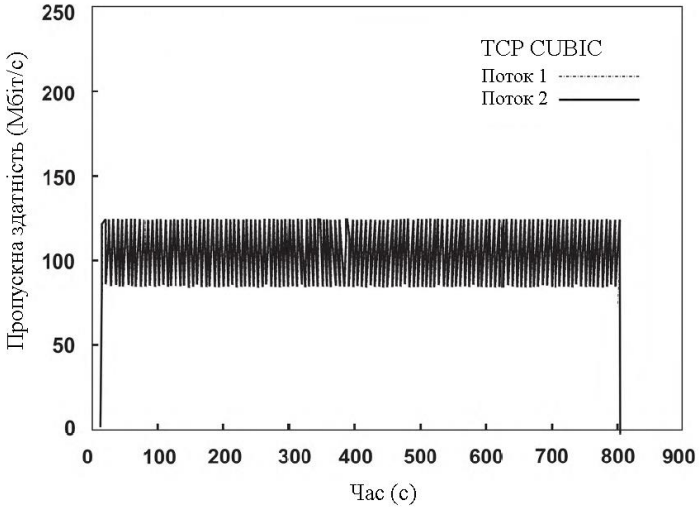

Рисунок 4.4 – Поведінка протоколу TCP CUBIC справедливість = 0,9

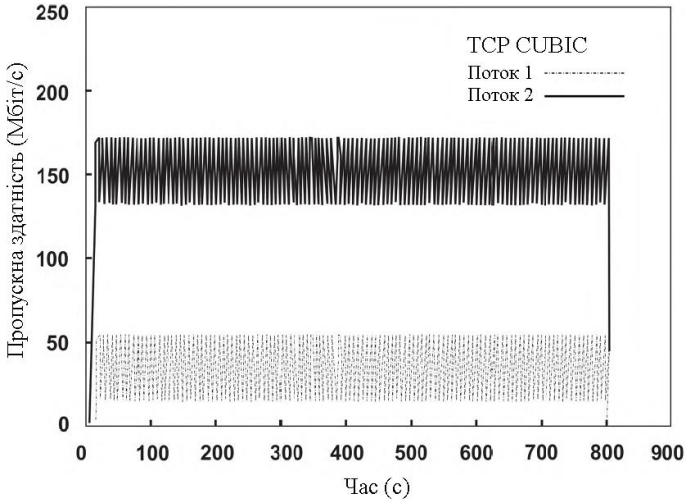

Рисунок 4.5 – Поведінка протоколу TCP CUBIC справедливість = 0

Джерелом трафіку встановлено протокол передачі файлів (FTP). Під час експериментів з моделюванням агенти FTP використовують TCP для надсилання та отримання пакетів даних, а механізми контролю перевантажень підключаються до TCP.

Для оцінки продуктивності M-SS використовуються показники коефіцієнта втрати пакетів, справедливості протоколу, часу конвергенції та хорошої пропускної здатності. Перевірка статистичної значущості, а саме t-критерій, також розраховується для оцінки переваг удосконалень, отриманих завдяки вдосконаленому механізму контролю перевантаження. Результати t-критерію цього дослідження задовольняли рівень значущості, тобто менше 0,05. Усі експерименти повторюють три рази, щоб отримати більш точні результати, а кінцеві результати усереднюють, щоб мати 95% довірчий інтервал середнього значення.

Для оцінки продуктивності M-SS проводяться кілька експериментів з моделюванням, а їх результати порівнюються з найбільш відповідними сучасними механізмами контролю перевантаження (TCP Reno, HighSpeed TCP, TCP BIC, TCP CUBIC і TCP Compound).

4.5 Розробка методу M-SS

M-SS зменшує пакетні втрати пакетів, збільшуючи нижню межу розміру контролю перевантаження (cwnd) у фазі повільного старту та визначаючи безпечну точку виходу для завершення фази повільного старту. Оскільки M-SS є модулем розширеного TCP CUBIC, і його методи базується на методах TCP CUBIC за замовчуванням; тому при розробці цього метода також буде використано багато параметрів TCP CUBIC за замовчуванням.

Нехай B, представляє доступну пропускну здатність каналу поточного мережевого шляху, minD представляє односторонню затримку прямого шляху, S представляє доступний розмір буфера і β представляє параметр зменшення cwnd.

M-SS зосереджується на експоненціальному зростанні розміру контролю перевантаження (cwnd) у фазі повільного старту, щоб уникнути втрати пакетів. У фазі повільного старту розмір cwnd зростає експоненціально.

Граничні межі розміру cwnd впливають на експоненціальне зростання cwnd, що, у свою чергу, впливає на точку завершення фази повільного старту.

Якщо з’єднання не завершує фазу повільного старту вчасно, втрата пакетів виникає через експоненціальне зростання cwnd. Таким чином, швидкість втрат пакетів можна зменшити, використовуючи розширену нижню межу для розміру cwnd у фазі повільного старту. Використовуючи розширену нижню межу розміру cwnd, з’єднання вчасно припиняє фазу повільного старту та переходить у фазу уникнення перевантаження без втрати надто великої кількості пакетів.

На рисунку 4.6 схематично показано точку завершення фази повільного старту та умовну змінну.

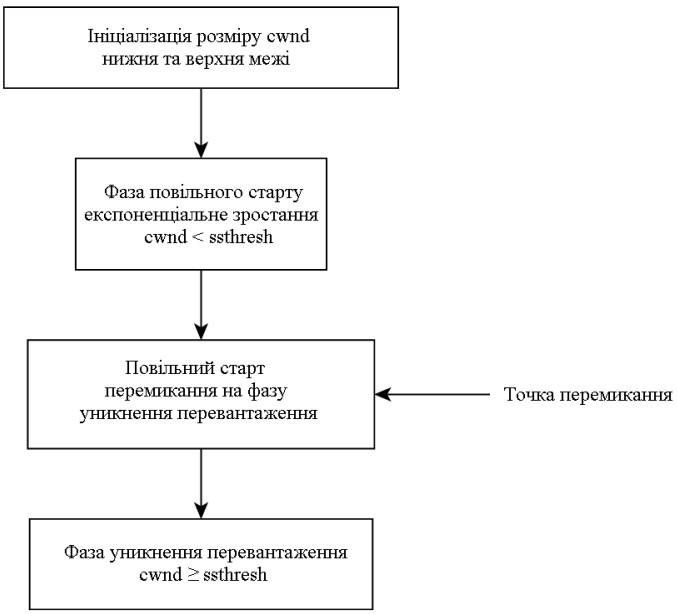

Рисунок 4.6 – Точка завершення фази повільного старту

M-SS виявляє точку виходу контролю перевантаження для cwnd, щоб переключити з’єднання з повільного старту на фазу уникнення перевантаження. Ця точка виходу має бути меншою за умовну змінну під назвою SCCP, значення якої обчислюється під час виконання за допомогою

рівняння 4.9, де $\bar{B}$ – це кількість доступної пропускної здатності каналу, $\min\bar{D}$ – мінімальна затримка прямого шляху в один бік (RTT/2), а $\bar{S}$ представляє доступний розмір буфера, який залежить від розміру черги продукту затримки пропускної здатності (BDP). Втрата пакетів відбудеться, якщо розмір cwnd перевищує значення SCCP. Ідея SCCP взята з точки безпечного виходу модуля HyStart, який є модулем повільного старту за замовчуванням у TCP CUBIC.

$$\text{SCCP} = (\bar{B} \cdot \min\bar{D} + \bar{S}) \quad \text{if } (\text{cwnd} > \text{SCCP}), \quad \text{Буде втрата пакету}. \tag{4.9}$$

4.6 Оцінка пропускної здатності та затримки

Для оцінки доступної пропускної здатності каналу M-SS використовує концепцію, подібну до пакетної передачі. Припустимо, що джерело передає $\bar{N}$ один до одного пакетів розміром $\bar{L}$ до пункту призначення. Для ($\bar{N} > 2$) ці односторонні пакети називаються послідовностями пакетів. Довжину цієї послідовності пакетів позначають $\Delta(\bar{N})$, що дорівнює $\sum_{1}^{\bar{N}-1}\delta_k$, як позначено в рівнянні 4.10:

$$\Delta(\bar{N}) = \sum_{1}^{\bar{N}-1}\delta_k, \tag{4.10}$$

де $\bar{N}$ – це кількість пакетів у серії, δ_k – інтервал часу між пакетами k і k+1, як показано на рисунку 4.7.

Використовуючи довжину серії пакетів, пункт призначення може виміряти смугу пропускання $b(\bar{N})$ каналу, як зазначено в рівнянні 4.11:

$$b(\bar{N}) = \frac{(\bar{N}-1)\cdot\bar{L}}{\Delta(\bar{N})}, \tag{4.11}$$

$$b(\bar{N}) = \frac{(\bar{N}-1)\cdot \bar{L}}{\sum_{1}^{\bar{N}-1} \delta_k}. \qquad (4.12)$$

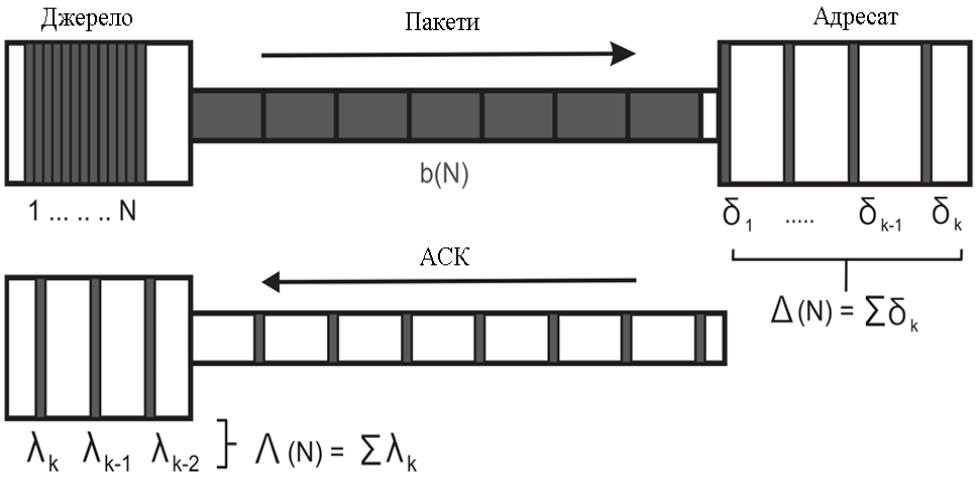

Рисунок 4.7 – Концепція пакетної передачі

За допомогою концепції послідовностей пакетів обчислюється доступна пропускна здатність каналу. Відповідно до цього підходу, якщо $\bar{B}$ представляє доступну пропускну здатність каналу для прямого тракту, а $\min\bar{D}$ представляє мінімальну затримку прямого одностороннього зв'язку, яка дорівнює половині RTT, тоді продукт затримки пропускної здатності (BDP) каналу зв'язку можна позначити як $(\bar{B}\cdot\min\bar{D})$, що позначено в рівнянні 4.13:

$$BDP = \bar{B}\cdot\min\bar{D} = b(\bar{N})\cdot\min\bar{D}. \qquad (4.13)$$

Розв'язування рівнянь 4.11 і 4.13 $(\bar{B}\cdot\min\bar{D})$ оновлено та показано у рівнянні 4.14:

$$BDP = \bar{B}\cdot\min\bar{D} = \frac{(\bar{N}-1)\cdot\bar{L}}{\Delta(\bar{N})}\cdot\min\bar{D}. \qquad (4.14)$$

Якщо $\Delta(\bar{N})$, дорівнює $\min\bar{D}$, тоді $(\bar{B}\cdot\min\bar{D})$ дорівнюватиме $(\bar{N}-1)\cdot\bar{L}$, як описано в рівнянні 4.15:

$$BDP = \bar{B}\cdot\min\bar{D} = (\bar{N}-1)\cdot\bar{L}. \quad (4.15)$$

Тоді $(\bar{N}-1)\cdot\bar{L}$ представляє розмір cwnd, тобто коли $\Delta(\bar{N})$ дорівнює $\min\bar{D}$, cwnd стає рівним $(\bar{B}\cdot\min\bar{D})$, як описано в рівнянні 4.16:

$$BDP = \bar{B}\cdot\min\bar{D} = cwnd. \quad (4.16)$$

Розв'язавши рівняння 4.16, можна розрахувати доступну пропускну здатність зв'язку $\bar{B}$, як описано в рівнянні 4.17:

$$\bar{B} = \frac{\min\bar{D}}{cwnd}. \quad (4.17)$$

Використовуючи серію підтвердження, оцінюється $\Delta(\bar{N})$, що дорівнює сумі часу між надходженнями пакетів у серії, як показано на рисунку 4.9. $\Lambda(\bar{N})$ представляє період часу між отриманням першого та останнього ACK в послідовності ACK. $\min\bar{D}$ розраховується шляхом ділення мінімального спостережуваного RTT на 2, як визначено в рівнянні 4.18:

$$\min\bar{D} = \frac{\min RTT}{cwnd}. \quad (4.17)$$

Метою M-SS є налаштування нижньої граничної межі $(\bar{B}\cdot\min\bar{D}\cdot\mu\beta)$ розміру cwnd у фазі повільного старту, щоб мати більш низькі початкові значення розміру cwnd для повільного використання доступної пропускної здатності каналу. Таким чином, M-SS збільшує кількість значень cwnd, що, у

свою чергу, спричиняє низьку швидкість втрати пакетів. Порівняння між нижніми та верхніми межами HyStart і M-SS для початкового розміру (cwnd) під час фази повільного старту схематично зображено на рисунку 4.8. На цьому рисунку перший прямокутник представляє граничні межі cwnd HyStart, а другий показує граничні обмеження розміру cwnd M-SS у фазі повільного старту. M-SS здвинув нижню межу розміру cwnd HyStart у фазі повільного старту.

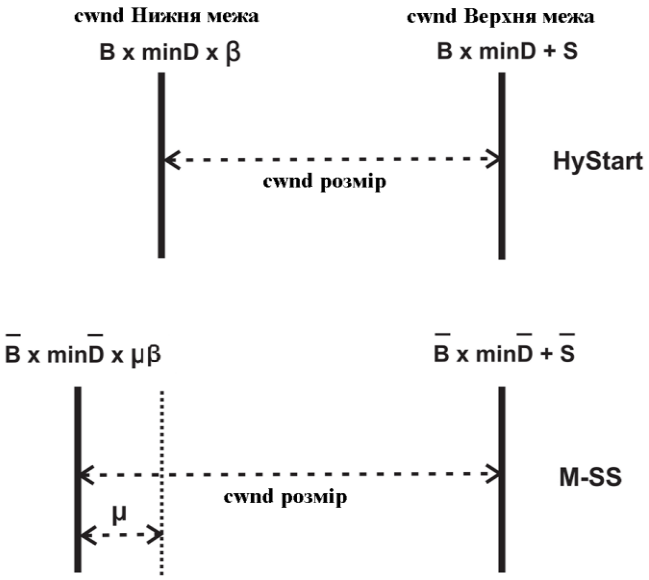

Рисунок 4.8 – Граничні обмеження вікна перевантаження в HyStart і M-SS

HyStart використовує формулу $(B \cdot minD \cdot \beta)$ для обчислення нижньої межі розміру cwnd, де B – доступна пропускна здатність каналу, minD – мінімальна одностороння затримка, а β – параметр зменшення cwnd. B і minD розраховують час роботи за допомогою методів пакетної передачі, і ці значення змінюються відповідно до стану мережі, тоді як β має фіксоване значення, яке дорівнює 0,2. Щоб змінити нижню межу розміру cwnd, необхідно змінити значення β.

Для цього μ множиться на β, експериментально-статистичне значення якого дорівнює 1,5. Таким чином, після множення (μ×β) значення β змінюється від 0,2 до 0,3. Ця зміна значення β оновлює нижню межу розміру

cwnd у фазі повільного старту, яка також оновлює точку виходу або точку завершення фази повільного старту та умовну змінну. Таким чином, M-SS використовує нову формулу ($\overline{B} \cdot \min\overline{D} \cdot \mu\beta$) для розрахунку нижньої межі для розміру cwnd у фазі повільного старту. Змінна μ допомагає нижній межі cwnd мати більше наборів значень для cwnd. Ця зміна нижньої межі спричиняє гнучкість розміру cwnd, що призводить до низької швидкості втрати пакетів під час фази повільного старту.

Блок-схема M-SS показана на рисунку 4.9.

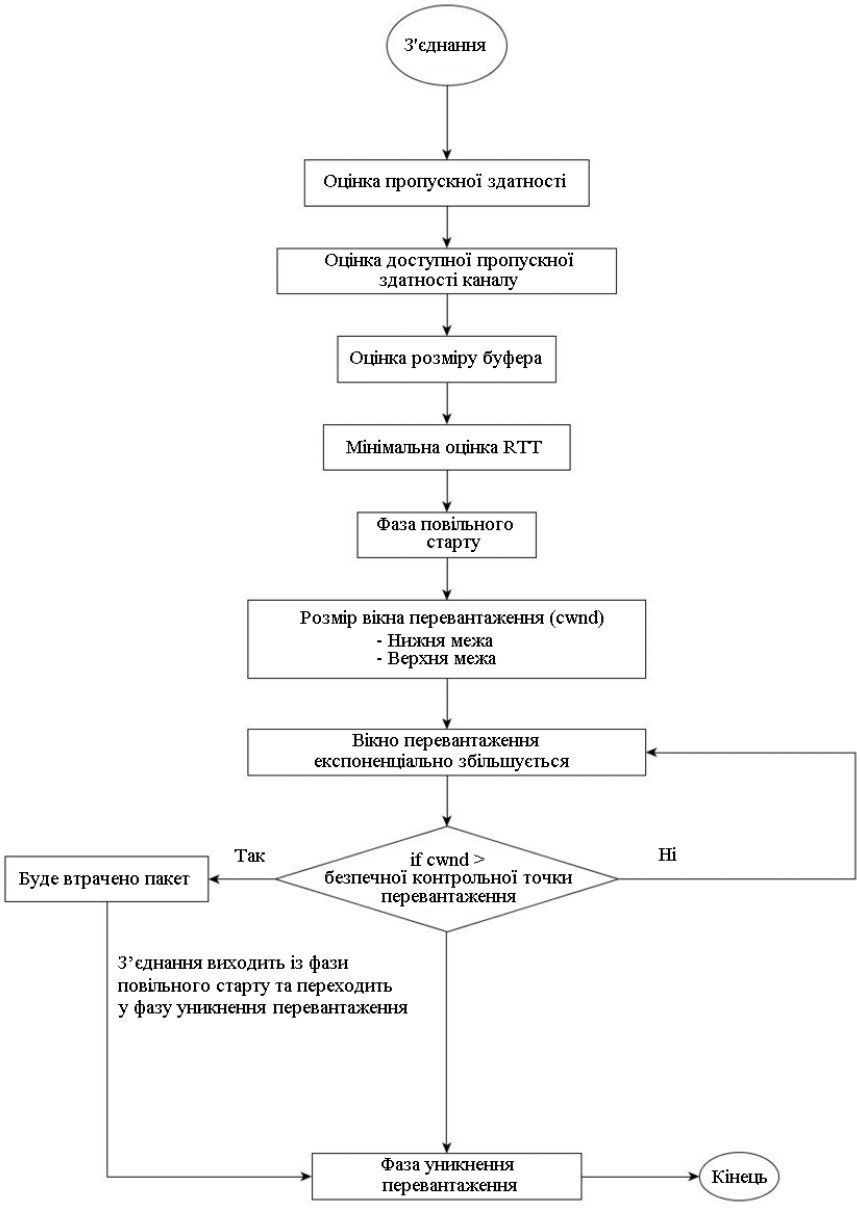

Рисунок 4.9 – Алгоритм контролю перевантаження для повільного старту

Коли зв’язок починається, з’єднання оцінює доступну пропускну здатність каналу та мінімальне RTT. Щоб розпочати зв’язок, з’єднання встановлює початковий розмір cwnd і експоненціально збільшує розмір cwnd, щоб зайняти доступну пропускну здатність каналу.

Коли розмір cwnd стає рівним або більшим за ssthresh, з’єднання припиняє фазу повільного старту та переходить у фазу уникнення перевантаження. Якщо розмір cwnd стає більшим за SCCP, відбудеться втрата пакетів.

Псевдокод M-SS щодо експоненціального зростання при кожному ACK описаний в лістингу 4.1.

Лістинг 4.1 – Псевдокод алгоритму експоненціального зростання cwnd

```
При кожному ACK у M-SS
if dMin then
  dMin←min(dMin, RTT)
else
  dMin←RTT
  if cwnd≤ssthresh then
    cwnd←cwnd+1
  else
    cnt←cwnd+1
    if cwnd_cnt>cnt then
      cwnd←cwnd+1
      cwnd_cnt←0
    else
      cwnd_cnt←cwnd_cnt+1
    end if
  end if
end if
```

cwnd ← cwnd + 1 означає експоненціальне зростання cwnd у фазі повільного старту.

Метод M-SS базується на наступних ключових припущеннях. Розрахункова пропускна здатність каналу визначається сумою доступної пропускної здатності каналу та розміру буферів на вузьких місцях маршрутизаторів.

Розмір cwnd у фазі повільного старту має збільшитися до максимальної верхньої межі, щоб можна було досягти максимального використання каналу, уникаючи максимально можливих втрат.

Доступна пропускна здатність каналу, мінімальна одностороння затримка прямого шляху та розмір буфера можуть бути обчислені за допомогою методів, які використовує HyStart.

Контрольна точка безпечного перевантаження (SCCP) може бути обчислена за допомогою рівняння 4.9.

У фазі повільного старту, якщо розмір cwnd стає більшим за SCCP, виникне перевантаження.

Нижня гранична межа розміру cwnd M-SS встановлена на ($\overline{B} \cdot \min\overline{D} \cdot \overline{\beta}$), де $\overline{\beta}$ дорівнює ($\mu \times \beta$). β є параметром мультиплікативного зменшення cwnd, а μ є змінною M-SS, що має експериментальне та статистичне значення 1,5.

Нижня та верхня межі cwnd розміру CCM-SS у фазі повільного старту наведено в рівняннях 4.19 та 4.20:

$$(\overline{B} \cdot \min\overline{D} \cdot \mu \cdot \beta) < \text{cwnd} < (\overline{B} \cdot \min\overline{D} + \overline{S}). \quad (4.19)$$

Для $\mu = 1{,}5$ і $\beta = 0{,}2$:

$$(\overline{B} \cdot \min\overline{D} \cdot 0.3) < \text{cwnd} < (\overline{B} \cdot \min\overline{D} + \overline{S}). \quad (4.20)$$

Результати M-SS описуються наступним чином:

- HyStart обмежує розмір cwnd нижньою та верхньою межами, а обмеження межі призводить до меншого використання доступної пропускної здатності каналу, що, у свою чергу, спричиняє пакетні втрати пакетів у фазі повільного старту;

- втрата пакетів зменшується завдяки використанню більш гнучкого обмеження розміру cwnd у фазі повільного старту, що також оновлює точку

виходу фази повільного старту та умовну змінну (ця зміна граничного обмеження розміру cwnd також зменшує ефект експоненціального зростання cwnd у фазі повільного старту);

- M-SS забезпечує кращу нижню межу межі для розміру cwnd під час фази повільного старту, що не тільки знижує швидкість втрати пакетів, але і, як додаткові результати, також покращує продуктивність, справедливість протоколу та час збіжності (справедливий та швидкий розподіл доступної смуги пропускання каналу) потоків.

4.7 Результати моделювання

Проводиться порівняння продуктивності TCP CUBIC за допомогою методів повільного старту HyStart і M-SS. Для цього M-SS реалізовано в TCP CUBIC. Таким чином, TCP CUBIC з HyStart і M-SS оцінюється щодо рівня втрати пакетів.

Далі на рисунках показано порівняння продуктивності методів M-SS і HyStart щодо швидкості втрат пакетів у мережах з великим та малим RTT. На рисунку 4.10 показана швидкість втрати пакетів потоків M-SS і HyStart, у мережі із великим RTT, що відноситься до мереж великих відстаней з високою пропускною здатністю.

Рисунок 4.10 ілюструє, що графік методуM-SS нижче, ніж HyStart від 200 до 800 секунд. Таким чином, швидкість втрати пакетів потоків M-SS нижче, ніж у потоків HyStart. M-SS покращує продуктивність з погляду швидкості втрати пакетів у мережах з великим RTT.

На рисунку 4.11 показано порівняння методів M-SS та HyStart у мережі з малим RTT. Графіки M-SS та HyStart дуже близькі один до одного, тому в мережах з малим RTT є невелика різниця між швидкістю втрати пакетів M-SS та HyStart. Проте M-SS покращив свою продуктивність в обох випадках. Таким чином, можна зробити висновок, що метод M-SS маює нижчу швидкість втрати пакетів порівняно з HyStart.

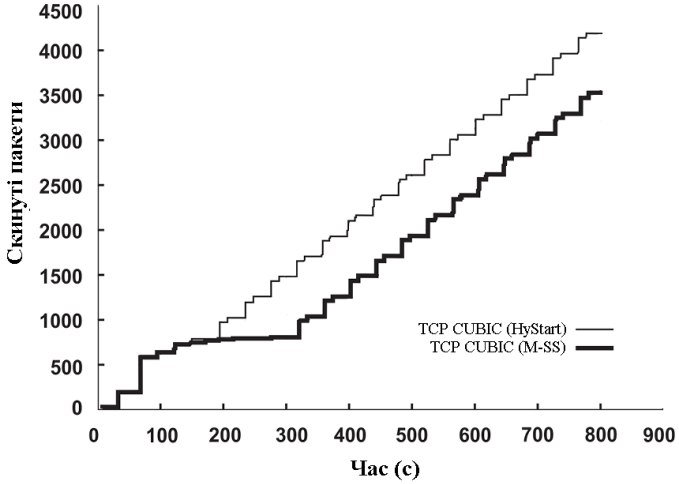

Рисунок 4.10 – Порівняння рівня втрати пакетів для мереж з великим RTT

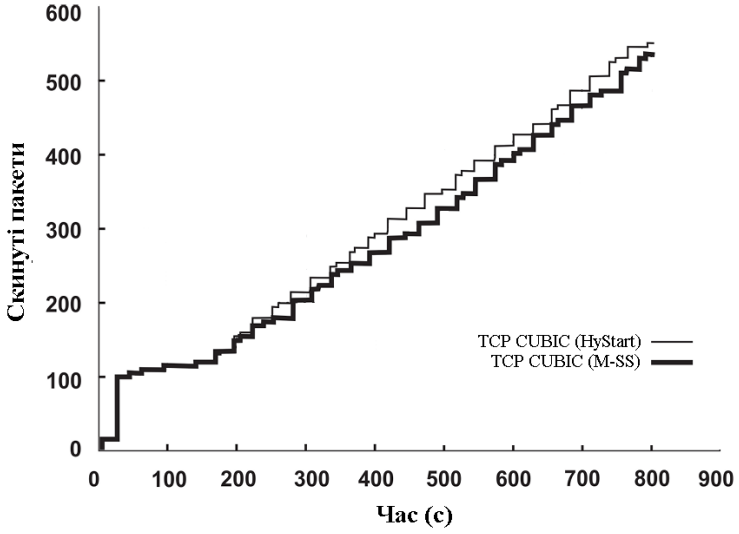

Рисунок 4.11 – Порівняння рівня втрати пакетів для мереж з малим RTT

Швидкість втрати пакетів також впливає на корисну продуктивність потоків, таким чином зменшення швидкості втрати пакетів збільшує корисну продуктивність потоків. Таким чином, потоки, налаштовані M-SS, досягають більш високої корисної продуктивності в порівнянні з потоками HyStart.

На рисунку 4.12 показана ефективна пропускна здатність M-SS і HyStart щодо швидкості каналу в Мбіт/с між вузлами джерела та призначення. Результати показують, що TCP CUBIC показує кращу корисну продуктивність під час використання M-SS проти методу HyStart.

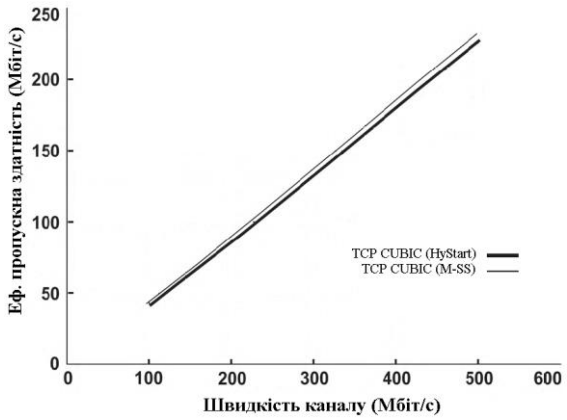

Рисунок 4.12 – Ефективна пропускна здатність M-SS і HyStart відносно швидкості каналу

На рисунку 4.13 показана ефективна пропускна здатність щодо RTT потоків. Коли TCP CUBIC використовує M-SS як метод повільного старту за умовчанням, він досягає більш високої ефективної пропускної здатності порівняно з HyStart, що підтверджує продуктивність M-SS порівняно з HyStart.

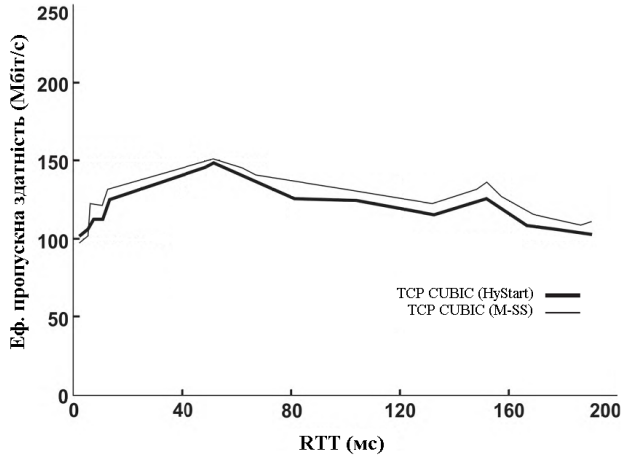

Рисунок 4.13 – Ефективна пропускна здатність M-SS і HyStart відносно RTT потоків

Отримані результати показали, що час конвергенції та справедливість протоколу потоків також покращуються за допомогою методу M-SS.

На рисунку 4.14 показано, що обидва потоки M-SS конвергують один в одного за менший час порівняно з методом HyStart, як показано на рисунку 4.15. Таким чином, потоки M-SS розподіляють доступну пропускну здатність каналу дуже швидко та рівномірно один до одного порівняно з потоками HyStart.

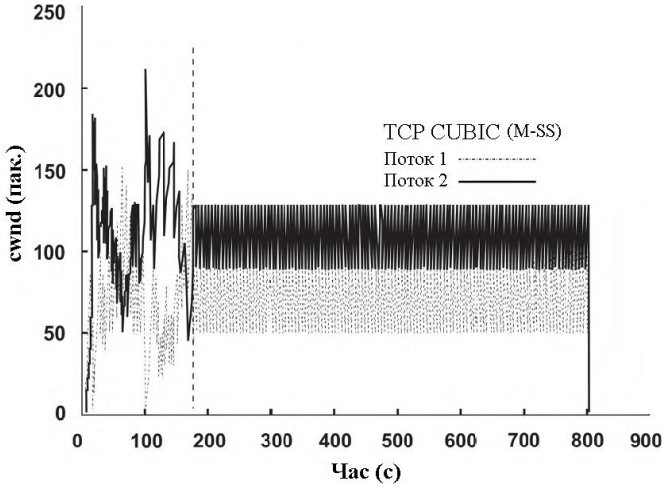

Рисунок 4.14 – Час конвергенції та порівняння справедливості розподілу пропускної здатності TCP CUBIC з M-SS

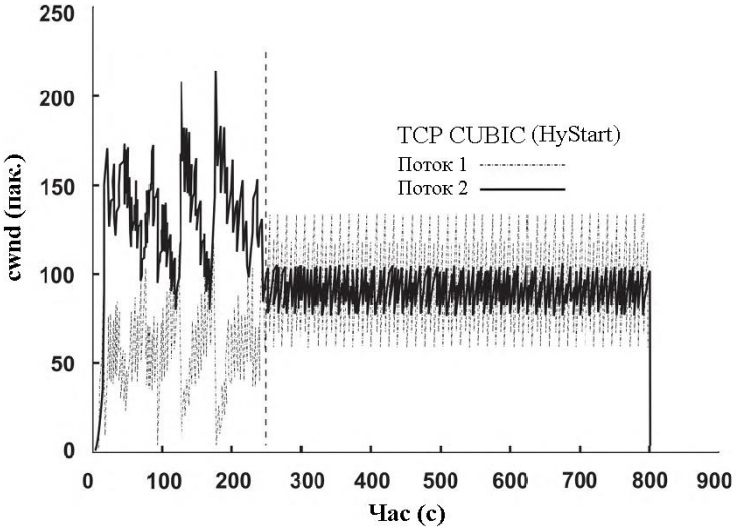

Рисунок 4.14 – Час конвергенції та порівняння справедливості розподілу пропускної здатності TCP CUBIC з M-SS

Таким чином:

- середня швидкість втрати пакетів в методі M-SS менша, ніж у HyStart в мережах великих і малих значень RTT;

- потоки, налаштовані за допомогою M-SS, зменшили рівень втрат пакетів на 10% порівняно з HyStart;

- потоки, встановлені за допомогою методу M-SS, можуть досягти вищої продуктивності порівняно з HyStart;

- зниження рівня втрат пакетів у методі M-SS також покращує час конвергенції та справедливість розподілу пропускної здатності;

- потоки з методом M-SS зближуються дуже швидко один з одним для розділу доступної пропускної здатності каналу в порівнянні з методом HyStart.

ВИСНОВКИ

Перевантаження виникає, коли сукупний попит на ресурси перевищує доступну пропускну здатність мережі. Ця проблема призводить до неприйнятно тривалого часу відгуку, особливо для програм реального часу. Експоненціальне зростання вікна перевантаження у фазі повільного старту TCP спричиняє втрати пакетів, і потоки TCP не розподіляють доступну пропускну здатність мережевого каналу справедливо.

Кваліфікаційна робота присвячена підвищення продуктивності та ефективності протоколу TCP для захищених мереж зв’язку з високою пропускною здатністю.

В кваліфікаційній роботі пропонується метод контролю перевантажень для фази повільного старту, який зменшує ефект експоненціального зростання перевантажувального вікна шляхом застосування нових обмежень у фазі повільного старту, що у свою чергу, зменшує швидкість втрати пакетів у високошвидкісних мережах та дозволяє досягти більшої мережевої продуктивності.

Проведене імітаційне моделювання підтвердило вірність теоретичних викладок, представлених в кваліфікаційній роботі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. J. Postel, “Transmission Control Protocol,” IETF RFC 793, September 1981.

2. J. Nagle, “Congestion control in IP/TCP Internetworks,”Request for Comments (RFC) 896, Internet EngineeringTask Force, January 1984.

3. V. Jacobson, “Congestion Avoidance and Control,” ACM SIGCOMM Computer Communication Review, Vol. 18, No. 4, pp. 314-329, August 1988.

4. V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3 Reno," Proceedings of the 18th Internet Engineering Task Force, University of British Columbia, Vancouver, BC, Aug. 1990.

5. Forouzan, B. (2001). Data Communications and Networking (2nd ed.). New York, NY, USA: McGraw-Hill.

6. Xu, W., Xu, Y., Wu, X., & Ou, K. (2011, Jan). Modeling TCP sack steady state performance in lossy networks. In Proceedings of the International Conference on Information Networking (ICOIN). (p. 278-283). doi: 10.1109/ICOIN.2011.5723193

7. Nagle, J. (1984, January). Congestion Control in IP/TCP Internetworks (No. 896). RFC 896. IETF. Retrieved from http://www.ietf.org/rfc/rfc896.txt

8. Arpaci, M., & Copeland, J. (2000). An adaptive queue management method for congestion avoidance in TCP/IP networks. In Proceedings of the IEEE Conference on Global Telecommunications (GLOBECOM). (Vol. 1, p. 309-315).

9. Sikdar, B., Kalyanaraman, S., & Vastola, K. (2003, Dec). Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK. Journal of IEEE/ACM Transactions on Networking, 11(6), 959-971.

10. Waghmare, S., Parab, A., Nikose, P., & Bhosale, S. J. (2011, April). Comparative analysis of different TCP variants in a wireless environment. In Proceedings of the 3rd International Conference on Electronics Computer Technology (ICECT). (Vol. 4, p. 158-162).

11. Cong, L., & Miki, T. (2000). New acknowledgement mechanism for TCP/IP over ATM. In Proceedings of the International Conference on Communication Technology (ICCT). (Vol. 1, p. 436-443).

12. Jacobson, V. (1988). Congestion avoidance and control. In ACM SIGCOMM Computer Communication Review, vol. 18. ACM, 314- 329

13. Floyd, S.; Henderson, T. & Gurtov, A. The NewReno modification to TCP's fast recovery algorithm RFC 2582, April, 1999

14. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanov, RFC2018TCP selective acknowledgment options RFC, 1996.

15. Xu, L., Harfoush, K. and Rhee, I. (2004). Binary increase congestion control (BIC) for fast long-distance networks. In INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 4. IEEE, 2514-2524.

16. Ha, S., Rhee, I. and Xu, L. (2008). CUBIC: A new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review. 42(5), 64-74.

17. Brakmo, L. and Peterson, L. (1995). TCP Vegas: End to end congestion avoidance on a global Internet. Selected Areas in Communications, IEEE Journal on. 13(8), 1465-1480.

18. Song, K., Zhang, Q. and Sridharan, M. (2006). Compound TCP: A scalable and TCPfriendly congestion control for high-speed networks. Proceedings of PFLDnet 2006.

19. S. Ha and I. Rhee, ''Taming the elephants: New TCP slow start,'' Comput. Netw., vol. 55, no. 9, pp. 2092–2110, 2011.

20. G. A. Abed, M. Ismail, and K. Jumari, ''Exploration and evaluation of tra- ditional TCP congestion control techniques,'' J. King Saud Univ.-Comput. Inf. Sci., vol. 24, no. 2, pp. 145–155, 2012.

21. R. Wang, G. Pau, K. Yamada, M. Sanadidi, and M. Gerla, ''TCP startup performance in large bandwidth networks,'' in Proc. 23rd Annu. Joint Conf. IEEE Comput. Commun. Soc. (INFOCOM), vol. 2. Mar. 2004, pp. 796–805.

22. N. Hu and P. Steenkiste, ''Improving TCP startup performance using

active measurements: Algorithm and evaluation,'' in Proc. 11th IEEE Int. Conf. Netw. Protocols, Nov. 2003, pp. 107–118.

23. C. Ho, Y. Chan, and Y. Chen, ''Gallop-Vegas: An enhanced slow-start mechanism for TCP Vegas,'' J. Commun. Netw., vol. 8, no. 3, pp. 351–422, 2006.

24. R. Jain, D.-M. Chiu, and W. Hawe, ''A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,''in Proc. Conf. Eastern Res. Lab., Digit. Equip. Corp. Hudson, 1998, pp. 141–167.

25. S. Ha, Y. Kim, L. Le, I. Rhee, and L. Xu, ''A step toward realistic performance evaluation of high-speed TCP variants,'' in Proc. 4th Int. Workshop Protocols Fast Long-Distance Netw. (PFLDnet), vol. 35. 2006, pp. 331–343.

26. G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, and M. Roccetti, ''TCP libra: Derivation, analysis, and comparison with other rtt-fair TCPs,'' Comput. Netw., vol. 54, no. 14, pp. 2327–2344, 2010.

27. Сидоренко Р.Ю., Козін М.В., Янковський О.А. Боротьба з перевантаженнями в IP-мережах//Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. Чотирнадцята міжнародна науково-технічна конференція. Баку – Харків – Жиліна. 2024 р. Том 1. С. 116.

28. Козін М.В., Сокирко М.А., Янковський О.А. Методи адаптації протоколу TCP до поточного мережевого стану//Збірник тез доповідей дванадцятої міжнародної науково-технічної конференції «Проблеми інформатизації». Баку, Харків, Бельсько-Бяла. 2024. С. 72.