Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Програмно-алгоритмічні методи боротьби з перевантаженнями в комп'ютерних мережах

(тема)

Виконав:
студент II курсу, групи КСМм-22-1
Нечипоренко А.О.
(прізвище, ініціали)

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

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

Освітня програма
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)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

Тип програми освітньо-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 Нечипоренку Артему Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмно-алгоритмічні методи боротьби з перевантаженнями в комп'ютерних мережах

затверджена наказом по університету від "6" листопада 2023 р. № 1298 Ст

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

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

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

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайдів презентації – 17 шт.

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз стану проблеми та сучасних методів її	07.11.23 – 09.11.23	
	вирішення		
2	Огляд технологій управління перевантаженням	10.11.23 – 14.11.23	
3	Розробка моделі управління мережним	15.11.23 – 11.12.23	
	трафіком		
4	Вибір програмних та апаратних засобів	12.12.23 – 20.12.23	
	реалізації		
5	Тестування запропонованого метода	21.12.23 – 25.12.23	
6	Оформлення пояснювальної записки	26.12.23 – 12.01.24	

Дата видачі завдання 06 листопада 2023 р..

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 77 с., 19 рис., 1 дод., 22 джерела.

RTT, TCP, КАНАЛ, ПЕРЕВАНТАЖЕННЯ, ПРОПУСКНА ЗДАТНІСТЬ, СКИДАННЯ ПАКЕТІВ, ТАЙМ-АУТ ПЕРЕДАЧІ.

В кваліфікаційній роботі зосереджено увагу на механізмах, які можуть вирішити ключові проблеми продуктивності TCP без будь-яких модифікацій самого протоколу. Цей еволюційний підхід важливий на практиці, оскільки було доведено, що розгортання чистих транспортних протоколів в Інтернеті є надзвичайно складним.

Зокрема, визначено низку проблем, пов’язаних із TCP, які можуть спричинити низьку наскрізну продуктивність. Ці проблеми включають погано визначені розміри буфера на кінцевих хостах, неоптимальний розмір буфера на маршрутизаторах і комутаторах, а також накопичення трафіку TCP, що не реагує на перевантаження. Пропонуються рішення, які можуть вирішити ці проблеми без будь-яких змін у TCP.

ABSTRACT

Master’s thesis: 77 pages, 19 figures, 1 appendices, 22 sources.

RTT, TCP, CHANNEL, OVERLOAD, THROUGHT CAPACITY, PACKET DROP, TRANSMISSION TIMEOUT.

The qualification work focuses on mechanisms that can solve the key performance problems of TCP without any modification of the protocol itself. This evolutionary approach is important in practice because it has been proven that deploying clean transport protocols on the Internet is extremely difficult.

In particular, a number of TCP-related issues have been identified that can cause poor end-to-end performance. These problems include poorly defined buffer sizes on end hosts, suboptimal buffer sizes on routers and switches, and unresponsive TCP traffic build-up to congestion. Solutions are proposed that can solve these problems without any changes to TCP.

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 Протокол TCP 12
2.2 Особливості протоколу TCP 17
3 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ 20
3.1 Модифікації контролю перевантаження TCP 20
3.2 Методи визначення розміру буфера кінцевих вузлів 22
3.3 Розмір буфера маршрутизатора 23
3.4 Реагування на перевантаження та процес надходження потоку 25
4 ПОПЕРЕДЖЕННЯ ПЕРЕВАНТАЖЕНЬ НА ОСНОВІ ВТРАТ І ЗАТРИМКИ 28
4.1 LCA у трактах з високою пропускною здатністю та затримкою 29
4.2 Проблеми для DCA 32
4.2.1 Максимальна варіація затримки та розмір мережевого буфера 33
4.2.2 Частота дискретизації RTT 34
4.2.3 Ступінь мультиплексування 35
4.2.4 Наявність випадкових втрат 36
5 АВТОМАТИЧНЕ ВИЗНАЧЕННЯ РОЗМІРУ БУФЕРА 38
5.1 Розмір буфера сокета та пропускна здатність TCP 41
5.2 Максимально можлива пропускна здатність і доступна пропускна здатність 44
5.2.1 Шляхи без заторів 44
5.2.2 Перевантажені шляхи 48
6 АВТОМАТИЧНЕ ВИЗНАЧЕННЯ РОЗМІРУ БУФЕРА СОКЕТУ 51

6.1 Основна ідея 52
6.2 Деталі реалізації та діаграма стану 52
6.3 Затримка переповнення буфера 56
6.4 Експериментальні результати 59
ВИСНОВКИ 65
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 66
ДОДАТОК А Графічний матеріал кваліфікаційної роботи 68

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)

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

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

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

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

ВСТУП

Незважаючи на постійні зміни та зростання, які характеризують Інтернет, протокол керування передачею (TCP) все ще домінує на транспортному рівні, передаючи понад 95% глобального трафіку. Незважаючи на приголомшливий успіх, було помічено, що TCP може спричинити низьку наскрізну продуктивність, особливо для тривалих з'єднань і мережевих шляхів із високою затримкою в каналах.

У мережевих шляхах зі значною доступною пропускною здатністю збільшення вікна TCP до спостережуваних втрат може призвести до значно нижчої пропускної здатності, ніж доступна пропускна здатність шляху.

Неправильний розмір буфера маршрутизатора або кінцевих систем також може призвести до низької пропускної здатності TCP. Попередні дослідження визначення розміру буфера маршрутизатора були зосереджені на показниках продуктивності мережі, таких як використання каналу або рівень втрат. Замість цього слід зосередитись на впливі розміру буфера на продуктивність наскрізного протоколу TCP.

Розмір буфера маршрутизатора, який оптимізує пропускну здатність TCP, значною мірою визначається співвідношенням вихідної та вхідної ємності каналу.

Якщо це співвідношення більше одиниці, рівень втрат експоненціально падає зі збільшенням розміру буфера, і оптимальний розмір буфера близький до нуля.

В іншому випадку, якщо відношення вихідної та вхідної ємності менше одиниці, коефіцієнт втрат відповідає степеневому зменшенню розміру буфера, і потрібна значна буферизація.

Обсяг буферизації, необхідний у цьому випадку, залежить від того, чи закінчується більшість потоків у фазі повільного запуску чи у фазі уникнення перевантаження.

Пропускна здатність TCP також залежить від того, чи зменшує перехресний трафік швидкість пересилання через перевантаження. Властивість перехресного руху це реакція на затори. Оскільки більшість Інтернет-трафіку використовує протокол TCP, який зменшує швидкість надсилання під час перевантаження, вважається, що сукупність багатьох потоків TCP реагує на перевантаження.

Слід зазначити, що що реакція на затори сукупного трафіку також залежить від процесу прибуття потоку. Якщо процес надходження потоку відбувається за моделлю відкритого циклу, тоді навіть якщо трафік складається виключно з передач TCP, сукупний трафік все одно може не реагувати на перевантаження.

З іншого боку, потоки TCP, які надходять у мережу за замкнутим циклом, завжди реагують на перевантаження. Потрібна схема для оцінки частки трафіку, який відповідає моделі замкнутого циклу в заданому каналі зв’язку, а також необхідно збільшення цієї частки за допомогою простих модифікацій на прикладному рівні.

1 МЕТА КВАЛІФІКАЦІЙНОЇ РОБОТИ

TCP часто звинувачують у низькій наскрізній продуктивності в Інтернеті. Поява високошвидкісних мереж в останні роки підкреслила неефективність TCP. Хоча було зроблено багато пропозицій щодо вдосконалення/заміни TCP, мало було досягнуто згоди щодо відповідної заміни.

Необхідно розробити більш детальні моделі поведінки користувачів, щоб розробити кращі інструменти для прогнозування характеристик трафіку та його реакції на умови перевантаження.

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

- провести всебічний аналіз сучасних літературних джерел, повязаних з проблемами підвищення пропускної здатності комп'ютерних мереж;
- провести огляд існуючих моделей та методів керування потоками інформації;
- запропонувати метод управління мережевим трафіком;
- за допомогою імітаційного моделювання підтвердити запропоновані в кваліфікаційній роботі теоретичні викладки;
- проаналізувати отримані результати.

2 КОРОТКИЙ ОГЛЯД СТАНУ ПРОБЛЕМИ

Інтернет постійно розвивався з моменту свого створення як ARPAnet. Він вирос з мережі з кількох сотень хостів до мільйонів хостів, які підтримують постійно зростаючу кількість різноманітних програм і послуг.

Незважаючи на всі зміни в Інтернеті, протокол керування передачею (TCP) залишається найбільш широко використовуваним транспортним протоколом, передаючи понад 90% трафіку [1]. TCP використовується в широкому діапазоні програм і мережевих середовищ.

Наприклад, діапазон додатків, які використовують TCP, включає чутливі до затримки інтерактивні додатки, чутливі до пропускної здатності масові передачі та чутливі до затримки мультимедійні додатки. TCP використовується як у провідних, так і в безпровідних мережах із пропускною спроможністю від кількох Кбіт/с до кількох десятків Гбіт/с і затримками розповсюдження від кількох мілісекунд до кількох секунд.

2.1 Протокол TCP

Хоча TCP є основним механізмом в Інтернеті для управління ресурсами та спільного використання, це не найкращий протокол для всіх програм і середовищ, які використовують TCP.

Наприклад, на шляху високої пропускної здатності із затримкою, програма масової передачі може отримати значно нижчу пропускну здатність, ніж доступна пропускна здатність шляху, через зменшення вдвічі та подальше лінійне збільшення вікна перевантаження TCP після відкидання одного пакета. Проведені дослідження показали, що в Інтернеті 50% «масових» TCP-передач (тобто передачі понад 10 МБ) отримують пропускну здатність менше 5 Мбіт/с [2]. Деякі дослідницькі роботи намагалися виявити різні причини, які обмежують пропускну здатність передачі TCP [3].

Нещодавно кілька дослідників запропонували замінити TCP транспортними протоколами «з чистого листа» та схемами контролю перевантажень.

Однак ці пропозиції не мали успіху з точки зору розгортання, принаймні поки що. Деякі можливі причини.

1. Витрати на заміну застарілих інфраструктур. Деякі із запропонованих чистих рішень вимагають модифікації та/або заміни існуючої мережевої інфраструктури з точки зору програмного та апаратного забезпечення. Вартість таких змін може бути непомірно високою. Їх особливо важче досягти, враховуючи, що переваг від таких змін ще немає.

2. Труднощі з пошуком відповідної заміни TCP: з багатьма різними доступними альтернативами неясно, яка з них є найбільш прийнятною заміною. В основному це пов’язано з тим, що дослідження не змогли дійти загальноприйнятного висновку. Крім того, відсутність реалістичної моделі для Інтернет-трафіку ускладнює проведення дуже великомасштабних експериментів. Отже, продуктивність будь-якого нового транспортного протоколу можна перевірити лише після його повного розгортання, що робить це замкнутим циклом.

3. Співіснування різних версій TCP: також незрозуміло, як ці нові протоколи взаємодіють один з одним, якщо їм доведеться співіснувати. Більшість робіт, що оцінюють удосконалення TCP, припускають, що або існує лише новий протокол, або частина потоків використовує новий протокол, тоді як решта все ще використовує поточний TCP.

4. Розподілене адміністрування та контроль: труднощі з розгортанням нового протоколу ускладнюються децентралізованою природою сьогоднішнього адміністрування Інтернету, коли різні автономні об’єкти контролюють різні частини мережі.

Оскільки складність розгортання будь-яких змін залежить від того, де вони потрібні, використовується місце розгортання як критерій для класифікації різних пропозицій. Найпростішими з модифікацій є ті, що

вимагають змін лише на прикладному рівні. Такі зміни найпростіше впровадити, оскільки вони не вимагають жодних змін у мережевій інфраструктурі чи стандартах.

Друга категорія модифікацій вимагає змін на транспортному рівні. Їх важче розгорнути, ніж зміна додатків, оскільки вони передбачають оновлення стандартів і специфікацій протоколів. Крім того, такі зміни мають задовольняти суперечливі вимоги різних додатків. Цей клас модифікацій можна додатково розділити на ті, що вимагають змін лише на одному кінці, відправнику чи одержувачу. Прикладами цього є NewReno, HSTCP, STCP, оскільки вони прагнуть змінити поведінку TCP на стороні відправника. TCP-SACK є прикладом зміни поведінки протоколу в обох кінцях. Нарешті, деякі пропозиції також можуть потребувати змін на мережевому рівні. Початкову роботу з пошуку зворотного зв’язку щодо перевантажень від мережі можна простежити до ATM-ABR. Останні пропозиції щодо пошуку модифікацій на мережевому рівні для підвищення наскрізної продуктивності включають XCP [4] і RCP [5]. Якщо не оновити всі маршрутизатори та комутатори на шляху, такі зміни або не працюватимуть, або не працюватимуть належним чином.

Крім того, ці пропозиції можна класифікувати як еволюційні підходи та підходи з чистого аркуша. У еволюційному підході покращення можуть бути зроблені шляхом локальних змін і без потреби змін у всій мережі. Вони також повинні співіснувати з іншими підходами, існуючими чи новими, не будучи несправедливими до них. З іншого боку, у підході з чистого аркуша, зміни в усій мережі є прийнятними або навіть необхідними для функціонування.

Значні покращення пропускної здатності TCP можна отримати без зміни протоколу TCP або його реалізації. Зокрема, в кваліфікаційній роботі представлено метод прикладного рівня, який покращує продуктивність TCP і потребує модифікацій лише на стороні адресату. Застосовується простий і важливий параметр для визначення розміру буфера маршрутизатора, який

максимізує середню пропускну здатність TCP на потік. Показано важливість циклу зворотного зв’язку в процесі генерації сеансу для Інтернет-трафіку та розробляється методологія вимірювання реагування на перевантаження сукупного трафіку на каналі зв’язку. Сліпе дослідження TCP більшої доступної смуги пропускання може призвести до втрати пакетів на неперевантаженому шляху. Ці втрати називаються власними втратами. Такі втрати, у свою чергу, можуть призвести до зниження пропускної здатності для потоку TCP і зниження використання всього шляху.

У цій роботі розробляється механізм прикладного рівня, який уникає такої ситуації. Цей механізм, званий авторозміром буфера (АРБ), запобігає виходу протоколу TCP за межі доступної смуги пропускання шляху, обмежуючи розмір буфера сокета в неперевантаженому шляху. У перевантажених шляхах відкидання пакетів відбувається незалежно від «зондування» будь-яким конкретним потоком. Таким чином, немає жодного стимулу для обмеження розміру буфера сокета в таких шляхах.

АРБ використовує швидкість отримання пакету та оцінку RTT, щоб визначити, чи працює він у перевантаженому каналі чи ні, і яким має бути розмір буфера в останньому випадку. Проведені експерименти на шляхах із високою пропускною здатністю показують, що АРБ може забезпечити на 20–80% вищу пропускну здатність порівняно з передачами TCP, які не обмежують розмір буфера сокета. Він покращує пропускну здатність TCP у неперевантажених шляхах і дозволяє TCP отримати справедливу частку в перевантажених шляхах.

Попередні дослідження визначення розміру буфера маршрутизатора були зосереджені на показниках продуктивності мережі, таких як використання каналу або рівень втрат [6]. Ці роботи часто використовують відкритий трафік або аналізують лише постійні потоки TCP. В кваліфікаційній роботі увага зосереджується на впливі розміру буфера на наскрізну продуктивність TCP, розглядаючи при цьому більш реалістичний випадок непостійних потоків TCP із розподілом розміру з важкими хвостами.

Було встановлено, що розмір буфера маршрутизатора, який оптимізує пропускну здатність TCP, значною мірою визначається співвідношенням вихідної та вхідної ємності каналу. Якщо це співвідношення більше одиниці, рівень втрат експоненціально падає зі збільшенням розміру буфера, і оптимальний розмір буфера близький до нуля. В іншому випадку, якщо відношення вихідної та вхідної ємності менше одиниці, коефіцієнт втрат відповідає степеневому зменшенню розміру буфера, і необхідна значна буферизація. Обсяг буферизації, необхідний у цьому випадку, залежить від того, чи закінчується більшість потоків у фазі повільного запуску чи у фазі уникнення перевантаження.

Пропускна здатність TCP-потоку також залежить від того, чи зменшує перехресний трафік швидкість пересилання через місце перевантаження. Ця властивість перехресного руху називається реакцією на затори. Загальноприйнята думка полягає в тому, що якщо окремі передачі реагують на перевантаження так, як це робить TCP, то сукупний трафік у каналі також буде реагувати на перевантаження.

Отже, були пропозиції щодо застосування подібних алгоритмів контролю перевантажень у протоколах, що не є TCP (наприклад, TCP-Friendly control [7] або TCP-тунелі [8]). Це є результатом використання постійних з’єднань TCP, тобто передач, які мають нескінченну кількість даних для надсилання та які тривають нескінченно довго, як основної моделі мережевого трафіку. Коли розглядаються всі потоки TCP як непостійні (тобто з кінцевим розміром і тривалістю), тоді ключовою проблемою є випадковий процес, який визначає надходження потоків, а не пакетів, у мережу. Розглядаються дві моделі генерації потоку.

По-перше, модель замкнутого циклу (CL), де кожен користувач із певної популяції може створити новий потік лише після завершення попереднього сеансу. По-друге, модель відкритого циклу (OL), де потоки надходять незалежно від попередніх потоків. Ці дві моделі створюють трафік із дуже різними агрегатами трафіку, навіть якщо кожен потік контролюється

TCP. Показано, що модель замкнутого циклу створює трафік, що реагує на затори, тоді як модель із відкритим циклом призводить до трафіку, що не реагує.

2.2 Особливості протоколу TCP

TCP працює поверх IP, забезпечуючи надійний зв’язок у сучасному Інтернеті. TCP є надійним через його основний компонент, який є контролем перевантаження, який відповідає за виявлення та реагування на перевантажений трафік в Інтернеті. TCP все ще вимагає високої продуктивності, запобігаючи збою перевантаження.

Мережі великої пропускної спроможності швидко охоплюють кілька континентів, і TCP широко використовується як основний протокол передачі в цих мережах. Продуктивність TCP є однією з головних критичних проблем у мережах великої відстані, оскільки TCP не використовує максимальну пропускну здатність. Продуктивність TCP залежить від трьох основних компонентів: погане виявлення втрат пакетів, коефіцієнтів вікна перевантаження (cwnd) до або після втрати та збільшення cwnd на початку з’єднання. Невеликий адитивний ріст cwnd, який використовується для контролю перевантажень TCP, звинувачують у його низькій продуктивності в цих мережах. Було багато вдосконалених механізмів контролю перевантажень TCP, пропонуються застосування більш масштабованої функції збільшення вікна cwnd для кращої продуктивності в мережах з високою пропускною спроможністю та на великі відстані. Більшість цих нових механізмів контролю перевантажень лише змінюють поведінку протоколу під час фази уникнення перевантажень. Для міжміських мереж із високою пропускною спроможністю більшість існуючих механізмів контролю перевантажень TCP не оптимізовані, оскільки, коли вони працюють у таких середовищах, вони потрапляють у деякі рідкісні стани, коли TCP отримує надзвичайно низьку продуктивність.

Щоб уникнути збою перевантаження, контроль перевантаження TCP дотримується принципу пакетної передачі, який підтверджує доставку переданого пакету за допомогою підтвердження (ACK). Для кожного пакета, надісланого в мережу джерелом, очікується, що ACK буде передано назад від пункту призначення. Джерело контролює швидкість надсилання пакетів за допомогою змінної під назвою вікно перевантаження cwnd, яка визначає кількість пакетів, які джерело може надсилати.

Пункт призначення також повідомляє джерелу обсяг даних, який він готовий буферизувати для з'єднання, яке називається оголошеним 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 2.1 показано типову поведінку контролю перевантаження TCP.

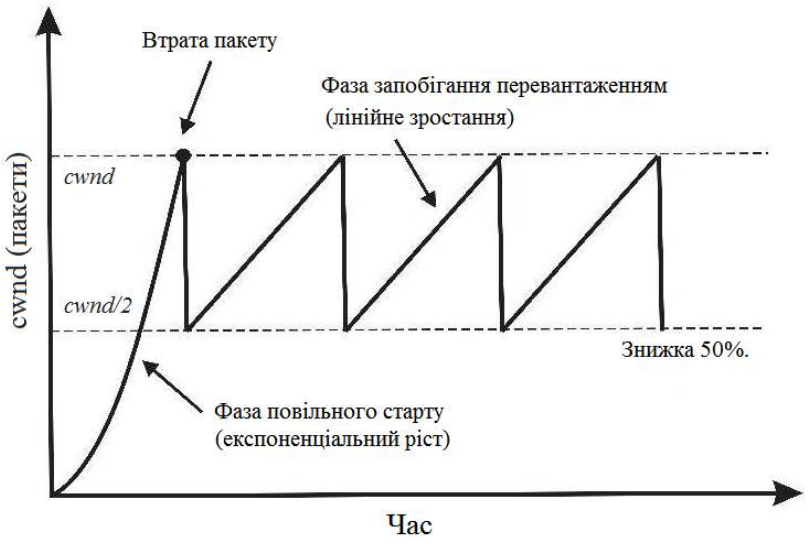

Рисунок 2.1 – Динаміка контролю перевантаження TCP

Контроль заторів TCP складається з чотирьох основних компонентів: повільний старт і уникнення перевантажень, швидкої повторної передачі та швидкого відновлення. Повільний старт і алгоритми уникнення

перевантажень контролюють передачі. Поріг повільного запуску (ssthresh) використовується для визначення того, який алгоритм, повільний запуск або уникнення перевантаження використовуються TCP для керування передачею даних. Якщо величина cwnd менша за ssthresh, то використовується алгоритм повільного запуску, а якщо cwnd більше або дорівнює ssthresh, то використовується алгоритм уникнення перевантажень.

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 стану рівноваги 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 запуску 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.

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 запуску та переходить у фазу уникнення перевантаження. ssthresh, що є оціночним консервативним показником доступної пропускної здатності каналу в мережевому шляху.

3 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ

Пропозиції щодо модифікації TCP варіюються від налаштування/зміни параметрів AIMD TCP до заміни механізму контролю перевантажень і вимагають співпраці від кожного маршрутизатора на шляху.

Більшість попередніх робіт зі зміни розміру буфера робилися спроби встановити розмір буфера відповідно до деякого поняття додатку затримки смуги пропускання. Вони відрізняються своєю інтерпретацією та оцінкою пропускної здатності та затримки. Ці роботи намагаються виділити довільно фіксований розмір буфера, наданий для всіх з'єднань за стандартним значенням з операційної системи. Однак вони не намагаються визначити стан мережі, пов'язуючи його з розміром буфера, який буде найбільш сприятливим для даного з'єднання.

Попередня робота над забезпеченням буфера маршрутизатора була зосереджена на мережевих матрицях продуктивності, таких як використання каналу зв'язку та швидкість відкидання пакетів. Зокрема, вони намагаються відповісти на питання: «Який мінімальний розмір буфера на даному маршрутизаторі максимізує використання каналу зв'язку, утримуючи рівень втрат нижче прийнятного рівня?», не беручи до уваги його вплив на наскрізну продуктивність. Більшість попередніх робіт у сфері реагування на перевантаження розглядають усі потоки TCP як реагуючий трафік і пропонують способи зробити трафік не-TCP чуйним за допомогою тунелювання TCP, обмеження швидкості для TCP тощо.

3.1 Модифікації контролю перевантаження TCP

Кілька дослідницьких робіт запропонували модифікації TCP, здебільшого зосереджені на алгоритмі контролю перевантаження, спрямовані на підвищення ефективності TCP у шляхах з високою пропускною здатністю.

Флойд запропонував високошвидкісний TCP [9], у якому коефіцієнти збільшення та зменшення вікна залежать від поточного вікна перевантаження. Ці фактори вибрано таким чином, щоб масова передача TCP могла наситити навіть шляхи з дуже високою пропускною здатністю в мережах із втратами даних.

З подібними цілями Келлі запропонував масштабований TCP [10]. Важливою відмінністю є те, що Scalable TCP використовує постійні коефіцієнти збільшення та зменшення вікна, а також правило мультиплікативного збільшення, коли немає перевантаження. З останньою модифікацією Scalable TCP відновлюється від втрат набагато швидше, ніж TCP Reno. Сю запропонований BIC-TCP, який виконує двійковий пошук відповідного розміру вікна перевантаження [11].

Деякі запропоновані протоколи заміни TCP вимагають співпраці з маршрутизаторами на шляху. Катабі запропонував новий транспортний протокол eXplicit Control Protocol (XCP) [4]. XCP вимагає, щоб кожен маршрутизатор повідомляв джерела XCP про обсяг доступної їм пропускної здатності. Дуккіпаті запропонований протокол керування швидкістю (RCS), де кожен маршрутизатор забезпечує постійну швидкість для всіх активних потоків [5].

TCP Westwood використовує оцінку пропускної здатності, отриману з дисперсії ACK передачі, щоб встановити вікно перевантаження після події втрати [99]. Вествуд ввів концепцію «прийнятної ставки», яка є оцінкою справедливої частки TCP.

Іншим варіантом TCP, який орієнтований на шляхи з високою пропускною здатністю, є TCP FAST. FAST має деякі важливі подібності з TCP Vegas [12]. Ключова ідея полягає в тому, щоб обмежити вікно надсилання переказу, коли RTT починають зростати. За принципом це схоже на АРБ, тобто FAST і Vegas також прагнуть наситити доступну смугу пропускання на шляху. Важливою відмінністю є те, що АРБ відключається на перевантажених шляхах, стаючи таким же агресивним, як з’єднання Reno.

Відомо, з іншого боку, що Vegas менш агресивний, ніж Reno на перевантажених шляхах [2].

Нещодавно було запропоновано варіант TCP, у якому вікно надсилання регулюється на основі доступної пропускної здатності шляху. Пропонований протокол називається TCP-Low Priority (TCP-LP). Незважаючи на те, що TCP-LP не є схемою визначення розміру буфера, він схожий на АРБ у тому сенсі, що має на меті охопити лише доступну пропускну здатність. Основна відмінність від TCP-LP полягає в тому, що АРБ відключається в перевантажених шляхах з невеликою доступною пропускною здатністю або без неї, і тому це призведе до вищої пропускної здатності, ніж TCP-LP у таких шляхах. Крім того, TCP-LP зменшує вікно надсилання кожного разу, коли RTT показують тенденцію до зростання; така поведінка призведе до нижчої пропускної здатності, ніж АРБ, навіть у неперевантажених каналах.

3.2 Методи визначення розміру буфера кінцевих вузлів

Технологія автоматичного налаштування, яка базується на оцінці активної смуги пропускання, це Work Around Daemon (WAD). WAD використовує ping для вимірювання мінімального RTT T_m перед початком з'єднання TCP і pipechar для оцінки пропускної здатності C шляху. Подібний підхід використовується реалізацією NLANR Auto-Tuning FTP.

Автоматичне налаштування TCP-буфера вперше було запропоновано Семке [13]. Мета цієї роботи полягала в тому, щоб дозволити хосту (як правило, серверу) справедливо розподіляти пам'ять ядра між декількома поточними з'єднаннями. Важливим моментом у цій роботі є те, що продукт пропускної здатності затримки (BDP) шляху оцінювався на основі вікна перевантаження (cwnd) TCP-з'єднання. Для розміру буфера приймального адресата було встановлено достатньо велике значення, щоб воно не обмежувало пропускну здатність передачі.

Технологія автоматичного налаштування буфера на основі програми, яка називається Dynamic Right Sizing (DRS), була запропонована в [14]. DRS вимірює RTT шляху до початку з'єднання. Щоб оцінити пропускну здатність шляху, DRS вимірює середню пропускну здатність на приймальній стороні. Однак важливо зазначити, що цільова пропускна здатність передачі залежить не лише від вікна перевантаження, але й від поточного розміру буфера. Таким чином, DRS не зможе загалом оцінити розмір буфера, який максимізує пропускну здатність передачі, оскільки він може бути обмежений поточним розміром буфера.

Деякі версії ядра Linux динамічно встановлюють розмір буфера сокета. Зокрема, навіть якщо програма вказала великий розмір буфера приймального сокета (за допомогою системного виклику setsockopt), TCP-одержувач рекламує невелике вікно отримання, яке поступово збільшується з кожним ACK-сегментом. Крім того, Linux динамічно налаштовує розмір буфера сокета відправника на основі доступної системної пам'яті та резерву буфера сокета надсилання.

3.3 Розмір буфера маршрутизатора

Більшість робіт із забезпечення буфера для маршрутизаторів і комутаторів зосереджено на проблемі великих втрат і недовикористання ємності. Зі збільшенням розміру буфера маршрутизатора рівень втрат зменшується, а використання збільшується. Такі роботи забезпечують мінімальний розмір буфера, який забезпечить прийнятний рівень втрат і використання.

Кілька статей з теорії масового обслуговування аналізують або ймовірність втрати в скінченних буферах, або ймовірність хвоста в нескінченних буферах. Зазвичай, однак, такий підхід до моделювання розглядає моделі екзогенного (або відкритого) трафіку, у яких процес надходження пакетів не залежить від стану черги.

Ранні експериментальні дослідження Villamizar і Song рекомендують, щоб розмір буфера дорівнював продукту пропускної здатності (BDP) цього каналу. «Затримка» тут відноситься до RTT одного постійного TCP-потоку, який намагається наситити цей канал, тоді як термін «пропускна здатність» стосується пропускної здатності C каналу. Однак не надано жодних рекомендацій щодо більш реалістичного випадку кількох потоків TCP з різними RTT.

Аппенцеллер [6] робить висновок, що вимога до буфера в каналі зв'язку зменшується разом із квадратним коренем із числа N «великих» TCP-потоків, які проходять через цей канал. Відповідно до їх аналізу, вимога до буфера для досягнення майже повного використання становить:

$$B=(C \cdot T_{avg})/\sqrt{N}, \tag{3.1}$$

де T_{avg} є середнім RTT N (постійних) конкуруючих з'єднань.

Ключове розуміння цієї моделі полягає в тому, що коли кількість конкуруючих потоків достатньо велика, що зазвичай буває в основних каналах зв'язку, N потоків можна вважати незалежними та несинхронізованими, і тому стандартне відхилення сукупного пропонованого навантаження (і зайнятості черги) зменшується разом із квадратним коренем із N. Важливим моментом у цій моделі є те, що вона прагне підтримувати використання смуги пропускання близько до 100%, без урахування ймовірності втрат.

Морріс був першим, хто розглянув ймовірність втрати в проблемі розміру буфера [15]. Ця робота визнає, що рівень втрат зростає з квадратом кількості конкуруючих потоків TCP, і що буферизація на основі правила BDP може спричинити часті тайм-аути TCP і неприйнятні зміни в пропускній здатності конкуруючих передач. Ця робота також пропонує механізм Flow-Proportional Queuing (FPQ) як варіант RED, який регулює обсяг буферизації пропорційно кількості потоків TCP.

Dhamdhere і Dovrolis розглядають вимогу до буфера для черги Drop-Tail, враховуючи обмеження на мінімальне використання, максимальну швидкість втрат і, коли це можливо, максимальну затримку в черзі. Вони визначають мінімальний розмір буфера, необхідний для повного використання зв'язку набором із N неоднорідних TCP-потоків, зберігаючи при цьому швидкість втрат і затримку в черзі. Однак аналіз цієї статті також обмежений припущенням про постійні зв'язки.

Нещодавно журнал ACM Computer Communication Review організував дискусію щодо визначення розміру буфера за допомогою послідовності літер. Dhamdhere і Dovrolis стверджують, що нещодавні пропозиції щодо значно менших розмірів буферів можуть призвести до значних втрат і зниження продуктивності на прикладному рівні.

3.4 Реагування на перевантаження та процес надходження потоку

Поняття здатності реагувати на перевантаження пов'язане з питанням зручності TCP. Флойд і Фол запропонували контролювати потоки з високою пропускною здатністю, щоб запобігти постійному перевантаженню.

Аналогічним чином, Network Border Patrol [16] є механізмом обмеження передач, які мають більш високу пропускну здатність, ніж швидкість, сумісна з TCP. Останнє – це середня пропускна здатність, яку матиме постійне з'єднання TCP на тому самому шляху. Було запропоновано метод оцінки частки перевантаженого трафіку, який не реагує на посилання.

У деяких попередніх роботах використовуються моделі непостійних потоків, але часто без обговорення того, яка модель OL чи CL є більш прийнятною. Бен Фредж [17] розглянув модель OL. Він зазначив, що єдине зниження запропонованого навантаження при перевантаженні відбувається через переривання передачі. Однак такі передачі призводять до втрати пропускної здатності та незадоволення користувачів. Автор запропонував контроль допуску як єдиний ефективний спосіб запобігти перевантаженню.

За останні кілька років, особливо після основоположної роботи Келлі [18], кілька дослідників застосували теорію управління для вивчення стабільності контролю перевантаження TCP. Ключовим моментом цього напрямку роботи є те, що він передбачає постійні потоки TCP і зосереджується на асимптотичній стабільності розміру черги у вузькому місці мережі. Припущення про постійні потоки знімає з проблеми важливість процесу надходження потоку.

Хейман [19] використовували модель CL для аналізу продуктивності веб-подібного трафіку через TCP. Він показав, що ефективна пропускна здатність сеансу та частка часу, протягом якого система має дану кількість активних сеансів, нечутливі до розподілу розмірів сеансу та «часу обдумування», і вони залежать лише від середнього значення цих розподілів..

Більшість попередніх робіт з моделями OL або CL припускають, що контроль перевантажень TCP може спільно використовувати пропускну здатність каналу як плавного сервера PS. Показано, що модель PS не завжди точна, здебільшого через те, що передача TCP не підтримує повне використання каналу за певних умов. У кваліфікаційній роботі використовуєтьсямо модель PS, щоб отримати деяке уявлення про реагування на перевантаження моделей OL і CL. З іншого боку, моделювання використовує фактичні передачі TCP.

Також досліджувались відмінності між моделями OL і CL в контексті мереж черг, зосереджуючись на зв’язку між середнім розміром черги в черзі вузького місця та мінливістю розподілу часу обслуговування роботи.

У статті, яка тісно пов’язана з кваліфікаційною роботою, Schroeder et al. [20] порівнюють моделі OL і CL у загальному контексті надходження завдань на сервер.

Вони підкреслюють відмінності між двома моделями з точки зору середнього часу виконання роботи та зосереджуються на ефективності різних політик планування роботи для кожної моделі. Моделі відкритого та замкнутого циклу є простими моделями поведінки користувача та програми.

Розуміння такої поведінки, а також її впливу на мережевий трафік було предметом деяких попередніх робіт [18]. Ці документи зосереджені на HTTP, здебільшого тому, що прикладні характеристики веб-перегляду добре зрозумілі. Представлені моделі для різних рівнів аналізу HTTP, а саме на рівнях пакетів, з'єднань, сторінок або сеансів. Була спроба реконструювати HTTP-сесії з односпрямованих трас заголовків TCP. Також використовувалася інформація про корисне навантаження для ідентифікації пов'язаних HTTP-запитів/відповідей і для вивчення впливу перевантаження на поведінку користувача з точки зору ймовірності перерваних і повторних сеансів.

4 ПОПЕРЕДЖЕННЯ ПЕРЕВАНТАЖЕНЬ НА ОСНОВІ ВТРАТ І ЗАТРИМКИ

Одна з фундаментальних вимог до стабільної мережі полягає в тому, щоб її трафік реагував на перевантаження. Однак виявлення перевантаження в Інтернеті є складним через те, що мережа не надає жодних сповіщень про перевантаження джерелам трафіку. Було кілька пропозицій щодо додавання явного сповіщення про перевантаження (Explicit congestion notification – ECN) в Інтернет. Однак ECN отримав мінімальне розгортання і ще не став основним механізмом. За відсутності ECN Інтернет-потоки повинні виявити перевантаження, перш ніж вони зможуть на нього відповісти.

Кінцеві системи можуть виявляти перевантаження мережі, вимірюючи збільшення затримок або втрати пакетів на шляху. Уникнення перевантаження на основі затримки (Delay-based Congestion Avoidance – DCA) сприймає збільшення затримок як ознаку перевантаження, тоді як уникнення перевантаження на основі втрат (Loss-based Congestion Avoidance – LCA) реагує на перевантаження лише після втрати пакета. Перші алгоритми, засновані на LCA або DCA, були запропоновані приблизно в той же час, і дебати щодо того, яка схема уникнення перевантажень краща, все ще тривають. Схеми LCA розглядають лише втрати пакетів як ознаку перевантаження мережі. Тому TCP-передача на основі LCA зменшує своє вікно надсилання, щоб зменшити навантаження на мережу, лише після втрати пакета. Канонічним прикладом LCA є оригінальний контроль перевантаження TCP Джейкобсона. LCA, як запропоновано в [22], є найпоширенішою схемою контролю перевантаження в поточному Інтернеті. Однак на шляхах зі значною доступною пропускною здатністю LCA може бути менш ефективним. Це пояснюється тим, що відправник LCA продовжує збільшувати своє вікно надсилання, навіть після того, як він зайняв доступну пропускну здатність, що зрештою спричиняє переповнення буфера. Мультиплікативне зменшення вікна надсилання після втрат пакетів і

подальше адитивне збільшення призводять до недостатнього використання каналу зв’язку протягом значних інтервалів часу. Ці «самостійні» скидання пакетів призводять не лише до збільшення рівня втрат, але також спричиняють зниження пропускної здатності та, принаймні з чергами Drop-Tail, значні варіації затримки.

Схеми DCA, з іншого боку, намагаються контролювати вікно надсилання передачі TCP на основі вимірювань часу проходження в обидві сторони (RTT). Основна інтуїція полягає в тому, що якщо вікно надсилання достатньо велике, щоб наситити доступну пропускну здатність, тоді будь-яке подальше збільшення вікна надсилання спричинить накопичення черги на вузькому з’єднанні та, отже, збільшення RTT.

Отже, TCP-відправник на основі DCA зменшує вікно надсилання передачі після виявлення збільшення RTT. Існує кілька варіантів схем DCA. Інтерес до алгоритмів DCA знову з’явився протягом останніх кількох років із появою високошвидкісних мереж, частково через те, що протокол TCP на основі LCA не може ефективно використовувати таку високу пропускну здатність і міжміські мережеві шляхи.

Однак останнім часом дослідження вимірювань показали, що існує незначна кореляція між збільшенням затримок (або RTT) і застійними втратами. Це експериментальне спостереження викликає деякі серйозні сумніви щодо того, чи алгоритми DCA будуть ефективними на практиці, оскільки їхнє основне припущення полягає в тому, що вимірювання RTT можна використовувати для прогнозування, виявлення та уникнення перевантажень мережі.

4.1 LCA у трактах з високою пропускною здатністю та затримкою

Залежність протоколу TCP на базі LCA від скидання пакетів для виявлення перевантажень може фактично спричинити такі скидання в інакше не перевантаженому каналі. У шляхах з високою доступною пропускною

здатністю та достатнім розміром буфера маршрутизатора DCA може виявитися привабливою альтернативою для уникнення втрат, викликаних власними силами через LCA.

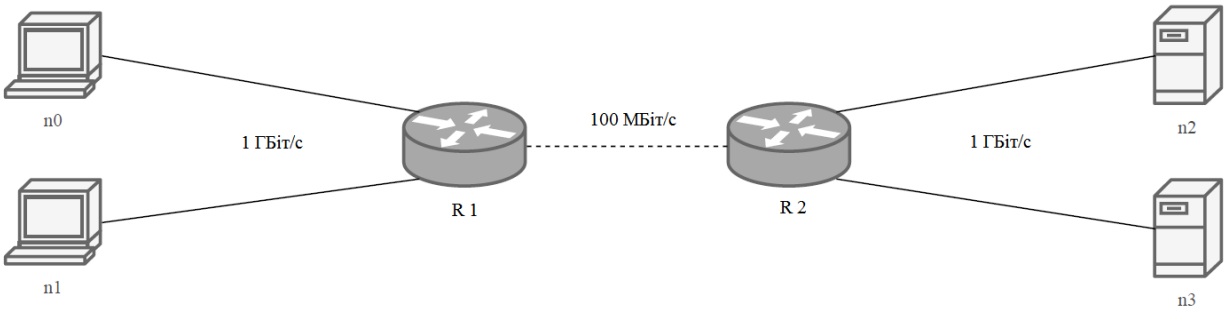

Рисунок 4.1 – Топологія мережі для моделювання

Network Simulator 3 – це симулятор повединки мережі з дискретними різноманітними подіями для Інтернет-мереж, призначений для дослідницьких або освітніх цілей.

Ns-3 – це безкоштовне програмне забезпечення, яке може підлягати ліцензуванню за GNU GPLv2 та загальнодоступне для наукових досліджень, розробок і використання.

Топологія, що обрана для проведення моделювання, представляє собою просту топологію гантелі, через яку проходить трафік від лівих вузлів до правих.

Ця топологія включає до себе 5 мереж, які розділені двома маршрутизаторами. При цьому, граничні канали мають пропускну здатність 1 ГБіт/с, а з'єднання між маршрутизаторами – 100 Мбіт/с. Роутер n_1 виконує роль вузького місця мережі.

Проведено аналіз деяких TCP-передач між хостами в представленій мережевій моделі. Ця модель має зв’язок FastEthernet як вузьке місце, і має відсутність перехресного трафіку під час експериментів. Під час цих експериментів доступна пропускна здатність шляху, стабільно становила близько 94 Мбіт/с, пропускна здатність IP-рівня з’єднання FastEthernet.

На рисунку 4.2 показано часові ряди пропускної здатності TCP і RTT. Пропускна здатність TCP вимірюється як хороша пропускна здатність. Можна спостерігати, що щоразу, коли пропускна здатність TCP наближається до доступної пропускної здатності шляху в 94 Мбіт/с, спостерігається тенденція до збільшення RTT. Оскільки TCP на основі LCA ігнорує ці сигнали RTT, він відчуває відкидання пакетів після збільшення RTT. Середня пропускна здатність для цієї передачі залишається лише близько 64 Мбіт/с, що на 30% менше доступної пропускної здатності каналу в вузькому місці.

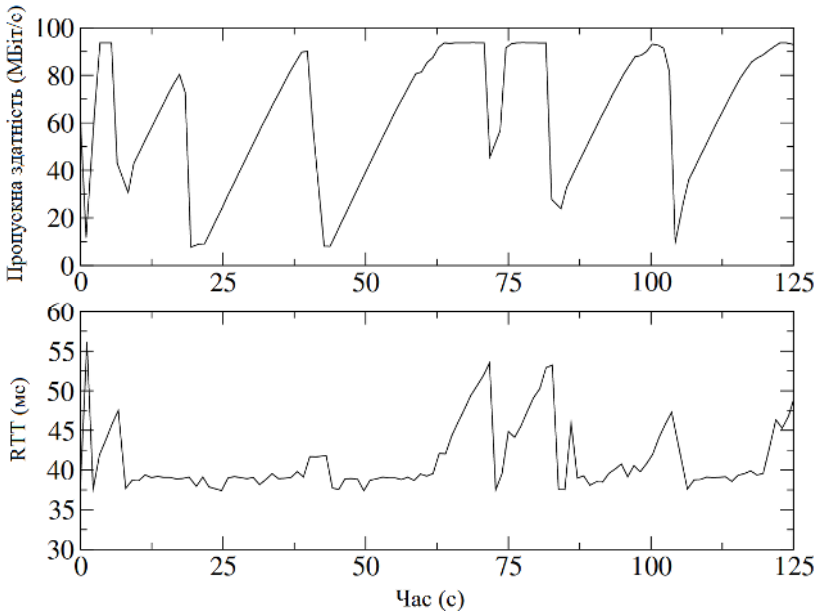

Рисунок 4.2 – Динаміка зміни пропускної здатності та RTT без обмеження розміру буфера сокета

На рисунку 4.3 показано часові ряди пропускної здатності TCP і RTT для того самого шляху. Однак у цьому експерименті було обмежено розмір буфера сокета для передачі добутком затримки та пропускної здатності вузького місця, таким чином змусивши його поводитися як TCP на основі DCA. З доступною смугою пропускання 94 Мбіт/с і RTT 40 мс передача з розміром буфера сокета 470 КБ не намагається збільшити швидкість

надсилання пакетів за межі доступної смуги пропускання, і таким чином утримує RTT близько до затримки поширення шляху близько 40 мс. У цьому випадку TCP не відчуває збільшення RTT, втрати пакетів, а середня пропускна здатність передачі залишається близькою до доступної пропускної здатності шляху.

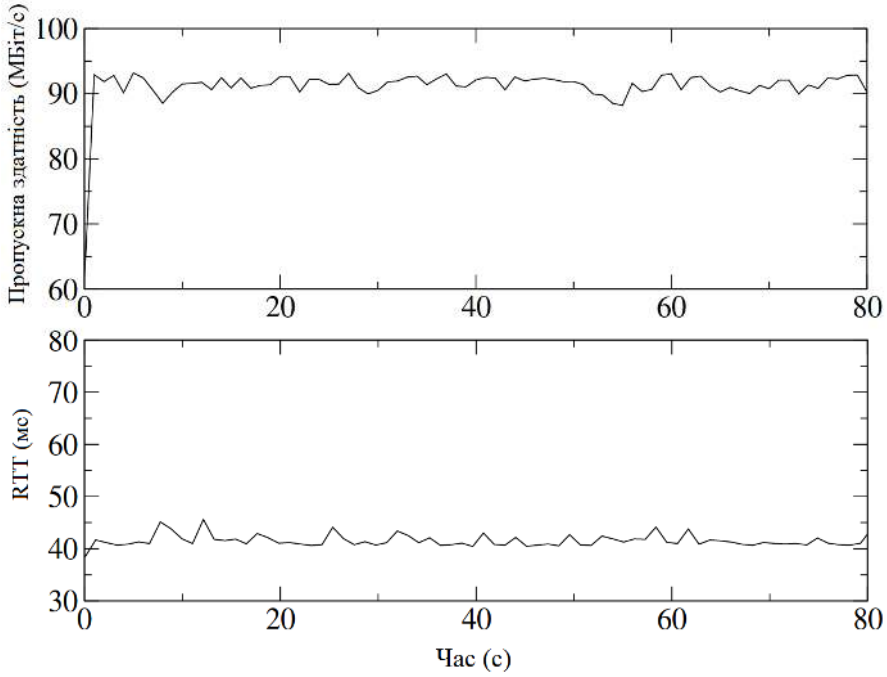

Рисунок 4.3 – Часові ряди пропускної здатності та RTT передачі TCP з обмеженим розміром буфера сокета 470 КБ

Підводячи підсумок, протокол TCP на основі LCA може спричиняти втрати, викликані власними силами. Такі втрати призводять до низької продуктивності TCP. З іншого боку, DCA дозволяє уникнути власних втрат.

4.2 Проблеми для DCA

Слід розглянути різні фактори, які впливають на продуктивність TCP на основі DCA. Ці фактори сприяють труднощам точного та своєчасного вимірювання зростаючих затримок, слабкої кореляції між збільшенням RTT і

подіями перевантаження, а також вимагають потреби в LCA, як резервного механізму для роботи з відкиданням пакетів, які не корельовані зі збільшенням RTT.

4.2.1 Максимальна варіація затримки та розмір мережевого буфера

Слід розглянути, чи є збільшення затримки в черзі більшим по відношенню до шуму в вимірюванні RTT шляху. Це також підвищує важливість розміру мережевого буфера, оскільки він визначає максимальне збільшення затримки в черзі.

Припустимо, що мережевий шлях передачі TCP має мінімальний RTT T_{min}. T_{min} визначається затримками розповсюдження та передачі вздовж шляху та не включає затримки в черзі. Крім того, припустимо, що щільна ланка на шляху має пропускну здатність C_t і розмір буфера B_t. Максимальна затримка в черзі в цьому каналі становить B_t/C_t. Якщо черга виникає лише на вузькому з'єднанні, тоді максимальне RTT буде задано:

$$T_{max} = T_{min} + \frac{B_t}{C_t}. \tag{4.1}$$

Алгоритми DCA керують вікном відправки на основі виміряної зміни RTT ΔT, яка буде меншою за максимальну затримку в черзі B_t/C_t. Перша проблема тут полягає в тому, що якщо $B_t/C_t << T_{min}$, то може бути складно надійно виміряти збільшення RTT через черги на вузькому каналі.

По-друге, випадковий шум у вимірах RTT, який неминучий на практиці, або помилки через обмежене розмежування тимчасових міток RTT можуть бути зіставні або навіть більше, ніж затримки у чергах у вузькому каналі. У цьому випадку відправник на основі DCA також може виявитися не в змозі виявити збільшення затримок у черзі на вузькому каналі та уникнути перевантажувальних втрат.

4.2.2 Частота дискретизації RTT

В алгоритмах DCA відправник вимірює RTT, використовуючи пакети передачі даних. Навіть якщо кожен пакет даних генерує вимірювання RTT, відправник буде ефективно вимірювати RTT кожні $T_s = L/R$ секунд, де L – розмір пакета даних, а R – швидкість відправки за цей період часу. Тоді частота дискретизації RTT дорівнюватиме R/L.

Проте з теореми Найквіста відомо, що якщо частота дискретизації менш ніж удвічі перевищує максимальну частоту в спектрі дискретизированного сигналу, ми не можемо відновити цей сигнал. Навіть якщо ми не прагнемо ідеального відновлення сигналу, частота дискретизації повинна бути принаймні порівнянна з максимальною частотою сигналу; в іншому випадку ми ризикуємо просто отримати шум вибірки.

RTT на мережному шляху варіюється в основному через затримки у чергах, і ці затримки викликані змінами вхідної швидкості відповідних черг. Щоб проілюструвати цей момент, розгляньмо чергу навантаженого каналу в момент часу t_0. Припустимо, що перед моментом t_0 черга мала деяку доступну смугу пропускання A і швидкість передачі була R.

У момент t_0 новий потік перехресного трафіку надходить на щільний канал зі швидкістю $R_c > A$. Це призведе до створення черги на навантаженому каналі зі швидкістю $R_c - A$. Якщо буфер навантаженого каналу був порожній у момент t_0 і має максимум розміру B_t байт, для переповнення цього буфера знадобиться $B_t/(R_c - A)$ секунд. Щоб передача DCA вчасно виявила збільшення RTT, вона повинна мати набагато менший період вибірки, тобто:

$$\frac{L}{R} << \frac{B_t}{R_c - A}. \tag{4.2}$$

Або, швидкість має бути значно вищою, ніж швидкість, з якою відставання збільшується на цьому зв’язку, нормалізована розміром буфера.

$$R >> \frac{R_c - A}{B_t / L}. \quad (4.3)$$

Рівняння (4.3) показує, що якщо передача є потоком з низькою пропускною здатністю в тракті з високою пропускною здатністю (тобто, R<<A), вона може бути не в змозі точно відібрати варіації RTT на шляху. Потоки перехресного трафіку з високою пропускною здатністю спричинять значні коливання RTT, які передача не зможе вчасно виявити та, отже, на них відповісти. Ця гіпотеза узгоджується з експериментальними результатами та результатами моделювання різноманітних дослідників, згідно з якими алгоритми DCA часто неправильно реагують на варіації RTT, особливо в трактах з високою пропускною здатністю.

4.2.3 Ступінь мультиплексування

DCA часто описується та моделюється з урахуванням однієї передачі з використанням тісного зв'язку на шляху. Навіть якщо перехресний трафік є частиною моделі, часто припускають, що він не викликає будь-яких варіацій затримки в черзі.

У цьому контексті будь-які зміни у RTT викликані відповідними варіаціями у вікні надсилання цього з'єднання. Тоді відносно просто розробити алгоритми DCA, які регулюють вікно надсилання на основі вимірювань RTT, щоб передача TCP повністю використовувала канал, не викликаючи перевантаження.

Однак ситуація стає зовсім іншою, якщо враховувати, що канал має кілька TCP-потоків, і що всі вони динамічно налаштовують свої вікна на основі свого власного сприйняття стану мережі. Зокрема, в якийсь момент часу швидкість надсилання пакетів дорівнює R, тоді як пропускна здатність вузького з'єднання дорівнює C_t, що повністю використовується декількома TCP-з'єднаннями перехресного трафіку із загальною швидкістю $R_c = C_t - R$.

Якщо $R>>R_c$, потік буде мати великий вплив на чергу вузького каналу, тому буде сильна позитивна кореляція між вікном передачі та виміряними RTT. У такому випадку можна очікувати, що алгоритм DCA працюватиме добре.

Однак якщо $R<<R_c$, потік буде незначним внеском у затримки в черзі на тісному каналі зв’язку, і тому виміряні RTT будуть слабо корельованими або навіть не корельованими з цим вікном передачі. Таким чином, цілком імовірно, що тотік буде реагувати на варіації RTT помилково, здебільшого через вплив перехресного трафіку на вузьке з’єднання, а не через вплив власного навантаження. Було помічено, що алгоритми DCA добре працюють у вузьких каналах з низькою пропускною здатністю, оскільки зазвичай існує лише одне з'єднання, яке використовує ці канали одночасно. З іншого боку, у каналах з високою пропускною здатністю, де існує велика кількість одночасних з’єднань від різних користувачів, схеми DCA працюють набагато гірше. У такому випадку схеми DCA не можуть точно передбачити перевантаження і часто помилково змінюють свої вікна відправки.

4.2.4 Наявність випадкових втрат

Незважаючи на те, що транспортний протокол на основі DCA в основному реагує на варіації RTT, він також повинен реагувати на втрату пакетів, зменшуючи вікно надсилання. Втрат пакетів в мережевих каналах не уникнути з двох причин. По-перше, випадкове відкидання пакетів відбувається незалежно від перевантаження через бітові помилки, апаратні/програмні помилки тощо. Без глобального розгортання біта явного сповіщення про перевантаження (ECN) кінцеві хости не можуть розрізнити застійні та випадкові втрати, а тому їм потрібно реагувати на обидва випадки.

По-друге, великі втрати пакетів можуть бути спричинені перехресним трафіком, незалежно від вікна передачі. Раніше на рисунку 4.2 спостерігалися такі відкидання пакетів через 17,5 і 40,0 секунд, яким не

передувало насичення шляху та збільшення затримки в черзі, і це також могло вплинути на TCP на основі DCA. Однак TCP-з'єднання не має засобів для висновку, чи спостережувана втрата пакетів була спричинена тим, що інший потік мав зменшити своє вікно, оскільки застійні втрати можуть вплинути на будь-який з активних потоків. Підводячи підсумок, на потоки DCA можуть впливати втрати пакетів, і вони повинні реагувати на них, зменшуючи свої вікна.

Якщо схеми DCA не зменшують свої вікна на той самий коефіцієнт, що й схеми LCA, може статися значна несправедливість у розподілі смуги пропускання. Ми вважаємо, що необхідно, щоб будь-який запропонований протокол DCA детально визначав, як він реагує на відкидання пакетів і як він підтримує (або не підтримує) справедливість щодо існуючих алгоритмів контролю перевантаження TCP.

Інший спосіб поглянути на цю проблему полягає в тому, що, незважаючи на те, що алгоритми DCA здатні уникнути власних втрат пакетів і, таким чином, досягти вищої пропускної здатності у випадках, коли всі втрати є спонтанними, алгоритми DCA можуть не забезпечити жодних переваг у продуктивності, якщо вони працюють на мережевих каналах, де випадкові або застійні втрати через перехресний трафік є звичайним явищем. Отже, важливо, щоб оцінка схем DCA також проводилася в мережах із втратами, а також у мережах, де потоки DCA конкурують з потоками LCA.

5 АВТОМАТИЧНЕ ВИЗНАЧЕННЯ РОЗМІРУ БУФЕРА

Поява високошвидкісних мереж викликає новий інтерес до наскрізної продуктивності додатків із інтенсивним використанням даних. Зокрема, наукове співтовариство підвищує продуктивність мереж за допомогою таких додатків, як розподілене моделювання, віддалені спільні лабораторії та часті багатогігабайтні передачі. Як правило, такі додатки працюють у добре забезпечених мережах (Internet, ESnet, GEANT тощо), створених за допомогою каналів з високою пропускною здатністю (OC-12 або вище), які здебільшого мало завантажені. Крім того, завдяки розгортанню інтерфейсів 10-Gigabit Ethernet перевантаження також стають рідкісними на краях мережі та кінцевих хостах. З огляду на всю цю пропускну здатність, не дивно, що такі користувачі очікують чудової наскрізної продуктивності. Однак це не завжди так. Недавнє вимірювальне дослідження в Internet показало, що 90% довгих передач TCP (тобто більше 10 МБ) отримують пропускну здатність менше 10 Мбіт/с].

Поширеною є думка, що основною причиною відносно низької наскрізної пропускної здатності є TCP. Це пов’язано або з самим TCP (наприклад, алгоритмами та параметрами контролю перевантаження), або через конфігурацію локальної системи (наприклад, стандартний або максимальний розмір буфера сокета).

TCP звинувачують у тому, що він повільно захоплює доступну пропускну здатність високопродуктивних мереж, головним чином через дві причини:

- невеликі буфери сокетів на кінцевих хостах обмежують ефективне вікно передачі, а отже, максимальну пропускну здатність;

- втрати пакетів викликають велике скорочення вікна з наступним повільним (лінійним) збільшенням швидкості вікна, що зменшує середню пропускну здатність передачі.

Іншими проблемами, пов’язаними з TCP, які перешкоджають продуктивності, є численні втрати пакетів наприкінці повільного запуску (зазвичай призводять до тайм-аутів), неможливість розрізнити застійні та випадкові втрати пакетів, використання малих сегментів або початкове значення ssthresh.

Паралельні з’єднання TCP можуть збільшити загальну пропускну здатність, яку отримує програма. Однак ця методика викликає питання справедливості, оскільки сукупність N з’єднань зменшує своє сукупне вікно на коефіцієнт 1/2N, а не на 1/2, після втрати пакета. Крім того, загальна швидкість збільшення вікна в N разів швидша, ніж у одного підключення.

Нарешті, методи, які автоматично регулюють розмір буфера сокета, можуть бути виконані на прикладному рівні, тому вони не вимагають змін у реалізації TCP або протоколі.

В кваліфікаційній роботі використовується підхід автоматичного визначення розміру буфера сокета.

Буфери сокетів надсилання та отримання мають бути достатньо великими, щоб передача могла наситити основний мережевий шлях. Зокрема, припустимо, що вузьке місце шляху має пропускну здатність C біт/с, а шлях між відправником і одержувачем має час проходження (RTT) T секунд. Коли немає конкуруючого трафіку, з’єднання зможе наситити шлях, якщо його вікно надсилання дорівнює C×T, тобто добре відомий продукт затримки пропускної здатності (BDP) шляху. Однак для того, щоб вікно було таким великим, керування потоком TCP вимагає, щоб менший із двох буферів сокетів (надсилання та отримання) був однаково великим. Якщо розмір S меншого буфера сокета менший, ніж C×T, підключення буде використовувати шлях недостатньо. Якщо S більше, ніж C×T, з’єднання перевантажить шлях.

У цьому випадку, залежно від обсягу буферизації у зв’язку з вузьким місцем, передача може спричинити переповнення буфера, зменшення вікна та падіння пропускної здатності.

Розмір буфера сокета має дорівнювати BDP лише тоді, коли мережевий шлях не несе перехресного трафіку. Наявність перехресного трафіку означає, що «пропускна здатність» шляху буде не C, а трохи менше. Наступний пункт представляє модель мережевого шляху, яка допомагає зрозуміти ці проблеми, і вводить важливий показник, який називається максимально можливою пропускною здатністю (Maximum Feasible Throughput – MFT).

В роботі розрізняються перевантажені та не перевантажені мережеві шляхи. В останньому ймовірність застійної втрати (переповнення буфера) практично дорівнює нулю. Неперевантажені шляхи сьогодні поширені, особливо у високопродуктивних добре забезпечених мережах.

В неперевантаженому шляху передача TCP може наситити доступну пропускну здатність, якщо це не спричиняє переповнення буфера. Щоб уникнути таких власних втрат, пропонується обмежити вікно надсилання за допомогою буферів сокетів відповідного розміру. З іншого боку, у перевантаженому тракті втрати виникають незалежно від вікна передачі, тому обмеження останнього може лише зменшити кінцеву пропускну здатність.

Необхідна розробка механізму прикладного рівня, який автоматично визначає розмір буфера сокета, який насичує доступну смугу пропускання в мережі, поки триває передача.

Авторозмір буфера SOcket базується на прямих вимірюваннях отриманої пропускної здатності та відповідного RTT на прикладному рівні. Ключова ідея полягає в тому, що вікно надсилання має бути обмежене після того, як передача наситить доступну смугу пропускання на шляху, щоб передача не спричиняла переповнення буфера, тобто щоб уникнути власних втрат.

З іншого боку, у перевантажених маршрутах алгоритм вимикається, щоб не обмежувати вікно передачі. Експериментальні результати в кількох сценаріях, показують, що Авторозмір буфера SOcket забезпечує стабільне значне збільшення пропускної здатності (від 20% до 80%) порівняно з

передачами TCP, які використовують максимально можливий розмір буфера сокета. Ключовим моментом Авторозміра буфера SOcket є те, що він не вимагає попереднього знання характеристик шляху, і тому його простіше використовувати, ніж схеми визначення розміру буфера сокетів, які покладаються на попередні вимірювання ємності або доступної пропускної здатності на шляху.

5.1 Розмір буфера сокета та пропускна здатність TCP

Розглянемо односпрямовану передачу TCP від відправника до одержувача. TCP використовує керування потоком на основі вікон, що означає, що відправнику дозволяється мати до певної кількості переданих, але непідтверджених байтів, які називаються вікном надсилання W_s, у будь-який час.

Вікно надсилання обмежено наступним чином:

$$W_s = \min\{W_c, W_r, B_s\}, \tag{5.1}$$

де W_c – вікно перевантаження відправника, W_r – вікно отримання, оголошене отримувачем, а B_s – розмір буфера надсилаючого сокета відправника. Вікно прийому W_r – це обсяг доступної пам’яті буфера приймального сокета обмежене розміром B_r буфера приймального сокета, тобто $W_r \leq B_r$.

Припускаємо, що $W_r = B_r$, тобто програма-одержувач достатньо швидка, щоб споживати будь-які доставлені дані, зберігаючи буфер приймального сокета завжди порожнім. Тоді вікно надсилання обмежується:

$$W_s = \min\{W_c, S\}, \tag{5.2}$$

де $S = \min\{B_s, B_r\}$ – менший із двох розмірів буфера сокета.

Якщо вікно надсилання W_s обмежене W_c, ми говоримо, що передача обмежена перевантаженням, а якщо воно обмежено S, ми говоримо, що передача обмежена буфером. Якщо $T(W_s)$ є RTT з'єднання, коли вікно надсилання W_s, пропускна здатність передачі становить:

$$R=\frac{W_s}{T(W_s)}=\frac{\min\{W_c, S\}}{T(W_s)}. \quad (5.3)$$

RTT може змінюватися залежно від W_s через затримки в черзі через саму передачу.

Розглянемо модель мережевого шляху P, через який проходить передача TCP. Передача TCP, називається цільовою передачею; решта трафіку в P називається поперечним трафіком. Передбачається, що прямий шлях від відправника до адресата і зворотний шлях від адресата до відправника є фіксованими та унікальними протягом цільової передачі.

Кожна ланка *i* шляху передає пакети з пропускною здатністю C_i біт/с. Пакети, що надійшли, відкидаються за методом Drop Tail. Нехай ρ_i − початкове середнє використання ланки *i*, тобто використання ланки *i* до цільової передачі. Доступна пропускна здатність A_i каналу *i* тоді визначається як $A_i=C_i\times(1-\rho_i)$. Таким чином, називаємо зв'язок прямого шляху P_f з мінімальною доступною пропускною здатністю $A=\min P_f\{A_i\}$ як вузький зв'язок. Розмір буфера вузького зв'язку позначається B_t.

Канал є насиченим, коли його доступна пропускна здатність дорівнює нулю. Крім того, канал є неперевантаженим, якщо коефіцієнт втрати пакетів через перевантаження практично дорівнює нулю; інакше канал перевантажений.

Шлях називається перевантаженим, якщо його вузька ланка перевантажена; інакше шлях називається неперевантажений. Екзогенний RTT T_e шляху є сумою всіх середніх затримок уздовж шляху, включаючи затримки поширення та черги, до початку цільової передачі.

Середнє значення RTT T_a, з іншого боку, є сумою всіх середніх затримок уздовж шляху, поки виконується цільова передача. Загалом $T_a \geq T_e$ через збільшення черги, викликаної цільовою передачею.

З рівняння (5.3) можна розглядати цільову пропускну здатність передачі як функцію R(S). Тоді постає важливе питання: яке значення S розміру буфера сокета, заданого мережевим шляхом P, максимізує цільову пропускну здатність передачі R(S). Максимальне значення R(S) є максимальна можлива пропускна здатність (MFT) R. Загальноприйнята думка полягає в тому, що розмір буфера сокета S повинен дорівнювати продукту затримки пропускної здатності шляху BDP, де «пропускна здатність» – це пропускна здатність шляху C, а «затримка» – це екзогенний RTT шляху T_e, тобто $S=C\times T_e$. Якщо вікно надсилання дорівнює $W_s=C\times T_e$, і припускаючи, що на шляху немає перехресного трафіку, вузьке з'єднання стає насиченим (тобто A=0), але не перевантаженим, і тому цільова передача досягає свого MFT (R=C).

На практиці мережевий шлях завжди несе деякий перехресний трафік, і, таким чином, A<C. Якщо S>A×Te, цільова передача наситить вузький канал, і, залежно від B_t, це також може спричинити втрату пакетів.

Однак втрати спричиняють мультиплікативні скидання вікна надсилання цільової передачі та, потенційно, зниження пропускної здатності. Таким чином, обсяг буферизації B_t у вузькому зв'язку є важливим фактором для визначення розміру буфера сокета, оскільки він визначає точку, в якій навантажений канал стає перевантаженим. Наявність перехресного трафіку має додатковий важливий наслідок. Якщо перехресний трафік є TCP, він реагуватиме на присутність цільової передачі, знижуючи її швидкість через втрату пакетів або через те, що цільова передача збільшила RTT на шляху ($T_a>T_e$). У цьому випадку цільова передача може досягти вищої пропускної здатності, ніж початкова доступна смуга пропускання A. MFT може бути більшою, ніж доступна смуга пропускання, залежно від реагування на перевантаження перехресного трафіку.

5.2 Максимально можлива пропускна здатність і доступна пропускна здатність

5.2.1 Шляхи без заторів

Припустимо спочатку, що мережевий шлях P не перевантажений. Далі розглянемо вплив розміру буфера сокета S на пропускну здатність R(S) на прикладі передачі TCP.

На рисунку 5.1 показано пропускну здатність і RTT TCP-з'єднання, коли розмір буферу S=128 КБайт. У цьому випадку пропускна здатність передачі залишається відносно постійною, підключення не зазнає втрат пакетів, а передача обмежена буфером.

Трафіку не вдається наситити шлях, оскільки $R(S)=S/T_e=25,5$ Мбіт/с, що набагато менше, ніж A. Очевидно, що будь-яка схема розміру буфера сокета, яка встановлює S менше, ніж A/T_e, призведе до низької продуктивності.

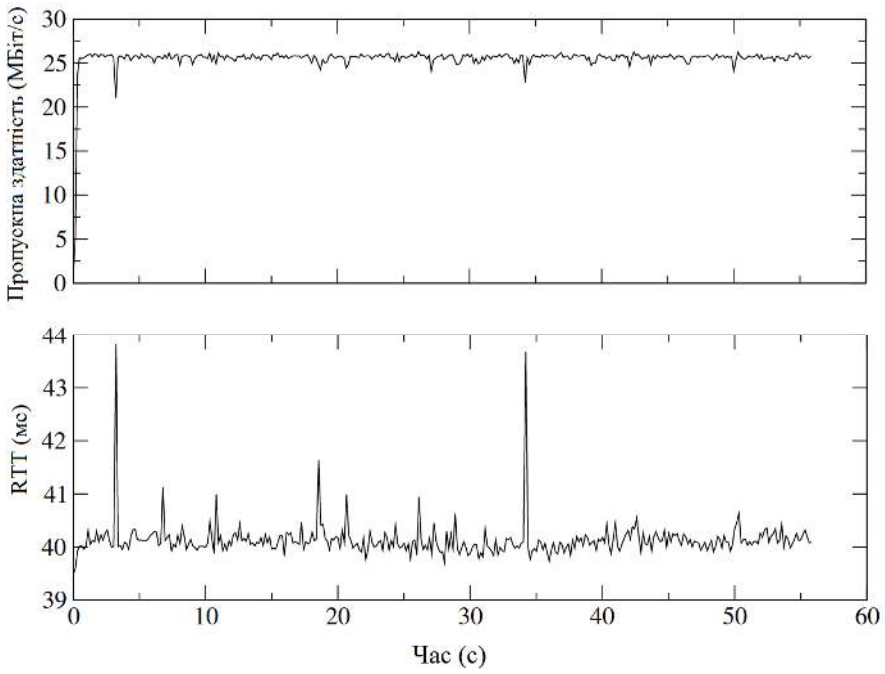

Рисунок 5.1 – Пропускна здатність і RTT передачі 200 МБ із S=128 КБайт

Далі ми збільшуємо S до значення, яке визначається доступною пропускною здатністю, тобто $S=A\times T_e=400$ КБайт (рисунок 5.2). Очікується, що в цьому випадку передача наситить шлях, не спричиняючи постійної черги та втрат пакетів. Дійсно, з'єднання все ще обмежене буфером, отримуючи приблизно доступну пропускну здатність на шляху (R(S)=79,6 Мбіт/с). Оскільки S визначався доступною пропускною здатністю, передача не призвела до постійного відставання в черзі вузького каналу, отже, $T_a=T_e=40$ мс.

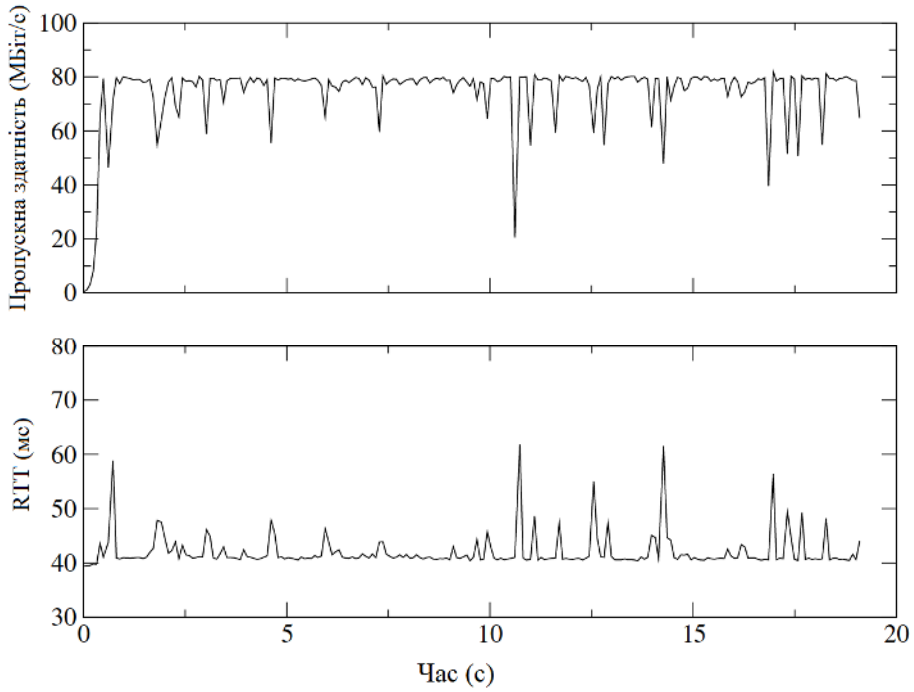

Рисунок 5.2 – Пропускна здатність і RTT передачі 200 МБ із S=400 КБайт

Може бути оманою думати, що попередній випадок відповідає оптимальному розміру буфера сокета, тобто, що MFT дорівнює R=A. Однак MFT шляху залежить від реагування на перевантаження перехресного трафіку. Якщо перехресний трафік не реагує на перевантаження, наприклад трафік UDP, або є сукупністю коротких потоків TCP, він підтримуватиме майже постійну пропускну здатність, доки цільова передача не спричиняє переповнення буфера та втрати пакетів.

У цьому випадку MFT буде дорівнювати доступній смузі пропускання. Однак якщо перехресний трафік складається з постійних TCP-передач з обмеженим буфером, будь-яке збільшення RTT призведе до зменшення їх пропускної здатності. У цьому випадку цільова передача може «вкрасти» частину пропускної здатності передачі перехресного трафіку, спричиняючи постійне відставання в навантаженому каналі та збільшуючи MFT, ніж доступна пропускна здатність.

Для визначення ефекту реагування на перевантаження перехресного трафіку, можна збільшити S до 550 КБайт (рисунок 5.3). По-перше, передача все ще обмежена буфером, оскільки вона не зазнає втрат пакетів. По-друге, RTT збільшується на 9 мс від T_e=40 мс до T_a=49 мс. Отже, пропускна здатність цільової передачі досягає R(S)=S/Ta=89,6 Мбіт/с, що більше, ніж доступна пропускна здатність для цільові передачі.

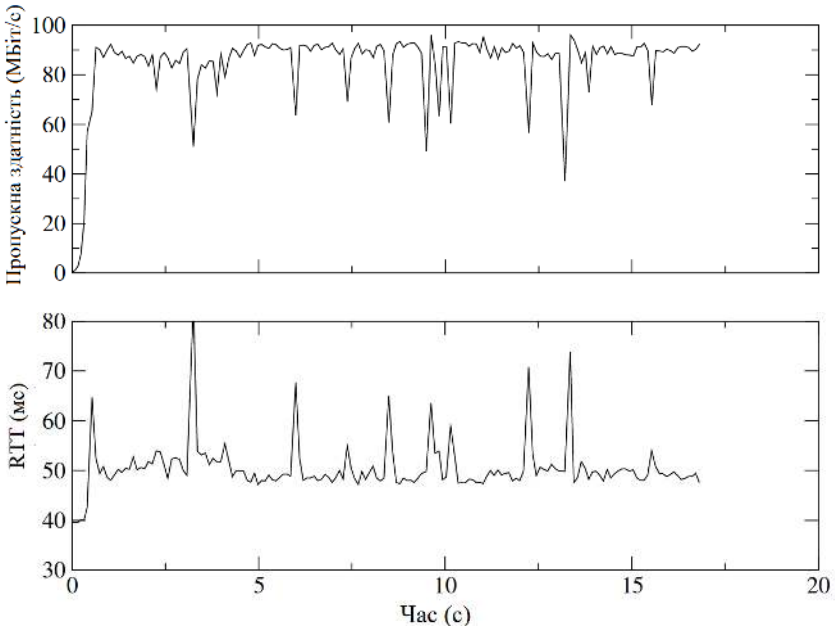

Рисунок 5.3 – Пропускна здатність і RTT передачі 200 МБ із S=550 КБайт

Можна подумати, що збільшення S ще більше призведе до більш високої пропускної здатності. При збільшені S понад певну точку, цільова передача спричиняє переповнення буфера в навантаженому каналі.

Тоді передача буде обмежена перевантаженням, реагуючи на відкидання пакетів великими скороченнями вікна та повільним збільшенням вікна.

Щоб проілюструвати цей випадок, на рисунку 5.4 показано, що відбувається з цільовою передачею, коли S встановлено на 900 КБайт. З'єднання зазнає кількох втрат під час початкового повільного запуску (приблизно через одну секунду після його початку), після чого наступає тайм-аут. Додаткові втрати виникають приблизно через 12 секунд, що також спричиняє значне зниження пропускної здатності.

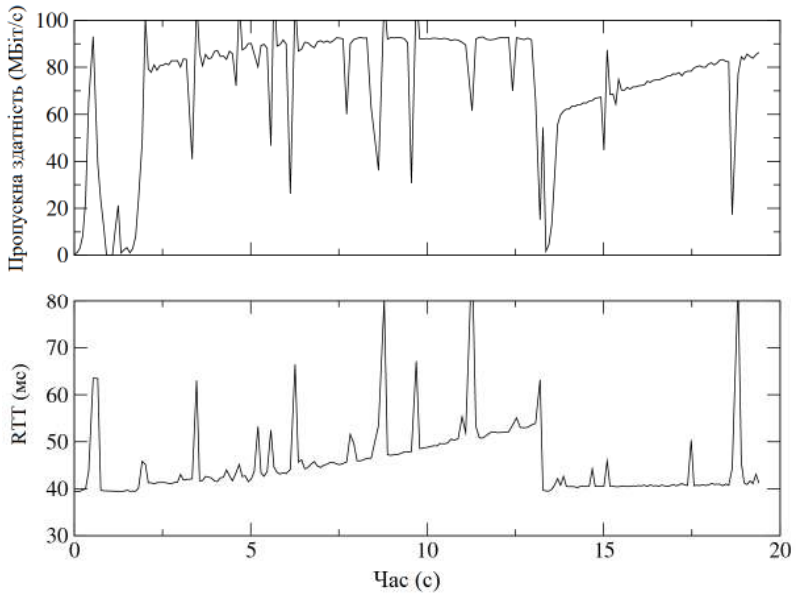

Рисунок 5.4 – Пропускна здатність і RTT передачі 200 МБ із S=900 КБайт

Попередні чотири випадки ілюструють, що розмір буфера сокета має великий вплив на пропускну здатність TCP у неперевантажених шляхах. Цільова передача може досягти свого MFT з максимально можливим розміром буфера сокета, який не спричиняє самостійних втрат пакетів. Залежно від реагування на перевантаження перехресного трафіку, MFT може бути досяжним, лише якщо цільова передача створює постійне відставання у вузькому каналі та значне збільшення RTT.

З іншого боку, обмеження розміру буфера сокета на основі доступної пропускної здатності не збільшує RTT шляху, але може призвести до неоптимальної пропускної здатності.

Важливим моментом є те, що кінцеві хости не знають обсягу буферизації на навантаженому каналі B_t або природи перехресного трафіку. Отже, може бути неможливо передбачити значення S, яке призведе до самоіндукованих втрат, і, отже, отримати MFT.

Замість цього можна визначити S на основі доступної смуги пропускання. Це просто точка, в якій отримана пропускна здатність стає практично постійною, а RTT починає збільшуватися. Незважаючи на те, що налаштування S на основі доступної пропускної здатності може бути неоптимальним порівняно з MFT, це краща ціль з таких причин:

- оскільки обсяг буферизації в вузькому каналі зв’язку невідомий, накопичення постійного відставання може призвести до ранніх застійних втрат, значно зменшуючи пропускну здатність цільової передачі;

- значне збільшення RTT може бути шкідливим для продуктивності реального часу та інтерактивного трафіку на тому самому шляху;

- збільшення пропускної здатності цільової передачі шляхом навмисного збільшення RTT інших TCP-з’єднань є нечесною поведінкою перевантаження.

5.2.2 Перевантажені шляхи

Шлях може бути перевантаженим, наприклад, якщо він несе одну або кілька постійних передач TCP, обмежених перевантаженням, або якщо є втрати пакетів у вузькому каналі зв’язку через різкий перехресний трафік.

Ключовим моментом, який відрізняє перевантажені шляхи від неперевантажених, є те, що цільова передача може мати втрати пакетів незалежно від розміру буфера сокета. Це наслідок постановки в чергу Drop Tail: відкинуті пакети можуть належати до будь-якого потоку. У цьому

випадку обмежений буфер сокета може лише зменшити пропускну здатність цільової передачі. Таким чином, щоб максимізувати пропускну здатність цільової передачі, розмір буфера сокета S повинен бути достатньо великим, щоб передача завжди була обмежена перевантаженням.

Середня пропускна здатність передачі TCP у перевантаженому тракті з рівнем втрат p і середнім RTT T становить:

$$R(S) \approx \min\left\{\frac{S}{T}, f(T,p)\right\}, \tag{5.4}$$

де S – максимально можливе вікно передачі (еквівалент розміру буфера сокета), а f(T,p) – функція, яка залежить від алгоритму уникнення перевантаження TCP. Рівняння (5.4) показує, що в перевантаженому шляху (p>0) обмежений розмір буфера сокета S може лише зменшити пропускну здатність цільової передачі, але ніколи її не збільшити.

Таким чином, оптимальний розмір буфера сокета в перевантаженому шляху $S=S_\infty$, де S_∞ є достатньо великим значенням, щоб зробити перевантаження обмеженим протягом усього терміну існування з'єднання, тобто $S_\infty > \max W_c$.

В наступному сценарії пропускна здатність шляху становить C=10 Мбіт/с, RTT 78 мс, а доступна пропускна здатність A – близько 3 Мбіт/с.

На малюнку 5.5 S обмежено 30 КБ, що є значенням, яке визначається доступною пропускною здатністю ($S=AT_c$). Незважаючи на те, що передача не перевантажує шлях, під час з'єднання втрачається кілька пакетів. Середня пропускна здатність передачі в цьому випадку становить 2,4 Мбіт/с.

З іншого боку, на рисунку 5.6 S збільшується до максимально можливого значення, тому передача завжди обмежена перевантаженням.

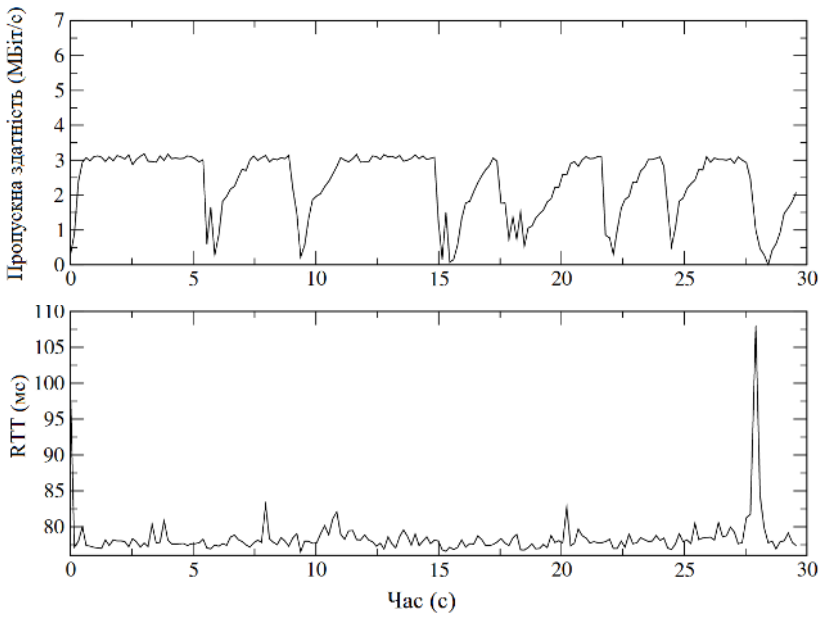

Рисунок 5.5 – Пропускна здатність і RTT передачі 30 МБ у перевантаженому шляху з S=30 КБайт

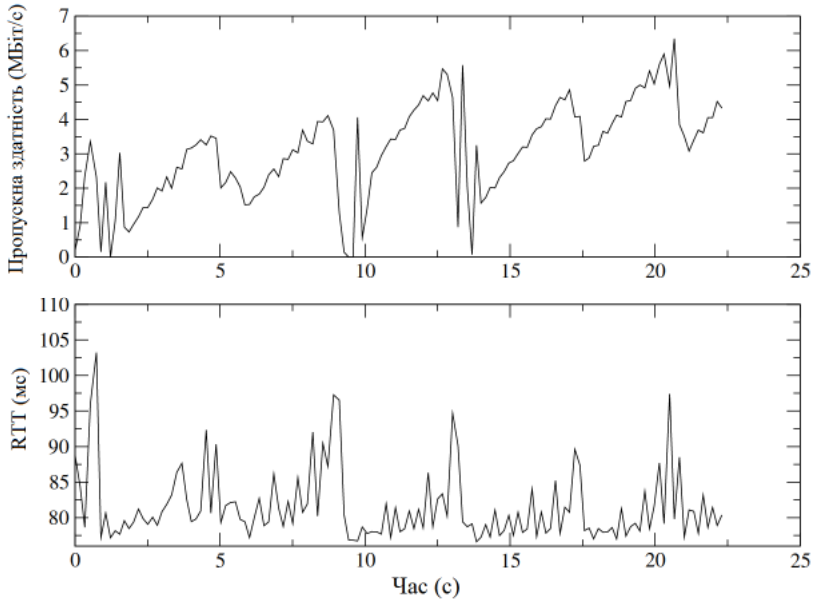

Рисунок 5.6 – Пропускна здатність і RTT передачі 30 МБ у перевантаженому шляху з $S=S_{\infty}$

Передача знову зазнає кількох втрат, але оскільки цього разу вона не обмежена S, вона досягає більшої середньої пропускної здатності, близької до 3,1 Мбіт/с.

6 АВТОМАТИЧНЕ ВИЗНАЧЕННЯ РОЗМІРУ БУФЕРА СОКЕТУ

Мета визначення розміру буферу полягає полягає в тому, щоб наситити доступну пропускну здатність неперевантаженого мережевого шляху, не викликаючи значного збільшення RTT. Такий метод не повинен вимагати змін у протоколі TCP або реалізації, тому його можна інтегрувати з будь-якою програмою передачі даних.

Для цього не потрібні попередні знання про пропускну здатність або доступну пропускну здатність, тоді як RTT і наявність втрат пакетів визначаються безпосередньо програмою за допомогою пакетів позасмугового зондування UDP.

Пропускна здатність передачі також вимірюється додатком на основі періодичних вимірювань отриманої пропускної здатності.

Ймовірно, це не покращить пропускну здатність коротких передач, особливо якщо вони закінчуються до кінця повільного старту. По-друге, метод діє лише на незавантажених шляхах. У шляхах із постійним перевантаженням або обмеженою доступною пропускною здатністю автоматичне визначення розміру буфера вимкнеться, встановивши для розміру буфера сокета максимально можливе значення. Таким чином, його пропускна здатність обмежена вікном перевантаження та отримує таку ж пропускну здатність, як і звичайна передача TCP.

Маються певні вимоги до хоста та маршрутизатора для ефективної роботи методу. Реалізація TCP на обох кінцевих хостах повинна підтримувати масштабування вікна.

Максимально допустимий розмір буфера сокета як у відправника, так і в отримувача має бути достатньо великим, щоб не обмежувати пропускну здатність з’єднання. Елементи мережі вздовж шляху використовують буфери Drop Tail, а не активні черги.

6.1 Основна ідея

Основна ідея методу полягає в наступному. У неперевантажених шляхах метод має обмежити розмір буфера сокета приймача, а отже, максимально можливе вікно надсилання, щоб передача насичувала шлях, але не спричиняла переповнення буфера. З іншого боку, у перевантажених шляхах має встановити максимально можливе значення розміру буфера сокета, щоб передача була обмежена перевантаженням.

Метод визначає точку, в якій трафік насичує доступну смугу пропускання, використовуючи дві «сигнатури» у вимірюваннях пропускної здатності: фіксовану швидкість і спад швидкості. Умова фіксованої швидкості виявляється, коли пропускна здатність виглядає майже постійною протягом деякого періоду часу. Умова постійного падіння швидкості виникає, коли метод не може уникнути власних втрат, і це виявляється як падіння швидкості після короткого періоду часу, протягом якого пропускна здатність була постійною.

```
if (фіксована швидкість або постійний спад швидкості)
{
  if (не перевантажений шлях)
    S=R×T; (встановити розмір буфера сокету з оцінкою часу RTT)
  else S=SMAX; (встановити максимальне можливе значення розміру
буфера сокету)
}
```

Рисунок 6.1 – Базовий алгоритм методу визначення розміру буфера

6.2 Деталі реалізації та діаграма стану

Одержувач надсилає відправнику позасмуговий періодичний потік пакетів-зондів UDP. Відправник повертає пакети одержувачу з порядковим номером зонду. Одержувач використовує ці порядкові номери для оцінки рівня втрат на прямому шляху, роблячи висновок про те, перевантажений

шлях чи ні. Незважаючи на те, що добре відомо, що періодичне тестування може не призвести до точної оцінки рівня втрат, потрібно лише знати, чи перевантажений шлях, тобто чи коефіцієнт втрат відмінний від нуля.

Приймач запам'ятовує втрачені зонди із останніх 100 UDP-зондів. Якщо було втрачено більше ніж один зонд, він визначає шлях, який має бути перевантаженим, інакше шлях вважається незавантаженим. Крім того, періодичне зондування шляху дозволяє приймачу підтримувати поточне середнє значення RTT.

Кожен тестовий пакет становить 100 Байт, і вони надсилаються кожні 10 мс, що забезпечує швидкість 80 Кбіт/с. Ці накладні витрати є незначними порівняно з перевагами пропускної здатності, які мужуть бути отримані.

Необхідно вимірювати пропускну спроможність прийому R. Періодично вимірюється пропускна здатність приймача на прикладному рівні як відношення кількості байтів, отриманих у послідовних часових вікнах довжиною $\Delta=2\times RTT$. Важливим є період вимірювання Δ. Якщо він занадто малий, і особливо якщо він менший за RTT передачі, результуючі вимірювання пропускної здатності будуть дуже шумними через затримки між стеком TCP і прикладним рівнем, а також через різку передачу TCP під час RTT.

З іншого боку, якщо Δ занадто велике, метод не матиме достатньо часу, щоб виявити, що доступна смуга пропускання насичена, перш ніж відбудеться переповнення буфера.

Приймач визначає дві умови: постійну швидкість і падіння постійної швидкості. Умова фіксованої швидкості є істинною, коли п’ять послідовних вимірювань пропускної здатності майже однакові, тобто коли пропускна здатність стала постійною. Для цих необхідних вимірювань постійної пропускної здатності достатньо лише два, а не п’ятьвимірювань, коли передача знаходиться на початковій фазі повільного запуску (стани 1 і 2 на рисунку 6.2). У поточній реалізації вимірювання пропускної здатності вважаються майже рівними, якщо їх нахил відносно часу менше половини

швидкості збільшення вікна перевантаження (cwnd). Для приймачів, які використовують відкладене підтвердження cwnd, підвищена швидкість становить приблизно половину MSS на RTT за час RTT. У точці фіксованої швидкості будь-яке подальше збільшення вікна надсилання спричиняє постійну затримку в вузькому місці та збільшення RTT.

Умова постійного падіння швидкості є вірною, з іншого боку, коли пропускна здатність прийому значно впала (більш ніж на 20%) після періоду часу (два вимірювання пропускної здатності), протягом якого вона була майже постійною (в межах 5% від один одного). Це (const-rate-drop) трапляється, коли алгоритму не вдається обмежити вікно надсилання до того, як цільова передача зазнає втрати пакета. Ця втрата іноді неминуча на практиці, особливо в шляхах з недостатньою буферністю. Однак алгоритм уникне будь-яких подальших таких втрат, обмеживши буфер приймача після першої втрати.

Падіння постійної швидкості очікується лише на шляху з недостатньою буферизацією, де алгоритм побачить втрату пакету до того, як він зможе наситити шлях достатньо довго, щоб виявити постійну швидкість.

Оскільки const-rate-drop відбувається, коли пакет даних скидається після того, як шлях був насичений протягом двох періодів вимірювання швидкості, дуже ймовірно, що це пов’язано з вікном перевантаження TCP. З іншого боку, у перевантажених каналах втрати виникають незалежно від вікна перевантаження TCP і відображатимуться в періодичних UDP-пакетах.

Перед встановленням з’єднання алгоритм встановлює максимальні значення для буферів надсилання та отримання, щоб мати достатньо великий коефіцієнт масштабування вікна. Тоді значення ssthresh (перевантажувального вікна) стає однаково великим, тому початковий повільний запуск може призвести до кількох втрат пакетів.

Такі втрати часто призводять до одного або кількох тайм-аутів, а також можуть спричинити значне зменшення ssthresh, уповільнюючи подальше збільшення вікна перевантаження.

Основна ідея полягає в тому, щоб спочатку обмежити розмір буфера приймального сокета на основі приблизної оцінки ємності C' прямого шляху. C' є результатом середньої дисперсії п’яти серії пакетів із використанням тестових пакетів UDP. Якщо пізніше передача стає обмеженою буфером, алгоритм періодично збільшує розмір буфера сокета на один максимальний розмір сегмента (MSS) у кожному RTT. Це лінійне збільшення повторюється до тих пір, поки одна з умов фіксованої швидкості або постійного падіння швидкості не стане істинною.

На рисунку 6.2 показана діаграма станів повного алгоритму налаштування буферу. Тут стан 2 представляє початкову фазу повільного запуску алгоритму.

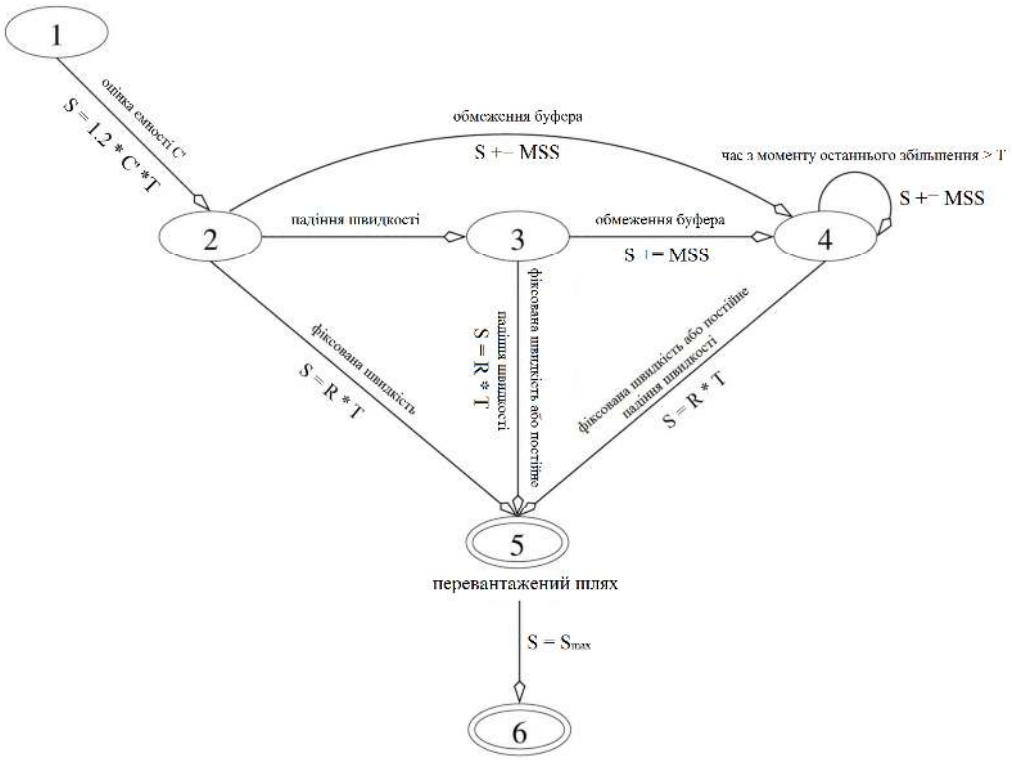

Рисунок 6.2 – Діаграма станів

Алгоритм також може вийти з повільного старту, якщо помітить падіння швидкості без дотримання фіксованої швидкості або обмеження буфером. Це показано переходом 2− >3 на рисунку 6.1. Це може статися

через втрати перед насиченням каналу. Слід зазначити, падіння швидкості не є таким самим, як const-rate-drop, оскільки воно не показує швидкість, яка буде насищати шлях. Стан 6 на діаграмі станів представляє кінцевий стан для перевантаженого шляху, який виводиться, коли алгоритм спостерігає втрати в періодичних тестах UDP, і може бути досягнутий із станів 2, 3 або 4. Загалом реалізація алгоритму становить приблизно 250 рядків коду C.

6.3 Затримка переповнення буфера

Затримка переповнення буфера (Buffer Overflow Latency – BOL) D_o – це період часу від моменту, коли з'єднання TCP насичує канал, до моменту, коли буфер цього каналу переповнюється вперше. BOL важливий, оскільки він визначає максимальний інтервал часу, протягом якого приймальна сторона повинна виявити умову фіксованої швидкості до того, як виникнуть втрати пакетів.

Розглянемо передачу TCP з початковим RTT T_0, обмеженим каналом ємності C і буфером B.

Припустимо, що пропускна здатність передачі R(t) досягає емності каналу в момент часу t_0, тобто $R(t_0)=C$.

Будь-яке наступне збільшення вікна передачі накопичується в буфері, що призводить до збільшення RTT. Нехай Q(t) – затримка в буфері в момент часу $t \geq t_0$. BOL – це мінімальний період часу D_o, для якого $Q(t_0+D_o) \geq B$.

RTT T(t) у момент часу t є функцією миттєвої затримки:

$$T(t)=T_o+\frac{Q(t)}{C}, \tag{6.1}$$

тоді як затримка Q(t) визначається як:

$$Q(t)=W(t)-CT_o=C[T(t)-T_o]. \tag{6.2}$$

Попереднє рівняння показує, що швидкість збільшення відставання дорівнює швидкості збільшення вікна:

$$\frac{dQ}{dt}=\frac{dW}{dt}, \tag{6.3}$$

що залежить від того, чи відбувається передача в режимі уникнення заторів (CA) чи повільного старту (SS).

Під час уникнення перевантажень вікно збільшується на один пакет на RTT (наразі ігноруючи затримані ACK). Таким чином:

$$\frac{dW}{dt}=\frac{L}{T}=\frac{LC}{W}. \tag{6.4}$$

З (6.3) можна побачити, що BOL можна визначити наступним чином:

$$\int_{CT_o}^{CT_o+B} WdW=\int_0^{D_o} LCdt. \tag{6.5}$$

Розв’язування попереднього рівняння дає нам BOL для уникнення заторів:

$$D_o^{CA}=\frac{B^2+2BCT_o}{2LC}. \tag{6.6}$$

Подібним чином, під час повільного запуску швидкість збільшення вікна дорівнює цілому вікну на RTT:

$$\frac{dW}{dt}=\frac{W}{T}=C. \tag{6.7}$$

Тому:

$$\int_{CT_o}^{CT_o+B} dW = \int_0^{RSL} Cdt, \tag{6.8}$$

що дає нам BOL у повільному старті:

$$D_o^{SS} = \frac{B}{C}. \tag{6.9}$$

При наявності відкладених ACK, швидкість збільшення вікна зменшується в два рази. У цьому випадку рівняння (6.6) і (6.9) слід замінити на:

$$D_o^{CA} = \frac{B^2 + 2BCT_o}{LC}, \tag{6.10}$$

та:

$$D_o^{SS} = \frac{2B}{C}, \tag{6.11}$$

відповідно. Слід зазначити, що $D_o^{SS} << D_o^{CA}$.

Попередні результати показують, що BOL значною мірою визначається співвідношенням «буфер до ємності» B/C, тобто максимальною затримкою в черзі на каналі.

Загальне правило надання буфера полягає в забезпеченні достатньої буферизації в каналі зв'язку, щоб співвідношення буфера до ємності дорівнювало максимальному RTT серед TCP-з'єднань, які проходять через цей канал. Наприклад, великий постачальник маршрутизаторів рекомендує B/C=500 мс. У цьому випадку (6.11) показує, що алгоритм вибору розміру

буфера має не більше однієї секунди під час повільного запуску, щоб виявити, що передача насичує доступну смугу пропускання, і обмежити розмір буфера сокета.

Як було зазначено вище, алгоритм вимірює отриману пропускну здатність кожні два RTT і таким чином, виявляє умову фіксованої швидкості після двох послідовних постійних вимірювань, коли він перебуває в станах (1) і (2) (рисунок 6.2). Таким чином, мінімальний період часу, протягом якого алгоритм може обмежити розмір буфера сокета, становить приблизно чотири RTT. Іншими словами, алгоритм ефективно запобігає втратам під час повільного запуску, якщо:

$$4T<D_o. \qquad (6.12)$$

6.4 Експериментальні результати

Алгоритм реалізовано як простий додаток, який дозує трафік у моделі в NS3 для передачі даних на основі TCP. Прототип був протестований у великій кількості сценаріїв моделювання.

Доступний діапазон пропускної здатності 10-1000 Мбіт/с. Для кожного сценарію порівнюється пропускна здатність, отримана від алгоритма, з пропускною здатністю, отриманою в результаті використання максимально дозволеного розміру буфера сокета. Останнє – це те, що програми передачі даних роблять, щоб максимізувати свою пропускну здатність.

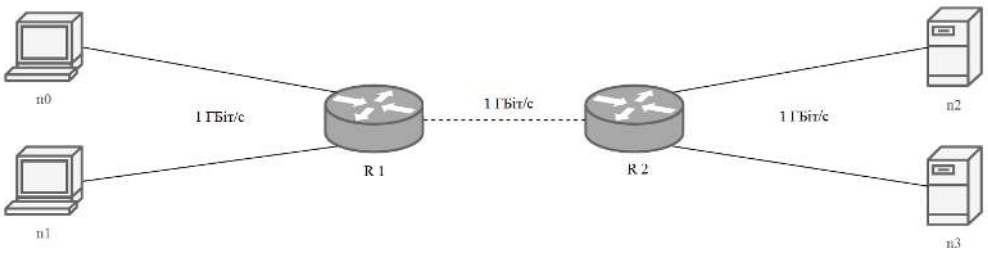

Рисунок 6.3 – Топологія мережі для моделювання

На рисунках 6.4 та 6.5 показано пропускну здатність і RTT для каналу 1 ГБіе/с. з і без алгоритму налаштування буферу. Середня пропускна здатність становить 918 Мбіт/с у першому та 649 Мбіт/с у другому. Об’єм передаваних даних ставить 200 Мбайт.

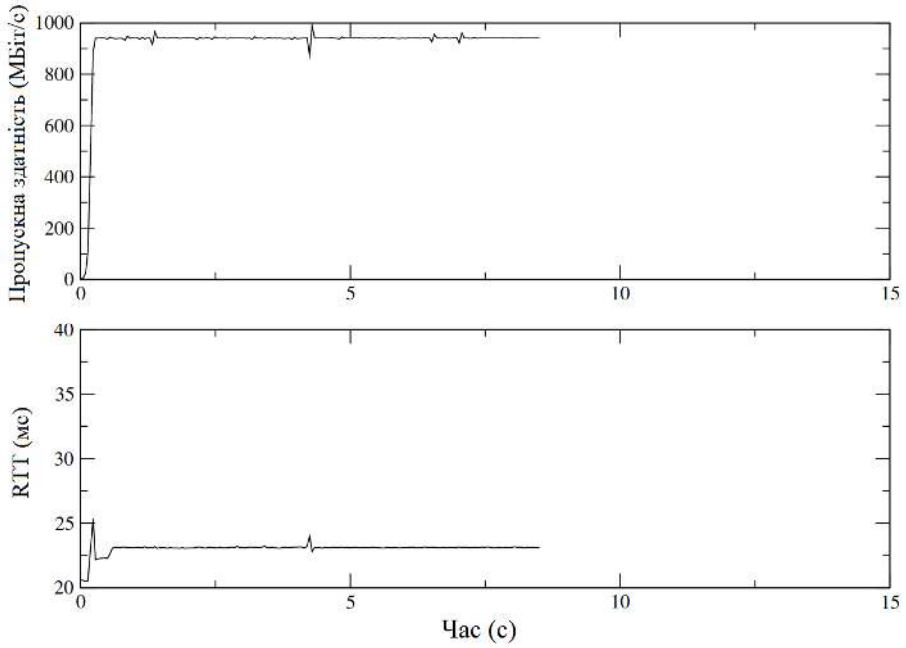

Рисунок 6.4 – З налаштуванням буферу, гігабітний канал без перехресного трафіку

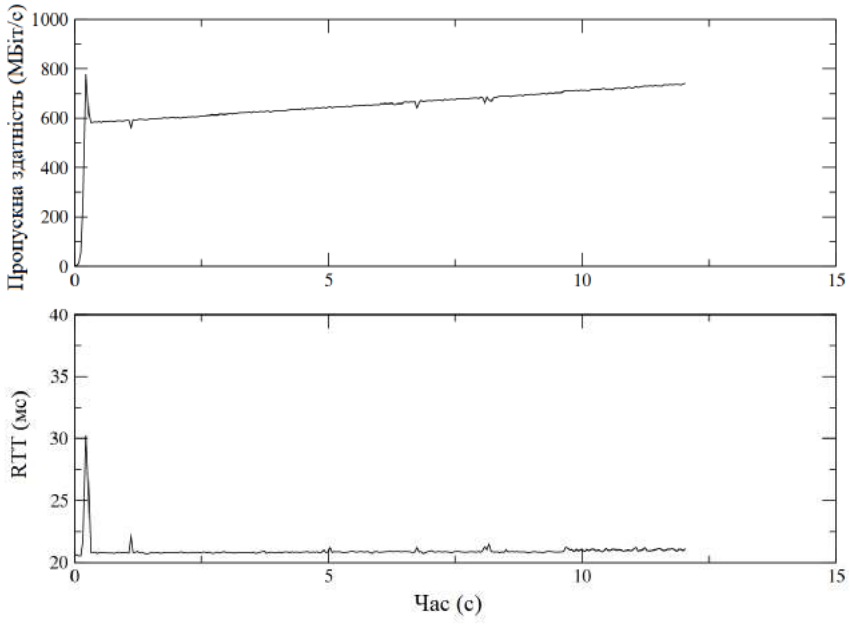

Рисунок 6.5 – Звичайна передача, гігабітний канал без перехресного трафіку

Аналіз трасування двох з’єднань показує, що потік, відмінний від потоку з алгоритмом, зазнав багатьох втрат під час початкового повільного запуску.

Після відновлення цих втрат передача почала повільну фазу уникнення перевантажень зі швидкістю приблизно 600 Мбіт/с, не досягнувши доступної пропускної здатності шляху. Алгоритм, з іншого боку, уникнув втрат при повільному старті за допомогою оцінки пропускної здатності на основі пакетної передачі.

Незабаром після цього, приблизно через 500 мс після початку передачі, алгоритм виявив умову фіксованої швидкості та встановив для S остаточне значення.

Далі розглянемо продуктивність алгоритму при перехресному трафіку, що не реагує на затори. Середня швидкість перехресного трафіку становить 400 Мбіт/с. На рисунках 6.6 та 6.7 показано пропускну здатність і RTT для цього каналу з і без алгоритму налаштування буферу відповідно.

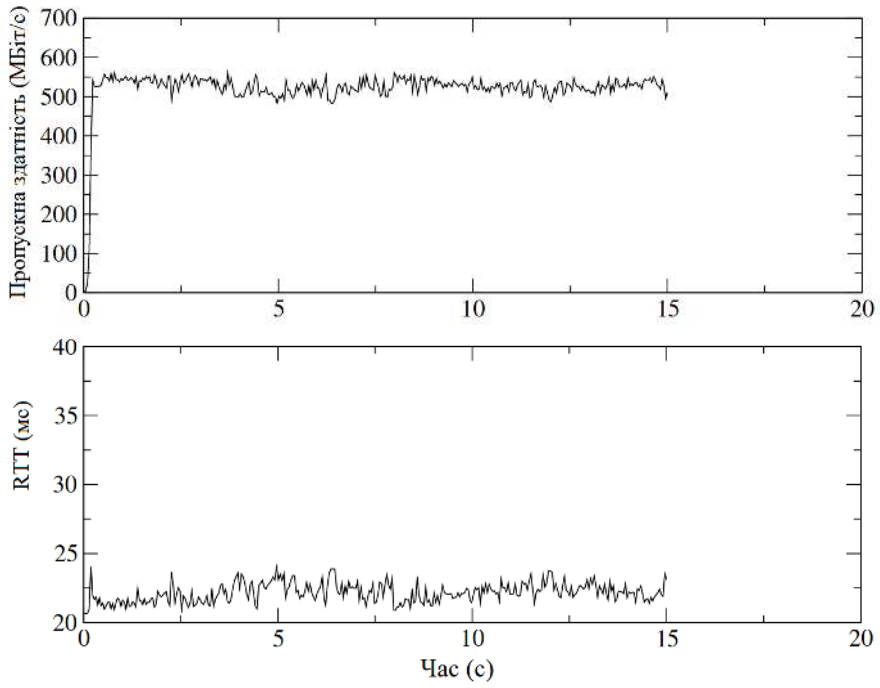

Рисунок 6.6 – З налаштуванням буферу, гігабітний канал з перехресним трафіком

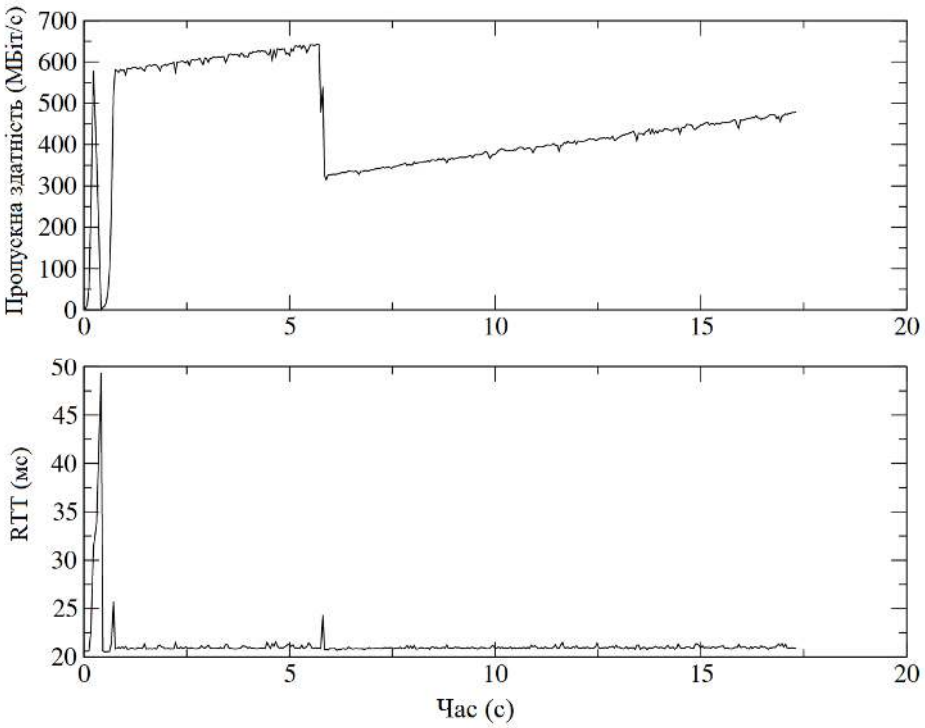

Рисунок 6.7 – Звичайна передача, гігабітний канал з перехресним трафіком

Є дві втрати пакетів в потоці, відмінному алгоритму. По-перше, початковий повільний старт спричинив значні втрати та кілька тайм-аутів, які фактично заглушили передачу приблизно на 2,5 секунди. Після того як втрати були відновлені, звичайний потік продовжував збільшувати своє вікно за межі доступної смуги пропускання.

У результаті RTT збільшився приблизно до 28 мс, потім передача зазнала ще більше втрат, а потім була повільна фаза уникнення перевантаження. Алгоритм налаштування буферу, з іншого боку, успішно визначив точку, в якій доступна пропускна здатність була насичена, і обмежив розмір буфера сокета до того, як виникнуть будь-які втрати.

Далі ми оцінюється алгоритм за допомогою перехресного трафіку на основі TCP, що реагує на перевантаження. Останній генерується за допомогою передавання TCP з обмеженим буфером, який не може отримати більше 500 Мбіт/с через розмір буфера сокета. На рисунках 6.8 та 6.9 показано пропускну здатність і RTT для цього каналу з і без налаштування.

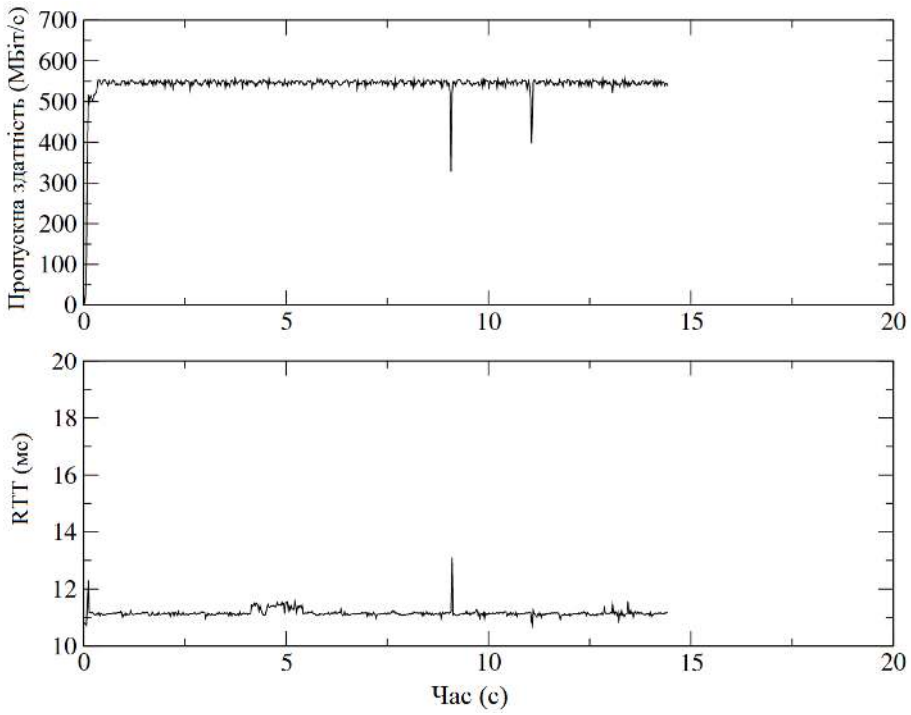

Рисунок 6.8 З налаштуванням буферу, гігабітний канал з перехресним трафіком, що реагує на перевантаження

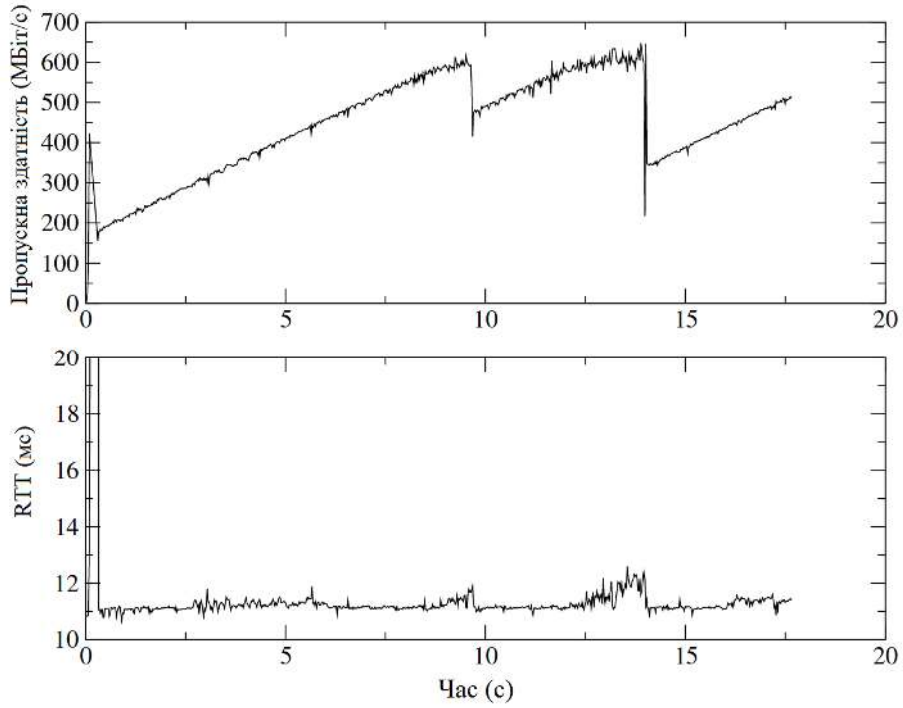

Рисунок 6.9 – Звичайна передача, гігабітний канал з перехресним трафіком, що реагує на перевантаження

Можна спостерігати, що звичайний потік зазнав втрат під час початкового повільного старту, хоча в цьому випадку вони були швидко відновлені. Ще одна втрата сталася приблизно через 6 секунд після початку передачі, що призвело до значного зниження пропускної здатності передачі. Алгоритм, з іншого боку, виявив умову фіксованої швидкості незабаром після початку передачі, уникаючи будь-яких втрат пакетів.

ВИСНОВКИ

Продуктивність TCP є однією з головних критичних проблем у сучасних комп’ютерних мережах, оскільки TCP не використовує максимальну пропускну здатність, а існуючі механізми пошуку більшої пропускної здатності TCP спричиняють самоіндуковані втрати в неперевантажених каналах. Такі втрати призводять як до низької наскрізної продуктивності, так і до втрати пропускної здатності зв’язку.

Значні покращення пропускної здатності TCP можна отримати без зміни протоколу TCP або його реалізації. В кваліфікаційній роботі представлено метод прикладного рівня, який покращує продуктивність TCP і потребує модифікацій лише на стороні адресату. Застосовується простий і важливий параметр для визначення розміру буфера маршрутизатора, який максимізує середню пропускну здатність TCP на потік.

Під час виконання кваліфікаційної роботи було виконано наступні завдання:

- досліджено причини низької продуктивності протоколу TCP;
- розроблено метод прикладного рівня, який автоматично встановлює розмір буфера кінцевих систем без попереднього знання будь-яких характеристик шляху, що дозволяэ наситити доступну смугу пропускання в мережевому тракті, не насичуючи вузький канал зв’язку на шляху;
- проведено імітаційне моделювання, яке підтвердило теоретичні викладки та показало значне покращення продуктивності у неперевантажених мережевих каналах.

Представлений в кваліфікаційній роботі метод надає можливість динамічно обирати оптимальні параметри передачі для отримання найвищої наскрізної пропускної здатності за допомогою існуючих протоколів і інструментів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Prasad, R. S. and Dovrolis, C., “Measuring the Congestion Responsiveness of Internet Traffic,” in the Proceedings of Passive and Active Measurements (PAM), 2007.

2. Shalunov, S. and Teitelbaum, B., Bulk TCP Use and Performance on Internet2, 2002.

3. Paxson, V., “Automated Packet Trace Analysis of TCP Implementations,” in the Proceedings of ACM Sigcomm, Sept. 1997.

4. Katabi, D., Handley, M., and Rohrs, C., “Congestion Control for High Bandwidth-Delay Product Networks,” in the Proceedings of ACM Sigcomm, Aug. 2002.

5. Dukkipati, N., Kobayashi, M., Zhang-Shen, R., and McKeown, N., “Processor Sharing Flows in the Internet,” in the Proceedings of Thirteenth International Workshop on Quality of Service (IWQoS), 2005.

6. Appenzeller, G., Keslassy, I., and McKeown, N., “Sizing Router Buffers,” in the Proceedings of ACM Sigcomm, 2004.

7. Floyd, S., Handley, M., Padhye, J., and Widmer, J., “Equation-Based Congestion Control for Unicast Applications,” in the Proceedings of ACM Sigcomm, 2000.

8. Lee, B. P., Balan, R. K., Jacob, L., Seah, W. K. G., and Ananda, A. L., “Avoiding Congestion Collapse on the Internet using TCP Tunnels,” Computer Net- works, vol. 39, no. 5, pp. 207–219, 2002.

9. Floyd, S., HighSpeed TCP for Large Congestion Windows, Dec. 2003. IETF RFC 3649 (experimental).

10. Kelly, T., “Scalable TCP: Improving Performance in Highspeed Wide Area Net- works,” ACM CCR, Apr. 2003.

11. Xu, L., Harfoush, K., and Rhee, I., “Binary Increase Congestion Control for Fast Long-Distance Networks,” in the Proceedings of IEEE 2004.

12. Brakmo, L. S. and L.L.Peterson, “TCP Vegas: End to End Congestion Avoidance on a Global Internet,” IEEE Journal on Selected Areas of Communications, vol. 13, Oct. 1995.

13. Semke, J., Madhavi, J., and Mathis, M., “Automatic TCP Buffer Tuning,” in the Proceedings of ACM Sigcomm, Aug. 1998.

14. Gardner, M. K., Feng, W.-C., and Fisk, M., “Dynamic Right-Sizing in FTP (drsFTP): Enhancing Grid Performance in User-Space,” in the Proceedings IEEE Symposium on High-Performance Distributed Computing, 2002.

15. Morris, R., “TCP Behavior with Many Flows,” in the Proceedings of IEEE ICNP, 1997.

16. Borman, D., Braden, R., and Jacobson, V., TCP Extensions for High Performance, May 1992. IETF RFC 1323.

17. Fredj, S. B., Bonald, T., Proutiere, A., Regnie, G., and Roberts, J. W., “Statistical Bandwidth Sharing: A Study of Congestion at Flow Level,” in the Proceedings of ACM Sigcomm, Aug. 2001.

18. Kelly, F. P., Maulloo, A., and Tan, D., “Rate Control in Communication Net- works: Shadow Prices, Proportional Fairness and Stability,” Journal of the Opera- tional Research Society, vol. 49, pp. 237–252, 1998.

19. Heyman, D., Lakshman, T. V., and Neidhardt, A. L., “A New Method for Analysis Feedback-Based Protocols with Applications to Engineering Web Traffic over the Internet,” in the Processings of ACM Sigmetrics, pp. 24–38, 1997.

20. Schroeder, B., Wierman, A., and Harchol-Balter, M., “Closed Versus Open: A Cautionary Tale,” in the Proceedings of USENIX NSDI, 2006.

21. Jacobson, V., “Congestion Avoidance and Control,” in the Proceedings of ACM Sigcomm, pp. 314–329, Sept. 1988.

22. Швецов К.О., Нечипоренко А.О., Янковський О.А. «Розробка нових методів для запобігання заторів та зниження затримок в мережах», Тринадцята міжнародна науково-технічна конференця «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління». – Баку-Харків-Жиліна-2023. – С. 94.