Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Алгоритми управління трафіком в IP-мережі

(тема)

Виконав:

студент II курсу, групи КСМзм-22-1

Вербицький Є.Р.
(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. Колтун Ю.М.
(посада, прізвище, ініціали)

Допускається до захисту

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

2024 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

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 Вербицькому Єгору Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Алгоритми управління трафіком в IP-мережі

затверджена наказом по університету від " 03 " листопада 2023 р. № 244 Стз

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

3. Вхідні дані до роботи 1) тип мережі – IP-мережа; 2) технологічна основа аналізу алгоритмів управління – алгоритми управління чергами, алгоритми планування черг; 3) мета управління – запобігання впливу і виникненню перевантажень; 4) формальний опис системи управління – модель СМО і теорія черг; 5) програмне середовище проведення моделювання – ПЗ GNS3; 6) додаткове програмне забезпечення та інструменти – ПЗ VirtualBox, Linux Xubuntu 14.04, утиліта ping, протокол ICMP.

4. Перелік питань, що потрібно опрацювати у роботі
1) обґрунтування підходів щодо організації системи управління трафіком в мережах з КП;
2) загальний опис особливостей алгоритмів управління і планування черг;
3) аналіз найпоширеніших алгоритмів управління чергами;
4) аналіз найпоширеніших алгоритмів планування черг;
5) обґрунтування базової моделі СМО для управління потоками трафіку в IP-мережах;
6) обґрунтування проведення моделювання на основі пакету GNS3;
7) моделювання та дослідження алгоритмів управління трафіком в IP-мережі;
8) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайд-презентація – 20 слайдів

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Обґрунтування підходів щодо організації системи		
	управління трафіком в мережах з КП;	07.11.23-10.11.23	
2	Загальний опис особливостей алгоритмів		
	управління і планування черг;	11.11.23-21.11.23	
3	Аналіз найпоширеніших алгоритмів управління		
	чергами;	22.11.23-30.11.23	
4	Аналіз найпоширеніших алгоритмів планування		
	черг;	01.12.23-08.12.23	
5	Обґрунтування базової моделі СМО для управління		
	потоками трафіку в IP-мережах;	09.12.23-17.12.23	
6	Обґрунтування проведення моделювання на	18.12.23-30.12.23	
	основі пакету GNS3		
7	Моделювання та дослідження алгоритмів		
	управління трафіком в IP-мережі;	31.12.23-12.01.24	
8	Подання кваліфікаційної роботи на рецензування	13.01.24-14.01.24	

Дата видачі завдання 06 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Колтун Ю.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 88 с., 22 рис., 3 табл., 2 дод., 24 джерела.

КОММУТАЦІЯ ПАКЕТІВ, IP-МЕРЕЖА, СИСТЕМА УПРАВЛІННЯ ТРАФІКОМ, АЛГОРИТМИ УПРАВЛІННЯ ЧЕРГАМИ, АЛГОРИТМИ ПЛАНУВАННЯ ЧЕРГ, CMO, TAILDROP, RED, WRED, PS, RR, DRR, PPS, GPS, CBWFQ, GNS3.

Метою кваліфікаційної роботи є аналіз і дослідження управління потоками трафіку на основі алгоритмів управління чергами та їх планування в IP-мережах в аспекті контролю і запобігання перевантажень.

У ході виконання кваліфікаційної роботи розглянута загальна концепція організації управління трафіком в IP мережах та проаналізовані особливості управління трафіком в мережах з КП, що базуються на алгоритмах планування і управління чергами. Зокрема проаналізовані найпростіший алгоритм пасивного управління чергами (TailDrop), один із різновидів алгоритму RED – зважений RED (WRED), а також один із різновидів алгоритмів планувальника WFQ, який базується на класах (CBWFQ). Проведений аналіз механізмів управління трафіком з використанням моделей на основі СМО, що описані в теорії масового обслуговування. Досліджені принципи функціонування алгоритмів управління та планування черг у IP-мережах із застосуванням інструментарію віртуального моделювання пакетної мережі на основі ПЗ GNS3.

THE ABSTRACT

Master’s thesis: 88 pages, 22 figures, 3 tables, 2 appendices, 24 sources.

PACKET SWITCHING, IP-NETWORK, TRAFFIC MANAGEMENT SYSTEM, QUEUE MANAGEMENT ALGORITHMS, QUEUE SCHEDULING ALGORITHM, MSS, TAILDROP, RED, WRED, PS, RR, DRR, PPS, GPS, CBWFQ, GNS3.

The purpose of the qualification work is to analyze and study the management of traffic flows based on queue management algorithms and their scheduling in IP networks in terms of controlling and preventing congestion.

In the course of the qualification work, the general concept of organizing traffic management in IP networks was considered and the features of traffic management in networks with control centers based on scheduling and queue management algorithms were analyzed. In particular, the simplest algorithm for passive queue management (TailDrop), one of the varieties of the RED algorithm – weighted RED (WRED), as well as one of the varieties of the class-based WFQ scheduler algorithms (CBWFQ) are analyzed. The analysis of traffic control mechanisms using models based on the CMO described in the theory of queuing is carried out. The principles of functioning of queue management and scheduling algorithms in IP networks are investigated using the tools of virtual modeling of a packet network based on software GNS3.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КП – комутація пакетів

ОС – операційної системи

ПЗ – програмне забезпечення

СМО – система масового обслуговування

СРР – справедливий розподіл ресурсів

СУ – система управління

ACL – списки управління доступом (англ., Access Control List)

API – прикладний програмний інтерфейс (англ., Application Programming Interface)

ATM – асинхронний режим передачі (англ., Asynchronous Transfer Mode)

CAC – управління доступністю з'єднання (англ., Connection Admission Control)

CBWFQ – алгоритм побудови черг, що базується на класах (англ., Class-Based Weighted Fair Queuing)

CoS – клас обслуговування (англ., Class of Service)

DRR – циклічна почерговість із дефіцитом часу (англ., Deficit Round Robin)

FIFO – першим прийшов, першим пішов (англ., First In First Out)

GPS – розподіл процесорного часу (англ., Generalized Processor Sharing)

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

LIFO – останнім прийшов, першим пішов (англ., Last In First Out)

NGN – мережа наступного покоління (англ., Next Generation Network)

OSI – еталонна модель взаємодії відкритих систем (англ., Open System Interconnection)

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

PFQ – справедлива буферизація черги на рівні пакетів (англ., Packet-by-Packet Fair Queuing)

PPS – спільне використання пріоритетного процесора (англ., Priority Processor Sharing)

PS – розподіл часу процесора (англ., Processor Sharing)

RAND – випадковий вибір заявок (англ., Random)

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

RR – циклічна почерговість (англ., Round Robin)

SLA – угода про рівень сервісу (англ., Service Level Agreement)

VM – віртуальна машина (англ., Virtual Machine)

WFQ – зважена справедлива буферизація (англ., Weighed Fair Queuing)

WRED – зважений алгоритм довільного раннього виявлення (англ., Weighted Random Early Detection)

ВСТУП

Із зростанням продуктивності комп'ютерної техніки та об'ємів переданого трафіку, збільшенням функціональності та складності мережних додатків, появою широкого і постійно зростаючого спектру різноманітних інфокомунікаційних послуг, – з’являється потреба в дієвих і все більш високопродуктивних мережних платформах. Впровадження сучасних інформаційних технологій у бізнес, дистанційну медицину (послуги телемедицини) та освіту всіх рівнів, а також географічна рознесеність мережних центрів зумовлюють розвиток концепції організації мереж нового покоління (Next Generation Network, NGN). Такі мережі з використанням процесів інтеграції та конвергенції об’єднують у єдине універсальне інформаційне середовище, що організується на основі технологій з розподіленою комутацією пакетів (КП), існуючі традиційні мережі та мережні технології, незалежно від їх функціонального призначення, структури та протоколів передачі [1].

Завдяки цим процесами відбуваються масштабні зміни у підходах щодо планування, проектування і побудови сучасних мереж, які орієнтовані на постійне підвищення пропускної здатності каналів і ліній зв'язку, на підвищення ефективності і продуктивності мережного обладнання. Такі підходи мають направленість на реалізацію можливостей щодо здійснення конвергенції різних видів трафіку, що передається, в єдину універсальну мережну інфраструктуру. Звідси можна бачити, що сучасні інфокомунікації, базовою платформою яких виступають мережі NGN, є складними комплексами на основі мережного обладнання і програмних додатків, які забезпечують вирішення багатьох задач щодо реалізації ефективного функціонування та управління [1].

Особливістю методу КП є одночасне використання каналів багатьма користувачами. При цьому фізичні тракти не організуються і жоден канал не виділяється у одноосібне володіння яким то двом конкретним системам користувача. По кожному із каналів передаються інформаційні блоки даних

по мірі їх надходження, які надсилаються різними користувачами з відповідним обладнанням. Тобто метод КП гуртується на тому, що інформаційне повідомлення ділиться на більш коротші інформаційні блоки даних, що отримали назву пакетів. Кожен з пакетів має адресну інформацією, яка розміщується в заголовку і передається за певним маршрутом [2].

Застосування методу на основі комутації пакетів для побудови сучасних мереж зв’язку надає такі переваги [2]:

- високу загальну пропускну здатність мережі у разі передачі пульсуючого трафіку;

- можливість динамічно перерозподіляти пропускну здатність фізичних каналів зв'язку між користувачами відповідно до реального обсягу трафіку.

Однак мережі з КП також мають низку недоліків, зокрема такі як [2]:

- неможливість точно визначити реальну швидкості передачі даних по мережі між користувачами, тому що ця швидкість залежить від затримок у чергах буферів комутаторів або маршрутизаторів, які у свою чергу залежать від загального завантаження мережі;

- затримки пакетів даних є змінними величинами і, крім того, вони можуть досягати великих значень у час виникнення миттєвих перевантажень мережі;

- у випадку переповнення буфера можливі бути втрати даних, оскільки обсяг буферів комутаторів і маршрутизаторів є обмеженим.

Для усунення цих недоліків розробляються механізми і застосовуються алгоритми управління потоками трафіка, що особливо актуально для трафіку, який є чутливим до затримок, та який орієнтований при цьому на постійну швидкості передачі (голос та відео). Різноманітні алгоритми управління трафіком в мережах з КП та їх дії у разі виникнення перевантаження будуть аналізуватися у цій роботі. Особливо потрібно звернути увагу на те, що на сучасному етапі розвитку технологій на базі методу передачі з КП та безлічі інфокомунікаційних послуг, що реалізуються на їх основі, відбувається в рамках мережі Інтернет, що орієнтована для передачі інформації на протокол IP. Тому аналіз алгоритмів управління трафіком, що наведений у цій роботі, орієнтований саме на IP-мережі, що також говорить про актуальність роботи.

1 ЗАГАЛЬНА КОНЦЕПЦІЯ ОРГАНІЗАЦІЇ УПРАВЛІННЯ ТРАФІКОМ В IP МЕРЕЖАХ

1.1 Узагальнена характеристика IP мереж і їх базових протоколів

Основне призначення IP-мережі полягає в організації міжмережної взаємодії автономних систем, які з'єднані між собою завдяки маршрутизаторам, що у цьому виступають у якості граничних шлюзів. Під автономними системами розуміють самостійні мережі, які перебувають під незалежним управлінням і використовують власні внутрішні алгоритми роботи. Основна задача, яку виконує IP-мережа, полягає в передачі даних між автономними системами через граничні шлюзи, за умови, що маршрути доставки повідомлень наперед невідомі, і таких можливих маршрутів може бути досить багато. В існуючій термінології пакет даних у процесі передачі від одного хоста (кінцевого вузла) до іншого хоста може пройти кілька автономних систем. Перевагами IP-мережі є поділ можливостей ресурсів мережі, поліпшення комунікаційних взаємодій та ефективний доступ до інформаційних ресурсів на основі швидкого і досить якісного прийняття рішень. В основі протокольної будови IP-мереж знаходиться стек TCP/IP [2].

Протокол IP у концепції еталонної моделі взаємодії відкритих систем (Open System Interconnection, OSI), яка описує взаємодію мережних пристроїв і протоколів, належить до протоколів мережного рівня з негарантованою доставкою пакетів без встановлення з'єднання. Пакети в класичній термінології IP протоколу називаються дейтаграмами, а, відповідно, метод їх передачі – дейтаграмним. Суть цього методу полягає в тому, що всі дейтаграми передаються та опрацьовуються мережею повністю незалежно, тобто відсутні зв'язки між окремими дейтаграмами, механізми контролю та відновлення втрачених дейтаграм, а також гарантії їх доставки [2, 3].

Задача щодо контролю за цілісністю повідомлень повністю покладається на протоколи транспортного рівня OSI. Стосовно стека TCP/IP, то ця задача забезпечується транспортними протоколами TCP і UDP. Якщо задачами мережного рівня є задачі здійснення управління взаємодією вузлів мережі під час обміну даними, то транспортний рівень відповідає за взаємодію прикладних процесів у цих вузлах. Для забезпечення управління на транспортному рівні потрібен номер порту (ідентифікатор прикладного процесу) та IP-адреса (ідентифікатор хоста), комбінація яких називається сокетом. Через сокет виконується управління потоком даних між процесами, що взаємодіють [3].

TCP-протокол створює гарантований потік даних між процесами, що встановили віртуальне з'єднання. З'єднання в рамках TCP являє собою набір характеристик, які визначають процедури обміну даними між процесами. Частина параметрів має бути незмінною, а деякі параметри можуть змінюватися, пристосовуючи параметри процедур до наявного стану мережі. UDP-протокол виконує негарантовану доставку даних без встановлення з'єднання між процесами хосту, що передає, і хосту, що приймає. Повідомлення вміщуються протоколом у поле даних однієї або декількох дейтаграм із певним ідентифікатором сокета. На стороні, що приймає, проводиться відновлення повідомлення з прийнятих дейтаграм [3].

Зазначимо, що стек TCP/IP у рамках організації міжмережної взаємодії, в якості основної своєї задачі, містить різноманітні протоколи мережних інтерфейсів, що здійснюють перетворення дейтаграм на пакети або інформаційні блоки різних інших мережних технологій. Також до стеку комунікаційних протоколів TCP/IP входять протоколи маршрутизації (RIP, OSPF та ін.), передачі службових повідомлень управління (наприклад, ICMP), протоколи перетворення мережних адрес автономних систем на IP-адреси (ARP і RARP), протокол підтримки символьних доменних імен (DNS), а також безліч інших протоколів, перелік яких постійно збільшується [3].

1.2 Загальні поняття та підходи щодо організації системи управління трафіком у мережах з комутацією пакетів

Організація ефективного управління трафіком є найважливішою задачею в будь-якій інфокомунікаційній мережі в рамках забезпечення заданої або гарантованої якості обслуговування (Quality of Service, QoS). Під управлінням трафіком найчастіше розуміють сукупність алгоритмічних засобів, що реалізовані у вигляді апаратно-програмного комплексу, і спрямовані на забезпечення роботи мережі зв'язку з необхідною QoS у разі використання її ресурсів [4].

З позиції топології та архітектури мережі найважливішими складовими системи управління (СУ) трафіком є мережне планування та оптимізація. Зокрема, мережне планування – це процес, що визначає топологію мережі та пропускну здатність її ліній зв'язку (каналів) із урахуванням обсягів трафіку, що планується передати, а також можливих темпів його зростання і т.ін. Оптимізація трафіку – це управління розподілом трафіку на діючій мережі [4].

У будь-якій пакетній мережі, де розв'язуються задачі забезпечення QoS, необхідно використовувати функцію управління доступністю з'єднання (Connection Admission Control, CAC), яка надає набір дій, що реалізуються мережею під час встановлення нового (або відновлення вже існуючого) з'єднання. Це робиться для того, щоб визначити, чи можлива його підтримка з необхідними характеристиками і QoS. З'єднання буде підтримуватись мережею, якщо запитані ресурси будуть вільні, а вимоги щодо QoS для існуючих з'єднань виконуються, і, при цьому, з'єднання, що створюється, на них не вплине. Важливою складовою процесу обробки пакетів для забезпечення QoS в IP-мережі є підтримка алгоритмів управління чергами в буферах мережного обладнання. Алгоритми управління чергами являють собою набір методів і механізмів, задачами яких є управління надходженням, зберіганням і подальшою передачею на обслуговування пакетів, що надходять на входи мережного пристрою. Вибір того чи іншого алгоритму управління чергою є досить складним завданням, тому що кожен із них

належить до різних типів, а ті, у свою чергу, мають свої переваги та недоліки. Враховуючи те, що трафік (і, як наслідок, вимоги до його QoS) є різнорідним, то доцільно застосовувати систему пріоритизації в процесі його обслуговування, тобто пакети з різними пріоритетами в маршрутизаторах необхідно обслуговувати в різних чергах [4].

Загалом управління та розподіл трафіку на мережі реалізується в такій послідовності [4, 5]:

- користувач надає мережі інформацію про характеристики навантаження потоку або з'єднання, яке буде здійснювати передачу даних у мережу протягом певного часу. Серед цих характеристик мають бути специфіковані наступні: пікова і середня швидкості передачі, максимальна допустима затримка, тощо;

- якщо мережа має достатню кількість ресурсів для забезпечення запитаних характеристик, то цей потік починає здійснювати передачу даних у мережу, інакше запит буде відкинуто;

- маршрутизатор робить класифікацію пакетів з метою встановлення приналежності до потоків і класів обслуговування, внаслідок чого стає можливим моніторинг навантаження кожного потоку та встановлення відповідності поточних значень характеристик тим, що були заявлені;

- мережа робить моніторинг навантаження, що надходить від цього потоку, і якщо значення її характеристик перевищують ті, що були задані на початку, то мережа вживає певних заходів щодо обмеження навантаження. Такі заходи називаються функціями «політики управління навантаженням».

В узагальненому вигляді СУ трафіком у пакетній мережі можна представити у вигляді трьох складових підсистем (рисунок 1.1) [6]:

- підсистеми укладення трафік-контракту між користувачем і мережею, відповідно до якого користувачеві надається необхідний набір послуг із необхідною якістю;

- підсистеми управління потоками пакетів від різних користувачів;

- підсистеми контролю за характеристиками трафіку.

У кожну з цих підсистем входять елементи, які підтримують механізми, що реалізують виконання більш конкретних функцій щодо управління трафіком.

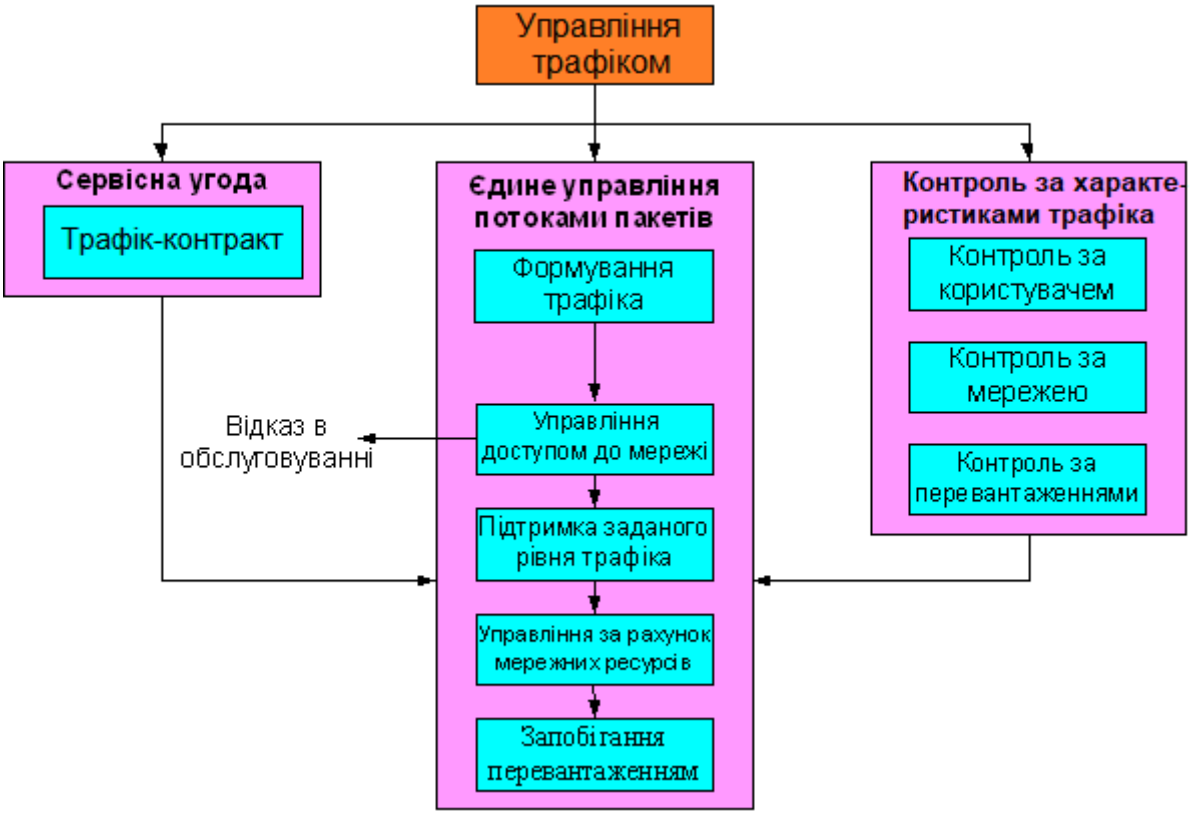


Рисунок 1.1 – Узагальнене подання СУ трафіком

Трафік-контракт користувача або угода про рівень сервісу (Service Level Agreement, SLA) визначається набором характеристик якості послуг, що надаються. Підсистема контролю за характеристиками трафіку включає елементи, що відповідають за здійснення контролю за діями користувачів і за мережею. Її основною задачею є контроль запобігання перевантаженням у мережі або її мережних пристроях. Зокрема тут вживаються заходи щодо зменшення часу тривалості перевантаження, а також щодо мінімізації його наслідків [6].

Загалом основні функції СУ трафіком, що приведена на рисунку 1.1, можна сформулювати таким чином [6]:

- укладання трафік-контракту, який буде прийнятним для більшості користувачів, що користуються ресурсами мережі;

- оптимальне надання ресурсів мережі користувачам і здійснення постійного управління ними для гарантованого виконання трафік-контракту, що був укладений;

- запобігання та/або мінімізація впливів від перевантажень.

Тобто можна бачити, що вся СУ трафіком орієнтована на надання якісних умов роботи всім користувачам і на попередження можливих перевантажень у мережі.

Перевантаження визначається як стан, у якому обладнання та елементи мережі (комутатори, маршрутизатори, фізичні канали зв'язку та ін.) не можуть забезпечити необхідні характеристики для всіх відкритих з'єднань, внаслідок чого основні показники QoS починають різко зменшуватися. Ці погіршення у мережі можуть проявлятися у збільшенні числа втрачених пакетів і значень величин різних затримок. Масштаб перевантаження може бути локальним (на конкретному пристрої або ж на окремих ділянках мережі) або глобальним (тобто проявлятися на всій мережі) [4].

Поява в мережі стану перевантаження призводить до різкого збільшення черг у буферах окремих мережних вузлів. У разі якщо відсутні механізми контролю виникнення перевантажень, або вони не є ефективними, то це може призвести до блокування доступу користувачів до мережних ресурсів на «ураженій» ділянці або ресурсів всієї мережі в цілому. У результаті параметр мережі, що визначає кількість обслужених пакетів (мережна продуктивність) буде різко падати, а затримки, навпаки – різко зростати [4].

1.3 Загальні особливості контролю характеристик трафіка для попередження перевантажень

Найчастіше перевантаження може бути спричинене порушенням характеристик потоків трафіку або виходом з ладу будь-якого мережного елемента, що також негативно позначається на характеристиках трафіку. Тому потрібно проводити вимірювання навантаження для кожного потоку трафіку, щоб визначити відповідність поточних значень щодо заявлених параметрів. Проведення моніторингу окремо кожного потоку не є можливим без попередньої класифікації пакетів. Така класифікація дасть змогу

визначити приналежність послідовності пакетів до відповідних потоків або класів обслуговування [4].

У роботі під потоком трафіку будемо мати на увазі послідовність пакетів, що унікально ідентифіковані на мережному рівні моделі OSI стосовно стека TCP/IP. Відповідно до принципів QoS, функція класифікації пакетів має бути також реалізована на мережному рівні моделі OSI. Робота цієї функції полягатиме в передачі пакета, який надійшов, у чергу певного класу обслуговування, на вході якої проводиться вимірювання характеристик потоку трафіку, до якого належить цей пакет [4].

Для динамічного контролю характеристик кожного з потоків трафіку відповідно до класів QoS необхідно здійснювати вимірювання навантаження і порівняння значень його параметрів з наявними. У разі, якщо пакет відповідає необхідним параметрам його буде обслужено, а в іншому випадку – пакет позначається або відкидається [4].

Існують різні підходи до реалізації такої політики управління потоками трафіку і вимірюванню навантаження, яке вони створюють. Наприклад, кожному потоку пакетів, що надходить до деякого маршрутизатора, для їх коректного опрацювання необхідно виділити певні ресурси: буферний простір, процесорний час, смугу пропускання вихідного каналу. Так, для надання гарантій, наприклад, щодо смуги пропускання, – для реальної IP-мережі необхідне виконання таких умов [4]:

- вхідне навантаження має відповідати смузі пропускання вихідного каналу, тобто, у разі, якщо буде перевищена встановлена межа, то вхідне навантаження має бути обмежено;

- не допускати переповнення буферів, тобто для кожного потоку пакетів визначається максимальне значення буферного простору, у разі перевищення якого пакети, наприклад, будуть відкидатися.

Для того щоб виконати першу умову, необхідно визначити, яким чином вимірюється швидкість потоків. Вона може бути визначена, наприклад, як кількість переданих пакетів або кількість переданих бітів за певний інтервал

часу. Вимірювання швидкості в кількості пакетів за певний інтервал часу має очевидний недолік для IP-мереж, який пов'язаний зі змінним розміром пакетів, що не дасть змоги здійснити точний підрахунок. Відповідно вимірювання швидкості в кількості біт за певний інтервал часу буде більш прийнятним, але тут в якості недоліку можуть бути складнощі, що пов'язані з вибором відповідного інтервалу часу, протягом якого здійснюється вимірювання швидкості потоку, тому що мілісекунда або хвилина можуть показати дуже істотну різницю [4].

Коли алгоритм вимірювання швидкості потоку трафіку сформований, то механізм обмеження вхідного навантаження може бути реалізований таким чином: під час надходження пакета в маршрутизатор визначається поточна швидкість цього потоку, і якщо ця швидкість не перевищує прописаного значення в угоді про трафік, то пакет надходить до буфера, інакше він буде скинутий. Найпростішим алгоритмом вимірювання швидкості потоків трафіку є алгоритм, за якого часова шкала ділиться на інтервали фіксованого розміру. У цьому випадку швидкість потоку вимірюється як кількість байт, що надійшли за інтервал часу. Недоліком такого алгоритму є неможливість правильної оцінки швидкості потоку трафіку з високою пачечністю [4].

Далі розглянемо особливості здійснення управління трафіком у пакетних мережах, що базуються на його плануванні.

1.4 Особливості планування трафіку та типи планувальників, що використовуються в мережах з КП

1.4.1 Класи та алгоритм функціонування планувальників трафіку

Як відомо, у мережах з КП використовується принцип розділення ресурсів у процесі передачі пакетів, наприклад, розділення смуги пропускання або буферного простору. При цьому ресурси між потоками

трафіку повинні розподілятися справедливо відповідно з певними алгоритмами. Наприклад, у разі забезпечення QoS в IP-мережах, різні потоки трафіку, що належать до різних класів обслуговування (Class of Service, CoS), розділяють ресурси маршрутизатора (буферний простір і час центрального процесора), через який вони проходять. Зазначимо, що алгоритми маршрутизаторів з розподілу процесорного часу реалізують механізми, які дають змогу планувати обслуговування пакетів, тому далі такі алгоритми називатимемо «планувальниками» (scheduler), а самі маршрутизатори, на основі яких реалізуються планувальники, називатимемо «системами» [4, 7].

Виділяють два основні класи планувальників [7]:

- планувальники, що працюють безперервно, особливістю функціонування яких є те, що навантаження завжди буде обслужене, якщо є вільний процесорний час. Іншими словами, якщо центральний процесор не завантажений і, якщо хоча б в одній черзі існує хоча б один пакет, то планувальник завжди передасть цей пакет на обслуговування;
- планувальники, що працюють із паузами, особливістю функціонування яких є те, що навантаження не завжди передається на обслуговування, навіть якщо є вільний процесорний час. Іншими словами, навіть якщо центральний процесор не завантажений і, якщо в одній із черг є хоча б один пакет, то він не завжди буде переданий планувальником на обслуговування.

Можна бачити, що основною задачею планувальника є визначення, з якої черги наступний пакет буде відправлений на обслуговування. Тому, саме завдяки планувальнику вирішується задача поділу смуги пропускання вихідного каналу і у відповідності до заданих параметрів QoS. Таким чином, планувальник для певного CoS має можливість забезпечити мінімальну доступну смугу пропускання. Відповідно з цим, у разі правильної реалізації планувальників у всіх маршрутизаторах, що стоять на шляху деякого потоку, з'являється можливість задати для цього потоку обмежене зверху значення затримки пакетів [7].

Слід зазначити, що в цьому випадку передбачається, що час безперервний, тобто реалізується планувальник, що належить до класу тих, які працюють безперервно. Алгоритм функціонування такого планувальника називається «розподіл часу процесора» (Processor Sharing, PS). Цей алгоритм здатний забезпечити принцип справедливого розподілу ресурсів (CPP) незалежно від механізмів управління перевантаженням і від поведінки джерел трафіку. Однак найважливішим недоліком алгоритму PS є неможливість його практичної реалізації, тому що в реальному маршрутизаторі можна реалізувати лише його апроксимацію. Це тому, що реальні системи є дискретними, тоді як модель алгоритму PS будується в безперервному часі [7].

Щоб розібратися у відмінності безперервного алгоритму PS від його апроксимації, що реалізується на практиці, проаналізуємо такий приклад. Припустимо, що на вході деякого маршрутизатора пакети, що мають постійну довжину, класифікуються і поміщаються у дві різні черги залежно від пріоритету. Припустимо, що ці черги містять по одному пакету довжиною L біт, тобто в деякий момент $t = 0$, коли звільняється процесорний час, обидві черги можуть передати свій пакет на обслуговування. Також припустимо, що швидкість обслуговування даних C дорівнює одному пакету на секунду. Таким чином, у моделі алгоритму PS з безперервним часом кожен пакет буде обслуговуватися із швидкістю 1/2 пакета на секунду, і передача двох пакетів триватиме 2 секунди, при цьому на кожен пакет (тобто чергу) буде виділено по 50 відсотків процесорного часу. З іншого боку, роблячи апроксимацію, буде очевидно, що необхідно спочатку повністю обслужити один пакет, і тільки після цього – другий. Таким чином, будемо бачити, що перший пакет, незалежно від того, в якій із черг він би не перебував, буде обслужено через одну секунду, а другий пакет, відповідно, через 2 секунди. Описаний вище приклад наведено на рисунку 1.2, на якому показано функції обслуговування пакетів для безперервного і дискретного випадків [4, 7].

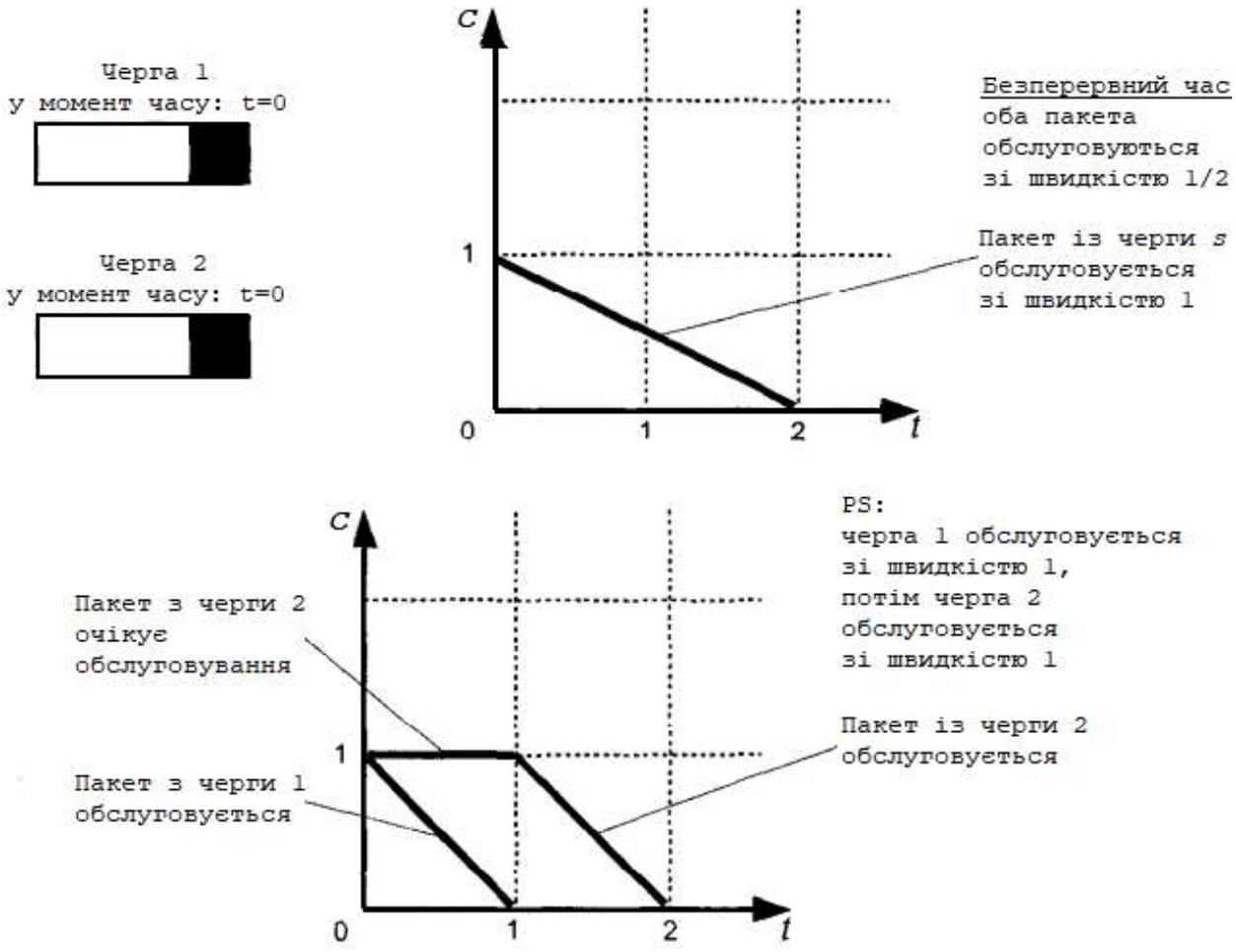


Рисунок 1.2 – Алгоритм PS та його апроксимація

1.4.2 Згладжування профілю трафіку на базі планувальника

Планувальники можна використовувати для згладжування пульсацій трафіку і приблизно задавати необхідну швидкість для деякого потоку на виході системи. Для цього встановлюється верхня межа значення смуги пропускання вихідного каналу для деякого з'єднання. Правильна конфігурація планувальника забезпечує, з одного боку, мінімальне значення інтервалу обслуговування, а з іншого – принцип справедливого розподілу ресурсів. Якщо пакет деякого потоку надходить у систему через інтервал часу, що буде меншим, ніж інтервал обслуговування, то цей пакет буде поміщений у буфер [4].

У реальних системах IP пакети досить сильно відрізняються за розміром, а також неможлива цілком точна реалізація принципу справедливого розподілу ресурсів, тому буде очевидним наявність впливу навантаження з різних черг один на одного. Ефект реалізації процедури згладжування трафіку на базі планувальника може полягати в можливості спрощення профілю трафіку для подальшої обробки іншими вузлами мережі [4].

1.4.3 Алгоритми планування черг пакетів без пріоритетів

Розглянутий вище алгоритм PS може бути апроксимований практично. Найпростішим рішенням є алгоритм «циклічна почерговість» (Round Robin, RR). Базовою ідеєю є забезпечення рівнозначного доступу кожної з черг до ресурсів процесора – щоразу, коли процесор звільняється, планувальник циклічно вибирає чергу, з якої приймається пакет на обслуговування. Цей алгоритм має істотний недолік – RR не забезпечує принцип CPP для випадків, коли пакети мають змінну довжину. Тому його застосування обмежене головним чином в обладнанні мереж на базі технології ATM [4].

Модифікацією алгоритму RR для пакетів змінної довжини є «циклічна почерговість із дефіцитом часу» (Deficit Round Robin, DRR). За принципом роботи DDR практично є аналогічним до алгоритму RR, проте він адаптований для обслуговування пакетів змінної довжини, у DRR додано функцію накопичення квантів виділеного процесорного часу. Квант часу - це час (порядку часток секунди) протягом якого алгоритм планування працює з потоком безперервно. Після закінчення часу, потік, що обслуговується, витісняється іншим. Коли настане черга знову обслуговувати цей потік, йому знову буде надано один квант часу. Таким чином, алгоритму планування завжди вистачатиме часу – він завжди буде наданий, поки потік знаходиться в обслуговуванні [4, 8].

Наприклад, припустимо, що для деякої черги виділено певну кількість процесорного часу для опрацювання пакета, який перебуває в ній. Однак

пакет виявився великого розміру і, відповідно, для його обробки потрібен більший кванту процесорного часу, ніж було виділено. Тоді цей пакет не буде обслужений, а під час наступного звернення планувальника до цієї черги, квант часу, що виділяється, буде просумовано з невикористаним, і в такий спосіб можуть бути обслужені пакети будь-якого розміру [4, 9].

1.4.4 Алгоритми планування черг пакетів із пріоритетами

Як зазначалося, застосування різних алгоритмів планування спрямоване на те, щоб реалізувати справедливий розподіл ресурсів між потоками трафіку, і водночас забезпечувати необхідний QoS щодо смуги пропускання і затримок для різних класів та типів трафіку. Це вимагає від планувальника, по-перше, реалізувати справедливий доступ до ресурсів, а, по-друге, забезпечити справедливу постановку пакетів у чергу. Існує безліч алгоритмів по справедливій постановці пакетів, що надходять у чергу, які стають найактуальнішими в момент часу, коли в чергу, де вже немає вільних місць для очікування, необхідно помістити новий пакет [4].

Ця задача стає ще складнішою, якщо пакети, що надходять на обслуговування, будуть різної довжини і з різним пріоритетом. Для простоти підтримки різних CoS у різні черги, що формуються в маршрутизаторі, поміщаються пакети з різних потоків, що дає змогу забезпечити незалежність різних потоків трафіку з різними вимогами щодо QoS. У цьому разі принцип CPP буде порушений, тому що якщо в одній із черг пакети будуть довшими за пакети з другої черги, то очевидно, що перша черга отримає вдвічі більше процесорного часу для обслуговування своїх пакетів [4].

Можливим виходом із ситуації, що склалася, може бути варіант, коли потоки, що належать різним класам QoS, отримуватимуть пропорційну кількість ресурсів, але надання різної кількості ресурсів необхідно теж регламентувати для дотримання все того ж CPP. Тут «справедливість» має визначатися відповідним CoS, до якого належить потік, що підлягає обробці,

а не довжиною пакета. Такий справедливий розподіл ресурсів під час обслуговування пакетів різної довжини, що належать різним CoS, трансформується в механізм справедливої буферизації черги на рівні пакетів (Packet-by-Packet Fair Queuing, PFQ). Цей механізм полягає в тому, що кожен пакет, незалежно від розміру, передається на обслуговування відповідно до вимог щодо QoS того класу, до якого цей пакет належить [9].

У даному аспекті, для забезпечення гарантій щодо затримок може бути використаний планувальник із пріоритетами, який реалізований на базі модифікованого алгоритму PS та заснований на спільному використанні пріоритетного часу процесору (Priority Processor Sharing, PPS). Його суть у тому, що кожній із черг присвоюється пріоритет, а планувальник виділяє процесорний час для обслуговування пакетів із черги відповідно до призначеного їй пріоритету. Наприклад, у маршрутизаторі реалізовано три черги A, B і C, у які поміщаються пакети трьох потоків із різними пріоритетами. Припустимо, що черзі A призначено вищий пріоритет, а черзі C – нижчий. При звільненні процесора планувальник перевіряє черги на наявність пакетів, що не були обслужені відповідно до пріоритетів цих черг. Якщо черга A вільна, то планувальник переходить до перевірки черги B, далі, якщо черга B вільна, то планувальник починає перевіряти чергу C. Перевагами такого алгоритму є простота реалізації та можливість забезпечення низької затримки для пакетів у потоці з високим пріоритетом. Недоліком є відсутність можливості виключення впливу потоків один на одного, тобто якщо пакети з високим пріоритетом будуть постійно надходити з високою інтенсивністю в чергу A, то пакети в чергах B і C будуть блокуватися. Інакше кажучи, можна бачити, що пріоритезація трафіку під час розроблення алгоритму планувальника може надати деякі переваги, якщо будуть задіяні механізми контролю кількості та інтенсивності пакетів, що надходять, а також буде реалізований гнучкий механізм пріоритезації доступу планувальника до черг маршрутизатора [9].

Далі розглянемо загальні особливості реалізації управління трафіком у мережах з КП, що ґрунтуються на алгоритмах управління чергами.

1.5 Загальні особливості та характеристики алгоритмів управління чергами

Основна задача функціонування алгоритмів управління чергами полягає у запобіганні виникненню або збільшенню ступеня перевантаження за рахунок наявності можливості скидання пакетів певних класів трафіку у певні моменти часу. Відповідно роботу таких алгоритмів оцінюють в аспекті їх можливостей щодо забезпечення ефективного контролю трафіку в період дії перевантаження. Зазвичай рішення щодо скидання пакета ухвалюють або під час його надходження в систему (маршрутизатор), або вже у разі виникнення перевантаження, коли можна скинути пакети, які вже розміщені в черзі для звільнення необхідного буферного простору для нових пакетів, що мають вищий пріоритет. Такі алгоритми поділяються на пасивні та активні, тому далі розглянемо їх загальні особливості та характеристики [4].

1.5.1 Пасивні алгоритми управління чергами

Найпростішим алгоритмом управління чергами і водночас найпоширенішим алгоритмом цього класу є TailDrop, в основі якого лежить дисципліна обслуговування черги за принципом «першим прийшов, першим пішов» (First In First Out, FIFO). Тут скидання пакета, що надходить на порт маршрутизатора, здійснюється тільки в разі, коли в черзі відсутні вільні місця. Зазначимо, що у разі появи в черзі вільного місця – туди стає пакет, який прийде першим з моменту появи вільного місця. Через простоту алгоритм TailDrop має кілька суттєвих недоліків, таких як

- черга може бути зайнята пакетами одного або декількох потоків, тобто порушується принцип CPP;
- заздалегідь не можна визначити момент настання перевантаження, тобто визначити його можна тільки після втрати першого пакета за рахунок переповнення черги [4].

Тобто черга тривалий інтервал часу може бути практично повністю завантажена, але джерела, що генерують трафік, про це не знатимуть. У цьому випадку надходження послідовності пакетів з будь-якого джерела, яке не сповіщене про наявність перевантаження або про можливу його появу, може призвести до втрати пакета або всієї послідовності пакетів [4].

Наявність буферів у маршрутизаторах вирішити питання з можливістю опрацювання пачкового трафіку. Головною задачею буфера відповідно до рекомендації RFC 2309 є можливість забезпечити приймання та обробку послідовності пакетів. Логічно припустити, що для того, щоб буфер міг прийняти послідовність пакетів досить великої довжини, необхідно збільшити розмір його вільного простору. Однак при цьому збільшується і час знаходження пакетів у черзі, тобто розмір буфера, з одного боку, має бути достатнім для прийому послідовності пакетів, а з іншого боку, не надто великим, щоб забезпечити необхідну QoS (стосовно обмеженого розміру затримки) [4].

З урахуванням проблем, що притаманні алгоритму TailDrop, було розроблено кілька функціональних механізмів, які покращують роботу цього алгоритму, серед яких можна виділити такі [4]:

- механізм імовірнісного скидання пакета, у разі якого у випадку переповнення черги здійснюється обчислення ймовірності, з якою деякий пакет скидається (Random Drop);

- механізм скидання початку черги, у разі якого у випадку переповнення черги здійснюється обчислення ймовірності, з якою перший пакет, що стоїть у черзі на обслуговування, скидається (Drop from Front);

Обидва механізми дають змогу уникнути проблеми захоплення черги пакетами одного або декількох потоків, однак залишається інша проблема, яка пов'язана з відсутністю можливості заздалегідь виявляти перевантаження. За певних умов ці алгоритми можуть скласти конкуренцію більш складним алгоритмам, що базуються на активному управлінні чергами [4].

1.5.2 Активні алгоритми управління чергами

Використання алгоритмів активного управління чергами дає змогу реалізувати такі переваги [4]:

- уникнути проблеми захоплення черги пакетами одного або декількох потоків;
- вирішити питання щодо виявлення перевантаження до його виникнення;
- забезпечити в буфері прийнятну затримку пакета;
- підвищити коефіцієнт використання мережних ресурсів.

Найбільш відомим і широко використовуваним алгоритмом цього класу є алгоритм, що забезпечує ймовірнісне завчасне визначення перевантаження або алгоритм випадкового раннього виявлення (Random Early Detection, RED). Цей алгоритм дає змогу контролювати трафік у межах черги маршрутизатора, а в разі виявлення стану, що може призвести до перевантаження, він робить так зване імовірнісне скидання пакетів, що здійснюється за допомогою обчислення ймовірності. У результаті буде дотримано справедливий розподіл ресурсів і можна здійснювати послідовне скидання пакетів, які належать різним потокам [4].

Функціонування та особливості алгоритму RED, як і інші алгоритми управління чергами і планування трафіку, більш докладно будуть розглянуті в наступних розділах.

2 АНАЛІЗ НАЙПОШИРЕНІШИХ АЛГОРИТМІВ УПРАВЛІННЯ ТРАФІКОМ В IP-МЕРЕЖАХ

2.1 Алгоритми управління чергами

2.1.1 Характеристика і особливості функціонування алгоритму TailDrop

Як вже було показано у попередньому розділі, алгоритм TailDrop є з одного боку найпоширенішим, а з іншого – найпростішим алгоритмом пасивного управління чергами. Принцип його функціонування узагальнено полягає в тому, що задається максимальний розмір черги, пакет, що надходить, поміщається в кінець черги. У випадку, якщо в черзі немає вільних місць, тобто вона заповнена, то пакет буде відкинений (рисунок 2.1) [4, 10].

Рисунок 2.1 – Принцип функціонування алгоритму TailDrop

Однак у разі використання транспортного протоколу TCP, коли пакети починають втрачатися, функціональні модулі TCP, які розташовані на робочих станціях, приймають рішення, що в мережі виникло перевантаження, і передають пакети більш повільно. Якщо черга заповнена, то може бути так, що кілька повідомлень будуть відкинуті одне за одним і, в результаті цього, ціла група додатків прийме рішення сповільнити передачу. Після цього додатки почнуть перевіряти мережу на предмет повноти її завантаженості, і, якщо буде отримано сприйнятливий прогноз, відновлять

передачу у початковому темпі, що знову може призвести до втрат повідомлень [4, 10].

Можуть виникнути ситуації коли алгоритм TailDrop може викликати ефект так звного «локауту» (lock-out). Зокрема, це може бути у випадках, коли черга монополізує один або декілька потоків пакетів, – або випадкових, або за необхідності синхронізованих потоків (наприклад, потоки, в яких передаються зображення та його звуковий супровід). Це буде перешкоджати стати в чергу пакетам інших потоків [10].

Функціонування алгоритму TailDrop призводить до того, що черги бувають заповненими на протязі тривалого інтервалу часу. Це стається тому, що TailDrop сигналізує тільки про повне заповнення черги. Зазначимо, що великі за своїми розмірами черги, впливають на збільшення часу доставки між робочими станціями. Тому треба контролювати, щоб середня довжина черг у маршрутизаторах була невеликою. З іншого боку, треба мати на увазі те, що трафік у IP-мережі є нерівномірним, і тому буфер у маршрутизатора має бути достатнього обсягу, щоб була можливість компесувати нерівномірність трафіку [10].

Як було зазначено, пакети, що надходять на вхідні порти маршрутизатора, отримують відмову тільки у разі заповнення буфера, тобто коли черга перевищить максимально допустимий її розмір ($q_{\max}$). Імовірність відмови в обслуговуванні – двопозиційна, тобто її значення може бути або 0, або 1. Звідси функція відкидання пакета для TailDrop має вигляд, що показаний на рисунку 2.2, а також може бути виражена наступним чином [11]:

$$d(q) = \begin{cases} 0 & q < q_{\max} \\ 1 & q > q_{\max} \end{cases}, \qquad (2.1)$$

де q – миттєве значення довжини черги.

Оскільки TailDrop, як зазначалося, сигналізує тільки про те, що черга переповнена, тому вона може досить тривалий час залишатися в такому

стані. При цьому рівень перевантаження мережі може зростати, тобто буде зростати довжина черги і, як наслідок, час доставки пакета буде збільшуватися. За рахунок того, що TailDrop має двопозиційний характер імовірності відмови в обслуговуванні, то він не може швидко адаптуватися до різких змін трафіку, що має пульсуючий характер. Черги, що створюються алгоритмом, мають схильність до нерівномірного розподілу потоків пакетів, що призводить до ефекту глобальної синхронізації [11].

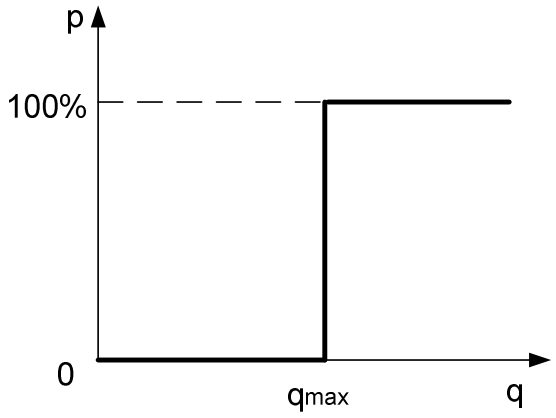


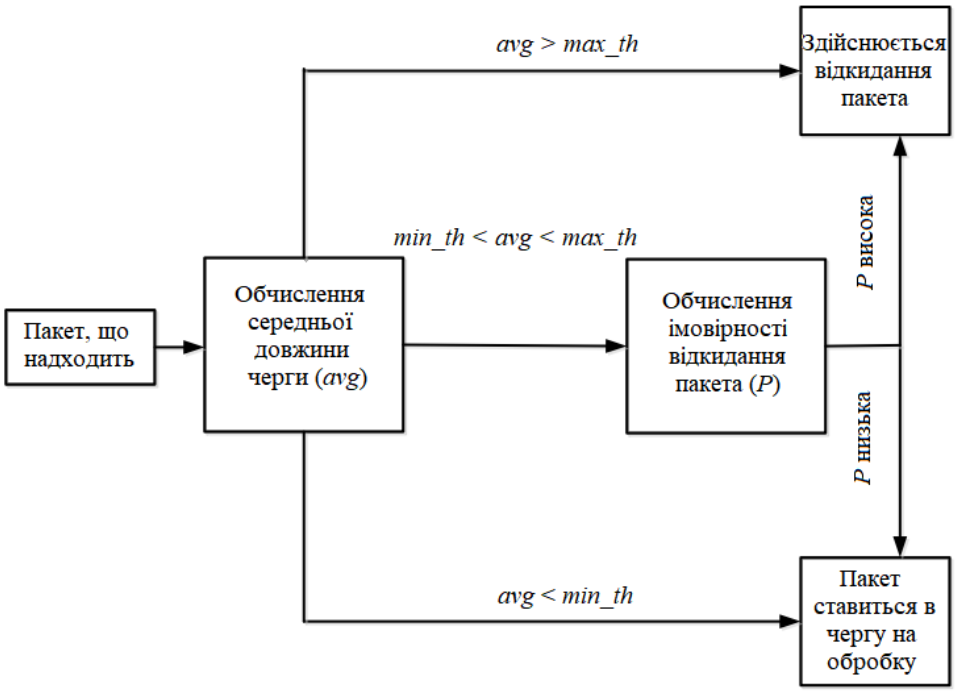
Рисунок 2.2 – Імовірність відкидання пакета в залежності від розміру черги для алгоритму TailDrop

2.1.2 Характеристика і особливості функціонування алгоритму RED

Також у попередньому розділі, узагальнено вже розглядався алгоритм випадкового раннього виявлення (RED), який відноситься до алгоритмів активного управління чергами. Він надає можливість здійснювати контроль трафіка в межах черги маршрутизатора. У разі виникнення і виявлення перевантажень або стану, що передує до перевантаженню, він може робити відкидання пакетів на основі визначення імовірності відкидання пакета [4].

Робота алгоритму RED будується на двох функціональних механізмах, зокрема механізму оцінювання середньої довжини черги та механізму приймання рішення щодо відкидання пакета, який надійшов. Згідно з

першим механізмом, алгоритм RED кожен раз при надходженні нового пакета дає змогу визначити середній розмір черги (*avg*), використовуючи фільтр високих частот сумісно з «експоненціально зваженою ковзаючою середньою» (low-pass filter with an Exponentially Weighed Moving Average, EWMA). Далі значення *avg* порівнюється із значеннями попередньо визначених границь порогів: верхнього (max_{th}) і нижнього (min_{th}), які окреслюють рівень навантаження в черзі. У випадку, коли значення *avg* належить до інтервалу [0, min_{th}], то пакет, що надійшов, розміщують у черзі. Після цього на основі обчислення імовірності (*P*) приймається рішення про його обслуговування або скидання. Це вже відповідає діям другого функціональному механізму алгоритму. Принцип функціонування алгоритму RED показано на рисунку 2.3[12, 13].



Позначення: *avg* – середня довжина черги;
max_th – верхня границя порогу відкидання пакета;
min_th –нижня границя порогу відкидання пакета;
P – імовірність відкидання пакета.

Рисунок 2.3 – Принцип функціонування алгоритму RED

Описаний вище алгоритм RED є базовим, тобто таким, що дає класичне розуміння принципів його функціонування. Далі проведемо аналіз розширеної версії алгоритму RED, яка застосовується для реалізації в маршрутизаторах у реальних IP-мережах.

Звернемо увагу на те, що RED крім відкидання пактів на основі аналізу імовірності *P*, також може маркувати пакети, що надходять. Це робиться коли значення *avg* перевищує мінімальну границю порогу min_{th}. Імовірність відмови зростає із зростанням *avg* аж до максимального значення імовірності відкидання пакета (max_p). Коли *avg* досягає значення максимальної границі порога max_{th}, усі пакети отримують відмову. Оскільки *avg* змінюється від min_{th} до max_{th}, то імовірність відкидання/маркування змінюється лінійно від 0 до max_p [11, 12]:

$$p_b = \frac{avg - \min_{th}}{\max_{th} - \min_{th}} \max_p . \tag{2.2}$$

Остаточна імовірність відкидання пакетів p_a потроху збільшується зі зростанням числа пакетів ($count$), які надійшли з моменту відкидання останнього пакета, що видно із виразу [11, 12]:

$$p_a = \frac{p_b}{1 - count \cdot p_b}. \tag{2.3}$$

Середня довжина черги (*avg*) розраховується щоразу після надходження нового пакета за формулою [11, 12]:

$$avg = (1 - w_q) \cdot \overline{avg} + w_q \cdot q, \tag{2.4}$$

де w_q – ваговий коефіцієнт черги.

Звідси, функція імовірності відкидання пакета набуває наступного вигляду, що описується виразом [11, 12]:

$$p = \begin{cases} \dfrac{avg - \min_{th}}{\max_{th} - \min_{th}} \cdot \max_p & \min_{th} < avg < \max_{th} \\ 0 & avg < \min_{th} \\ 1 & avg > \max_{th} \end{cases} . \qquad (2.5)$$

Залежність імовірності втрати пакета від середнього розміру черги для алгоритму RED показано на рис. 2.4 [12].

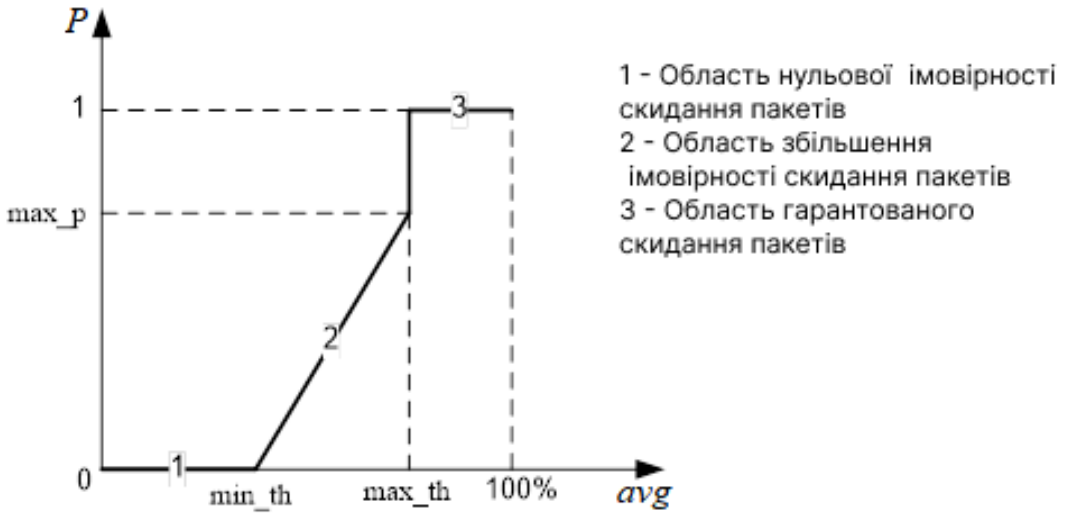


Рисунок 2.4 – Імовірність відкидання пакета в залежності від розміру черги для алгоритму RED

2.1.3 Характеристика і особливості функціонування алгоритму WRED

Зважений алгоритм довільного раннього виявлення (Weighted Random Early Detection, WRED) надає різні рівні обслуговування пакетів залежно від імовірності їх можливого відкидання і забезпечує вибіркову установку параметрів RED з урахуванням типу трафіку. Алгоритм WRED працює з єдиною чергою пакетів, для кожного типу трафіку задають власні параметри, і

обчислюють імовірність відкидання пакету. При цьому граничні порогові значення довжини черги для різних рівнів відкидання пакетів можуть повністю не збігатися, частково перекриватися або повністю співпадати [13].

Відмінність WRED від RED є лише в тому, що для пакетів кожного пріоритету призначається окремий набір характеристик. Наприклад, розглянемо дуже простий випадок, коли до черги маршрутизатора надходять пакети з двома пріоритетами: пакети із низьким пріоритетом, під якими будемо мати на увазі марковані пакети; і пакети із високим пріоритетом, під якими будемо мати на увазі звичайні немарковані пакет. Функція залежності імовірності відкидання пакета від значення *avg* поводиться так: якщо навантаження однакове, то має дотримуватися принцип, за якого імовірність відкидання маркованого пакета має перевищувати імовірність відкидання звичайного пакета [4, 14].

Параметризація алгоритму WRED доволі складна, тому що кількість значень, які необхідно встановити, прямо пропорційна кількості діючих пріоритетів, з якими пакети можуть передаватися до мережного маршрутизатора. Для розглянутого випадку, коли розглядається варіант функції імовірності відкидання пакетів двох пріоритетів за алгоритмом WRED показано на рисунку 2.5 [14, 15].

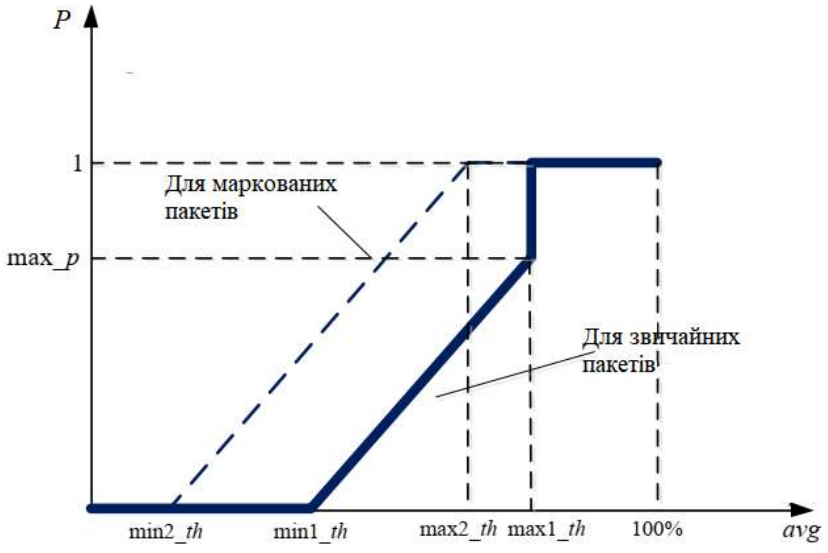


Рисунок 2.5 – Імовірність відкидання пакета для двох пріоритетів та в залежності від розміру черги для алгоритму WRED

2.2 Алгоритми планування черг

2.2.1 Характеристика і особливості функціонування алгоритму розподілу процесорного часу

Для здійснення справедливої пріоритизації доступу планувальника до черг маршрутизатора було створено алгоритм розподілу процесорного часу (Generalized Processor Sharing, GPS), який є модифікацією алгоритму PS, що був розглянутий у раніше у підрозділі 1.4 цієї кваліфікаційної роботи. Алгоритм GPS відноситься до класу планувальників, що працюють безперервно, і є ідеальним механізмом, що може забезпечити в безперервному часі принцип CPP відповідно із критерієм max-min для класів трафіку з різними вимогами щодо QoS [4].

Цей алгоритм може забезпечити низку можливостей [4]:

- надання необхідних гарантій на пропускну здатність для кожного потоку, як для одного конкретного планувальника, так і для їх послідовності;
- надання необхідних гарантій щодо затримок для потоків, які відповідають алгоритму «маркерного відра» (Token Bucket), як для одного конкретного планувальника, так і для послідовності планувальників;
- здійснює ізоляцію потоків один від одного, завдяки чому дії одного потоку не впливають на гарантії, що регламентуються іншим потокам.

Окремо треба зазначити, що за допомогою планувальника GPS неможливо обмежити значення джиттера для потоків, проте можна оцінити, що це значення перебуватиме в інтервалі [0, $D_{\max}$], де $D_{\max}$ – максимальна затримка пакета потоку, що розглядається [4].

Додатково, для реалізації механізму справедливої буферизації на рівні пакетів, алгоритм GPS для кожного пакета, що поступає у маршрутизатор, передбачає обчислення так званого часу закінчення обслуговування. Щоразу, коли обслуговування деякого пакета підходить до кінця, планувальник надає на обслуговування наступний пакет з деякої черги, для якого буде встановлено

найменше значенням часу закінчення обслуговування. Фактично на обслуговування надійде той пакет, який потрібно обслужити найшвидше [4].

Розглянемо механізми розподілу процесорного часу із застосуванням алгоритму GPS. Позначимо через $A_i(\tau,t)$ кількість інформації i-го потоку, що надійшла за інтервал часу $[\tau,t]$; через $W_i(\tau,t)$ позначимо кількість обслуговування, яке отримане i-м потоком (тобто кількість біт, що були обслужені, у i-му потоці) за той же проміжок часу; і через $Q_i(t)$ позначимо кількість трафіку i-го потоку, який перебуває в черзі в момент часу t. Таким чином, можна записати таке співвідношення [16]:

$$Q_i(t) = A_i(\tau,t) - W_i(\tau,t). \qquad (2.6)$$

Потрібно звернути увагу, що у разі перебування системи в стані простою в деякий момент часу, тобто у разі відсутності навантаження, то усі вище наведені параметри мають бути в стані «нуля». Приведемо ряд визначень [16]:

- «подія» – це надходження або відправлення на обслуговування потоку в рамках алгоритму GPS, що моделюється (далі будемо називати його «сервер GPS»). Позначимо через t_j момент часу, у який відбувається деяка подія j;

- «період зайнятості системи» – це інтервал часу, протягом якого сервер GPS перебуває у стані обробки трафіка;

- «беклог-період» для деякого i-го потоку – це безперервний інтервал часу, у період якого пакети цього потоку знаходяться у стані буферизації, і не передаються на обслуговування;

- «період зайнятості деякого i-го потоку» – це інтервал часу такої максимальної довжини $[\tau_1,\tau_2]$, що для будь-якого значення $t \in [\tau_1,\tau_2]$ пакети i-го потоку надходять із швидкістю, яка дорівнює або може бути вища за значення τ_i, тобто $A_t(\tau_1,t) \geq r_i(t-\tau_1)$.

Розглянемо як визначається значення параметру часу завершення обслуговування для деякого *k*-го пакета, що належить *i*-му потоку. Будемо мати на увазі, що реалізований принцип дотримання справедливої буферизації, та, що час безперервний, причому функціонування починається у момент часу $t = 0$. Позначимо через $V(t)$ кількість циклів, що проходить планувальник до моменту часу t. Тобто це значеннння можна виразити як відношення пропускної здатності каналу r (тобто швидкості обслуговування) до кількості активних черг [16]:

$$dV(t) / dt = r / n_{\text{activ}}(t). \tag{2.7}$$

Далі припустимо, що *k*-й пакет *i*-го потоку надходить у порожню чергу в момент часу $a_{i,k}$, а його довжина дорівнює $L_{i,k}$. Враховуючи той факт, що планувальник в кожний момент часу забирає на обслуговування з черги лише один біт, то буде очевидним той факт, що обслуговування цього пакета займе $L_{i,k}$ циклів. Тобто час завершення обслуговування пакета можна виразити наступною формулою [16]:

$$F_{i,k} = V(a_{i,k}) + L_{i,k}. \tag{2.8}$$

Час $F_{i,k}$ і є значення параметра час закінчення обслуговування. З іншого боку, якщо пакет надходить до черги, в якій перебуває один пакет (тобто черга непорожня), то у разі здійснення обчислення необхідно враховувати і час обслуговування цього пакета. Тоді можна записати такий вираз [16]:

$$F_{i,k} = F_{i,k+1} + L_{i,k}. \tag{2.9}$$

Поєднуючи вирази (2.8) і (2.9) отримуємо формулу, яка буде саме відбивати згадуваний вище принцип справедливої буферизації, а саме [16]:

$$F_{i,k} = \max \{F_{i,k}, V(a_{i,k})\} + L_{i,k}. \quad (2.10)$$

Слід звернути увагу, що параметр час закінчення обслуговування в цьому випадку не обов'язково точно буде дорівнює реальному часу завершення обслуговування пакета, тому що кількість процесорного часу, що виділяється для розглянутої черги, буде залежати від кількості активних черг [16].

2.2.2 Характеристика і особливості функціонування алгоритму зваженої справедливої буферизації

Розглянутий вище алгоритм GPS у реальному обладнанні реалізувати неможливо, проте в рамках маршрутизатора IP-мережі можна моделювати сервер GPS у безперервному часі на рівні пакетів і передавати на обслуговування пакет, у якому була обчислена функція віртуальний час закінчення обслуговування за допомогою алгоритму GPS у безперервному часі. Саме таким чином реалізується досить точна аппроксимація планувальника GPS, яка називається «зважена справедлива буферизація» (Weighed Fair Queuing, WFQ) [4, 13, 16].

Алгоритм планувальника WFQ визначається відповідно до принципів алгоритму GPS і реалізує принцип справедливої буферизації на рівні пакетів PFQ. Позначимо через d_p^{GPS} час закінчення обслуговування пакета *p*, інакше кажучи, час, коли центральний процесор передає пакет *p* у вихідний канал. Алгоритм, що аппроксимує GPS, має обслуговувати пакети відповідно до значення параметра d_p^{GPS}, дотримуючись стратегії, що пакет із меншим значенням цього параметра має бути обслугований раніше, ніж пакет із більшим значенням. Однак, на практиці таку стратегію поведінки планувальника реалізувати досить складно (це можливо тільки, якщо алгоритм планувальника належить до класу працюючих із паузами) [4].

Алгоритм планувальника WFQ для реалізації функції, що буде подібною до функції обчислення «віртуального часу закінчення обслуговування», яка реалізована в GPS, використовує підхід обчислення значення функції так званого «віртуального часу GPS», що розглядається нижче [4].

Позначимо час надходження першого пакета через $t_1 = 0$. Очевидно, що для $j = 2, 3,....$ кількість потоків, що перебувають у «періоді зайнятості» в інтервал часу $]t_{j-i}, t_j[$ буде фіксованою. Позначимо цю кількість через змінну B_j. Значення функції «віртуальний час GPS» $V(t)$ для будь-якого моменту часу, коли центральний процесор простоює, тобто на ньому відсутнє навантаження, дорівнює нулю. Далі проаналізуємо деякий період зайнятості, що починається в момент часу $t = 0$. Початкове значення функції «віртуальний час GPS» у цей момент дорівнює 0, тобто $V(0) = 0$, а далі його обчислення робиться відповідно до наступної формули [17]:

$$V(t_{j-1} + r) = V(t_{j-1}) + \frac{\tau}{\sum_{i \in B_j} r_i} \text{ для } \tau \le t_j - t_{j-1},\ j = 2,3,\ldots . \quad (2.11)$$

Швидкість зміни значення параметра V, тобто $dV(t_j + \tau)/d\tau$, обчислюється як відношення $1 \Big/ \sum_{i \in B_j} r_j$. Для кожного i-го потоку «беклог» швидкість обслуговування визначається як [17]:

$$r_i \cdot dV\left(t_j + \tau\right) \Big/ d\tau = g_i\left(t_j + \tau\right). \quad (2.12)$$

Таким чином, параметр V може трактуватися як швидкість, з якою обслуговуються «беклог»-потоки [17].

Нехай k-й пакет i-го потоку надходить у систему в момент часу $a_{i,k}$ та має довжину $L_{i,k}$. Зробимо задачу трохи складнішою, для чого введемо дві

нові функції: «віртуальний час надходження» пакета в чергу ($S_{j,k}$), і функцію закінчення обслуговування, що має назву «віртуальний час закінчення обслуговування» ($F_{j.k}$). Поклавши $F_{j0} = 0$ для всіх i, будемо мати такі відношення [17]:

$$S_{i,k} = \max\{F_{i,k-1}, V(a_{i,k})\}, \tag{2.13}$$

$$F_{i,k} = S_{i,k} + \frac{L_{i,k}}{r_i}. \tag{2.14}$$

Введення $V(a_{i,k})$ у цей вираз необхідне для скидання значення $S_{j,k}$ після того, як i -та черга стала активною, тобто в порожню чергу почали надходити пакети. Саме цьому досягається досить точна аппроксимація GPS в аспекті здійснення обчислення часу початку обслуговування пакета, що надійшов у порожню чергу [17].

З урахуванням вищевикладеного можна сформулювати переваги щодо використання функції «віртуальний час GPS» для роботи WFQ-планувальника з точки зору можливої реалізації на практиці [4, 17]:

- обчислення значення цієї функції можливе в момент надходження пакета в чергу;

- пакети обслуговуються послідовно по черзі у відповідності із значенням цієї функції;

- у процесі функціонування необхідно лише робити обновлення значення цієї функції, тоді як у алгоритмі GPS необхідно робити обробку подій.

В якості недоліку розглянутого алгоритму зазначимо накладні витрати, що пов'язані із зберіганням і управлінням множиною значень B_j, яка необхідна для здійснення обновлення значень функції «віртуальний час GPS» [4, 17].

Далі розглянемо інші планувальники, ідея побудови яких ідентична WFQ, а відмінність полягає у алгоритмах аппроксимації GPS, що застосовуються і в принципах вибору пакетів.

2.2.3 Загальна характеристика алгоритму побудови черг, що базуються на класах потоків трафіку

Подальшим розвитком алгоритму WFQ є формування класів потоків трафіка, що задаються користувачем. У результаті був створений алгоритм побудови черг, що базуються на класах (Class-Based Weighted Fair Queuing, CBWFQ). Цей алгоритм, як і всі інші, що були розглянуті у цьому розділі, також забезпечує управління трафіком в аспекті контролю перевантажень. На відміну від алгоритму WFQ, тут можна в достатньо широких межах перерозподіляти смугу пропускання між потоками трафіку [18].

Для виділення того чи іншого класу трафіка можна застосовувати так звані списки управління доступом (Access Control List, ACL) або також це можна робити через номер вхідного інтерфейсу. Кожному класу потоку пакетів ставиться у відповідність своя черга. У загальному випадку можна сформувати 64 класи, причому для кожного класу максимальний розмір черги буде залежати залежить від конкретної моделі маршрутизатора. Нерозподілена смуга пропускання може використовуватися потоками трафіка відповідно до їх пріоритетів. Така класифікація потоків робиться на підставі будь-яких критеріїв, які доступні для того чи іншого CoS. У разі потреби, для відкидання пакетів при виникненні перевантажень використовуються розглянуті вище алгоритми TailDrop або WRED, які налаштовуються для кожної черги у відповідності із класом, пекти якого до неї входять. Обслуговування цих черг відбувається з урахуванням виділеної смуги пропускання під кожну чергу [18].

Для здійснення подальших досліджень будемо використовувати алгоритми, що відносяться до активного і пасивного управління чергами (TailDrop і WRED відповідно) та алгоритм планувальника CBWFQ.

Із проведеного вище аналізу управління трафіком з метою уникнення перевантажень на основі алгоритмів управління чергами та їх планування, можна бачити, що в основі їх роботи лежить поняття «дисципліна обслуговування», яка визначає порядок вибору пакетів (заявок) із числа тих, що надійшли, і порядок розподілу їх між вільними каналами (портами, пристроями). Оптимальний вибір такої дисципліни обслуговування, а отже, і конкретного алгоритму управління трафіком багато в чому залежить від реальних мережевих параметрів, основними з яких є наступні:

- пропускна здатність каналів зв'язку;
- показники ефективності роботи мережі, такі як середнє число заявок, що обслуговуються за одиницю часу; середнє число заявок у черзі; середній час очікування обслуговування; імовірність відмови в обслуговуванні без очікування; імовірність того, що число заявок у черзі перевищить певне значення, тощо;
- кількості каналів зв'язку, тощо.

Зазначимо, що оцінка характеристик функціонування цих алгоритмів за певних умов роботи IP-мережі та пошук параметрів, які будуть оптимальними за деякими окремими критеріями, базуються, як правило, на раціональному виборі структури відповідної системи масового обслуговування (СМО). Такою СМО може виступати як окремий мережний пристрій (наприклад, маршрутизатор), що здійснює обслуговування потоків (заявок, пакетів), які надходять на його порти, так і група таких пристроїв, що об'єднані в одній мережній структурі. Тому далі проведемо аналіз механізмів управління трафіком з використанням моделей на основі СМО, що описані в теорії масового обслуговування (ТМО).

3 АНАЛІЗ МЕХАНІЗМІВ УПРАВЛІННЯ ТРАФІКОМ З ВИКОРИСТАННЯМ МОДЕЛЕЙ НА ОСНОВІ СИСТЕМ МАСОВОГО ОБСЛУГОВУВАННЯ

3.1 Загальна характеристика і основні визначення систем масового обслуговування стосовно аналізу потоків трафіка

На цей час теорія масового обслуговування (ТМО) широко застосовується в різних галузях науки і техніки, тому що дозволяє дослідити поширені на практиці ситуації, коли є деякий обмежений ресурс і багато запитів (потоків) на його використання. Наслідком такої взаємодії «ресурс-запит» можуть бути затримки або відмови в обслуговуванні деяких запитів, і тому виникає потреба у розумінні об'єктивних або суб’єктивних причин цих затримок чи відмов, а також з’ясування механізмів щодо зменшення їх впливу [19].

Як правило, запити надходять у випадкові моменти часу, а для здійснення їх обробки потрібен випадковий час використання ресурсу. Тому процеси обслуговування зазвичай досліджують в рамках теорії випадкових процесів із застосуванням досить складного математичного апарату. Важливим етапом у застосуванні ТМО для дослідження функціонування реального об'єкта є його формальний опис в термінах тієї або іншої системи масового обслуговування (СМО). Будь яка СМО буде вважатися заданою, якщо повністю будуть описані її компоненти, такі як [19, 20]:

- вхідний потік запитів (заявок, вимог, повідомлень, викликів);
- кількість і типи пристроїв (приладів), що здійснюють обслуговування;
- ємності буферів, де запити очікують початку обслуговування, оскільки всі прилади зайняті;
- часи обслуговування запитів у пристроях;
- дисципліна обслуговування (вона визначає порядок опрацювання запиту в системі, починаючи з моменту його надходження в систему і до моменту, коли він залишає СМО).

Параметри продуктивності функціонування СМО багато в чому визначає вхідний потік запитів. Тому правильний опис потоку запитів, що поступають у випадкові моменти часу в реальну систему, та здійснення ідентифікації його параметрів є досить важливим завданням. Випадковий потік можна вважати заданим, якщо є заданим спільний розподіл величин τ_k, $k = 1,\ldots,n$ для будь-якого n, $n \geq 1$ або заданий спільний розподіл величин x_t для усіх значень t, $t > 0$. Тут параметр n, $n \geq 1$ задає число ідентичних обслуговуючих приладів; величина τ_k, $k = 1,\ldots,n$ задає довжину інтервалу між моментами надходження (k – 1)-го і k-го запитів; величина x_t – це число моментів t_k, що лежать на осі часу ліворуч точки t, $t > 0$ [19].

Випадковий потік називається стаціонарним, якщо для будь-якого цілого числа m і будь-яких невід'ємних чисел $u_1,\ldots,u_m$ спільний розподіл величин $(x_{t+u_k} - x_t)$, $k = 1,\ldots,m$ не залежить від величини t. Тут параметр m, $m \geq 1$ задає число місць у буфері. Це означає, що розподіл числа запитів, що надійшли у деякий інтервал часу, залежить від довжини цього інтервалу, але не залежить від розташування цього інтервалу на часовій осі [19].

Випадковий потік називається ординарним, якщо для будь-якого t має місце вираз [19]:

$$\lim_{\Delta \to 0} \frac{P\{x_{t+\Delta} - x_t > 1\}}{\Delta} = 0. \qquad (3.1)$$

Тобто із виразу (3.1) слідує, що імовірність надходження більше однієї заявки за малий інтервал часу є величиною вищого порядку малості порівняно з довжиною інтервалу. Іншими словами, мається на увазі практична не можливість одночасного надходження двох або більше запитів [19].

Випадковий потік є потоком без післядії, якщо числа заявок, що надійшли на інтервалах часу, які не перетинаються, є у сукупності незалежними випадковими величинам [19].

Випадковий потік називається потоком з обмеженою післядією, у якому величини τ_k, $k \geq 1$ незалежні в сукупності [19].

Випадковий потік називається рекурентним потоком, якщо потік є потоком з обмеженою післядією, де величини τ_k, $k \geq 1$ однаково розподілені. Функцію розподілу потоку будемо позначати $A(t) = P\{\tau_k < t\}$. Зазначимо, що $A(t)$ повністю характеризує рекурентний потік [19].

Інтенсивністю λ стаціонарного випадкового потоку називається математичне очікування (середнє значення) числа запитів, що надходять у потоці за одиницю часу [19]:

$$\lambda = M\{x_{t+1} - x_t\} = \frac{M\{x_{t+T} - x_t\}}{T}, \quad T > 0. \tag{3.2}$$

Параметром α стаціонарного випадкового потоку називається позитивна величина, яка визначається співвідношенням:

$$\alpha = \lim_{\Delta \to 0} \frac{P\{x_{t+\Delta} - x_t \geq 1\}}{\Delta}. \tag{3.3}$$

Стаціонарний ординарний потік без післядії називається найпростішим. Для нього будуть справедливими такі твердження [19]:

- для того, щоб потік був найпростішим, необхідно і достатньо, щоб він був стаціонарним пуасонівським;

- для того щоб потік був найпростішим, необхідно і достатньо, щоб він був рекурентним із показовим розподілом довжин інтервалів між моментами надходження запитів;

- якщо є відомим, що на інтервалі, який має довжину *T,* надійшло *n* запитів найпростішого потоку, то імовірність надходження фіксованого запиту на частині цього інтервалу, що має довжину τ, не буде залежати від того, коли надходили інші запити, і буде дорівнювати τ / T.

- для найпростішого потоку параметр потоку та його інтенсивність мають співпадати. Середнє число запитів, які надходять на інтервалі часу, що має довжину T, буде дорівнює λT;

- потік, який отриманий у результаті здійснення накладення двох незалежних найпростіших потоків, що мають інтенсивності λ_1, λ_2 відповідно, є найпростішим потоком з інтенсивністю $\lambda_1 + \lambda_2$;

- потік, який був отриманий із найпростішого потоку з інтенсивністю λ у результаті застосування найпростішої процедури рекурентного просіювання, де довільний запит потоку з імовірністю p буде включеним у просіяний потік, а з імовірністю $1-p$ буде проігнорованим, є найпростішим потоком з інтенсивністю $p\lambda$.

- потік, що був отриманий у результаті суперпозиції n незалежних рекурентних потоків, які мають рівномірно малу інтенсивність, у разі збільшенні числа n сходиться до найпростішого потоку.

3.2 Базова модель СМО та її застосування для різних систем управління потоками трафіком

Одним із напрямів застосувань СМО є аналіз і моделювання алгоритмів управління чергами в мережі у разі взаємодії багатьох центрів обслуговування і потоків повідомлень (пакетів). Тут СМО є сукупністю скінченної кількості M центрів (маршрутизаторів), що здійснюють обслуговування, у якій циркулюють повідомлення (пакети), які відповідно до матриці маршруту просуваються з одного центру в інший. Узагальнений приклад реалізації такої СМО наведено на рис. 3.1. Тут кінцеве число повідомлень (N) у відповідності із імовірностями P_i, де $i = \overline{1, M}$, по черзі звертаються до одного з M центрів обслуговування. Центр обслуговування 1 здійснює моделювання роботи центрального процесора (ЦП), а у інші центри (2, 3, ..., M) представляють групу запам'ятовуючих пристроїв (наприклад,

буфери маршрутизатора). Узагальнено під центром обслуговування будемо мати на увазі СМО, що складається з A однакових приладів ($1 \le A \le \infty$) та буфера, що має деякий об'єм C ($0 \le C \le \infty$). Поки що далі будемо мати на увазі, якщо значення C не є оговореним, то C = ∞ [19, 21].

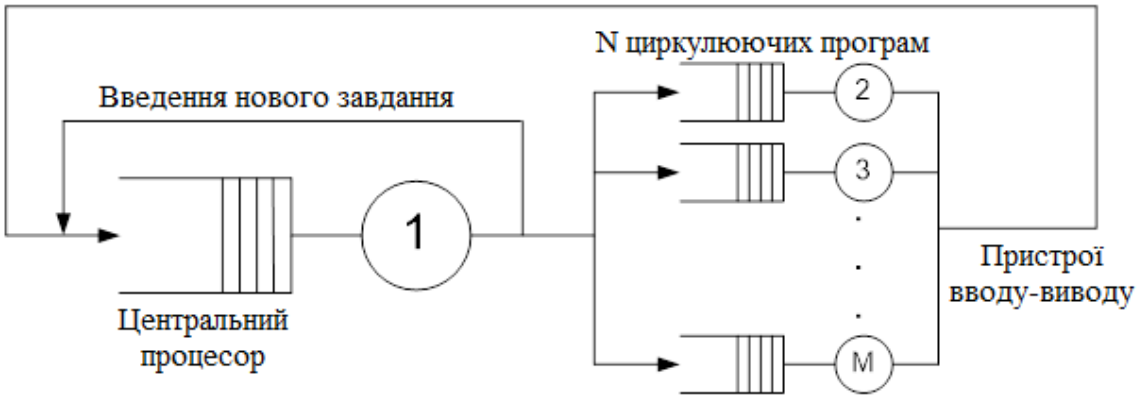


Рисунок 4.1 – Варіант реалізації узагальненої моделі СМО для управління трафіком

Якщо в момент надходження повідомлення всі прилади, що здійснюють обслуговування центру, зайняті, то повідомлення надходить у чергу в буфері, де очікує початку обслуговування. Тобто наявність черг – це неминучий наслідок стохастичного характеру надходження та обслуговування повідомлень у центрі [21].

Перехід повідомлення з одного центру в інший після закінчення обслуговування, виконується згідно із заданим маршрутом, під яким розуміється послідовність центрів мережі, що відвідуються повідомленням. Маршрут повідомлення по СМО задається матрицею маршрутів, вигляд якої є залежним від того, чи є СМО замкнутою або відкритою. У відкриту СМО повідомлення поступають із зовнішнього джерела і можуть залишати мережу після завершення їх обслуговування. Якщо зовнішнє джерело прийняти за новий мережний центр та позначити його індексом 0, то маршрут у відкритій мережі буде задаватися стохастичною нерозкладною матрицею $P = \|P_{ij}\|$ [21].

Для визначення потоків, що циркулюють у стаціонарному режимі у відкритій СМО, введем коефіцієнти передачі e_i. Тоді вираз $e_i\lambda(N)$ буди визначати загальну інтенсивність потоку повідомлень у i-му центрі мережі ($i=\overline{1,M}$). Видно, що інтенсивність $e_i\lambda(N)$ складається з інтенсивності надходження повідомлень до i-го центру з джерела $P_{0i}\lambda(N)$ та інтенсивностей надходження з інших центрів $e_jP_{ji}\lambda(N)$ ($j=\overline{1,M}$). Тут P_{0i} – імовірність надходження повідомлення в i-й центр із джерела, а P_{ij} – це імовірність того, що повідомлення, яке виходить із i-го центру перейде в j-й центр ($i=\overline{1,M}$; $j=\overline{1,M}$) [19].

У замкнутій СМО повідомлення із зовні не надходять і не залишають мережу, тобто кількість повідомлень, що в ній передаються, є постійною і дорівнює N. Матриця P, що визначає випадкові маршрути передачі повідомлень, як і для відкритої мережі, є також стохастичною та нерозкладною. Поки потенційне повідомлення є нереалізованим у якості вимоги на обслуговування, то вважається, що воно знаходиться в блоці затримки. У момент реалізації вимога поступає безпосередньо у саму систему [19].

Через статистичну однорідність повідомлень, що циркулюють у визначених вище відкритих і закритих СМО, такі системи називають однорідними. При цьому однорідну СМО називають експоненціальною, якщо функції розподілення $H(t)$ та $F_i(t)$ ($i=\overline{1,M}$) будуть експоненціальними [21].

Зазначимо, що узагальненням однорідних СМО є мережі з кількома класами повідомлень, але які будуть відрізнятися як маршрутами, так і тривалістю часу щодо обслуговування їх в центрах. Також СМО може бути змішаною, тобто відкритою для одних класів повідомлень (тих, що поступають до неї ззовні і після закінчення обслуговування залишають її), і замкнутою для інших класів. Кількість повідомлень кожного з таких класів усередині СМО є постійною [19].

В однорідних СМО існує єдиний вхідний потік, у якому всі заявки однотипні: надходять з однією інтенсивністю, мають однаковий середній час обслуговування тощо. Розглянемо систему такого виду [21]:

- до системи надходить M найпростіших потоків повідомлень, що мають інтенсивності $\lambda_1, \ldots, \lambda_M$. Ці потоки утворюють сумарний пуасонівський потік з сумарною інтенсивністю $\Lambda = \sum_{i=1}^{M} \lambda_i$;

- заявки, що входять до i-го потоку, називаються відповідно заявками i-го типу та характеризуються різними первісними моментами часу тривалості обслуговування, тобто – b_i і $b_i^{(2)}$ відповідно;

- завантаження приладу, яке буде створюватися заявками i-го типу визначається як $\rho_i = \lambda_i b_i$. Причому зазначимо, що завантаження, яке створюється сумарним потоком $R = \sum_{i=1}^{M} \rho_i$ має бути більшим за одиницю;

- коефіцієнт простою визначиться як $\eta = 1 - R$;

- середній час очікування (w_{cp}) однієї заявки з сумарного потоку визначається як $w_{cp} = \sum_{i=1}^{M} w_i P_i$. Тут $P_i = \frac{\lambda_i}{\Lambda}$ – це імовірність того, що заявка, яка надійшла, є заявкою i-го типу.

Слід зазначити, що в багатьох реальних системах запити, які надходять у систему, неоднорідні як за їх цінністю для системи, так і за розподілом часу обслуговування, і, відповідно, за правом претендувати на можливість першочергового обслуговування в момент звільнення приладу. Такі моделі досліджуються в рамках теорії пріоритетних СМО. Під пріоритетністю тут мається на увазі те, що в момент закінчення обслуговування запиту, наступним на обслуговування буде обраний запит з черги, який має максимальний пріоритет. Запити, що мають один і той самий пріоритет, обираються відповідно із діючим алгоритмом обслуговування, наприклад, згідно з алгоритмом FIFO.

При безпріоритетному обслуговуванні заявки не будуть мати переваг на здійснення дострокового обслуговування і вибираються з черги за алгоритмами FIFO, LIFO (Last In First Out – останнім прийшов, першим пішов) або за випадковим вибором заявок (Random, RAND) . Ці три безпріоритетні алгоритми мають однаковий середній часом очікування, але дисципліна FIFO може додатково мінімізувати дисперсію часу очікування. Внаслідок цього, коефіцієнт варіації часу очікування для FIFO буде меншим, тобто $v_w^{FIFO} < v_w^{LIFO} < v_w^{RAND}$ [21].

Система називається СМО з відносним пріоритетом, якщо надходження такого запиту не буде переривати обслуговування запиту. У випадку, якщо ж таке переривання відбувається, то система називається СМО з абсолютним пріоритетом. У цьому разі потрібно зробити уточнення щодо подальшої поведінки запиту, обслуговування якого було перерване. Зокрема є такі можливі варіанти поведінки запиту у разі його переривання [21]:

- запит, що був перерваний, виходить із системи і далі втрачається;
- запит, що був перерваний, стає в чергу і продовжує обслуговування з місця, де сталося переривання, після того, як із системи вийдуть усі запити з вищим пріоритетом;
- запит, що був перерваний, стає в чергу і починає обслуговування із самого початку після того, як із системи вийдуть усі запити з вищим пріоритетом.

Тобто, можна бачити, що є досить велика кількість варіантів поведінки СМО з пріоритетом, але загальним в аналізі всіх СМО такого типу є використання поняття періоду зайнятості системи пріоритетними запитами [21].

3.3 Особливості техніки моделювання мереж на основі СМО

Універсальним методом дослідження мереж масового обслуговування є імітаційне моделювання. Відразу потрібно заначити, що такий метод потребує значних витрат часу при здійсненні розробок та використанні

програм моделювання. Основною перевагою імітаційного моделювання порівняно з аналітичним є можливість забезпечити розв'язання більш складних задач. Імітаційні моделі дозволяю враховувати такі фактори, як нелінійні характеристики елементів системи, різні випадкові впливи, та інші фактори, які часто створюють труднощі під час проведення аналітичних досліджень. Результати імітаційного моделювання, можна використовувати для опису поведінки системи, робити оцінку впливів різних параметрів системи на її параметри, виявляти переваги та недоліки змін, що пропонуються, прогнозувати поведінку системи, тощо [20].

В основу імітаційного моделювання покладені наступні основних принципи, зокрема [19, 20]:

- принцип інформаційної достатності, згідно якого, якщо немає жодної інформації про систему, то здійснити моделювання такої системи неможливо;

- принцип здійсненності, згідно якого модель, що розроблюється має досягти реалізації поставленої мети дослідження з відмінною від нуля ймовірністю за певний час;

- принцип множинності моделей, згідно якого модель реалізовуються тільки ті властивості системи, які мають істотний вплив на показник ефективності. Тут для більш вичерпного проведення дослідження модельованого процесу може з’явитися потреба у побудові низки моделей, які б дали змогу з різних боків і з різним ступенем деталізації проводити аналіз параметрів реального процесу;

- принцип агрегування, згідно якого будь-яка складна система може бути представлена набором деяких підсистем, а для можливості їх математичного опису можна застосовувати певні математичні схеми;

- принцип параметризації, згідно якого системи, що часто зустрічаються, та до структури яких включені досить ізольовані підсистеми, що характеризуються деякими характеристиками, можливо замінити у

створюваній моделі відповідними числовими значеннями і не описувати їх функціонування.

Однією із найважливіших особливостей процесу імітаційного моделювання є вибір способу подання імітаційної моделі об'єкта, який досліджується. У разі використання дискретного підходу до створення імітаційних моделей в якості формалізованих подань об'єктів моделювання зазвичай використовують абстрактні системи трьох основних типів: СМО, автоматні та агрегативні системи. Основна відмінність між системами цих трьох типів полягає в рівні їх схожості. Найзагальніший вигляд мають агрегативні системи, а менш загальний – СМО [20].

Таким чином, із вищевикладеного стає зрозумілим, що СМО є сукупністю деякої кількості обслуговуючих вузлів, у яких циркулюють заявки, які передаються відповідно із маршрутною матрицею з одного вузла до іншого. Так, наприклад, якщо канали системи, що здійснює обслуговування, з'єднані паралельно, то буде мати місце багатоканальне обслуговування, а якщо паралельні композиції каналів з'єднані послідовно, то має місце багатофазне обслуговування. На підставі такого подання СМО при здійсненні реалізації процесу моделювання показників ефективності роботи конкретної мережі, на їх основі можна отримати досить ефективний і точний розрахунок характеристик об'єктів [20].

Звідси подальше моделювання доцільно проводити, з використанням наближених методів дослідження IP мереж на основі СМО і теорії черг, тому що вони за рахунок використання більш «гнучких» алгоритмів дають змогу досягти компромісу між суперечливими вимогами моделювання реальних систем і простоти пошуку рішення в мультиплікативній формі [20].

Зазначено, що найпростішим і найбільш універсальним методом дослідження черг є метод імітаційного моделювання, що дає змогу описати логіку функціонування досліджуваних систем і взаємодію окремих її елементів у часі, що свідчить про доцільність використання цього методу, особливо в разі виникнення перевантажень [20].

4 МОДЕЛЮВАННЯ АЛГОРИТМІВ УПРАВЛІННЯ ТА ПЛАНУВАННЯ ЧЕРГ В АСПЕКТІ УПРАВЛІННЯ ТРАФІКОМ У IP МЕРЕЖАХ

4.1 Обґрунтування проведення моделювання та основі пакету GNS3

У другому розділі цієї кваліфікаційної роботи було розглянуто алгоритми планування та управління потоків трафіку в IP мережах. З урахуванням цього, для проведення подальших розробок візьмемо алгоритми управління чергами такі як: найпростіший алгоритм здійснення пасивного управління чергами (TailDrop) та зважений алгоритм довільного раннього виявлення (WRED), що є різновидом алгоритму випадкового раннього виявлення (RED). Також візьмемо для дослідження алгоритм роботи планувальника CBWFQ, що підтримує зважену справедливу буферизацію, та який заснований на класах потоків, що підтримуються.

Мета проведення цього моделювання полягає в здійсненні порівняльного аналізу алгоритмів з використанням імітаційного моделювання на основі програмного забезпечення GNS3. Це програмне забезпечення (ПЗ) надає віртуальне середовище для здійснення розробок та оптимізації мережі будь-якого розміру без необхідності фізичної апаратної реалізації мережної інфраструктури. Також вибір цього ПЗ обумовлений тим, що воно досить точно демонструє роботу реального мережного обладнання [22, 23].

Для проведення дослідження зробимо припущення щодо параметрів моделювання IP-мережі:

- кількість віртуальних машин (Virtual Machine, VM) – до 4-х, оскільки є обмеження в апаратних обчислювальних ресурсах;
- інтенсивність вхідних заявок, що надходять, дорівнює 0,1 с;
- розмір одного пакета приймемо для спрощення 1 Кбайт;
- кількість переданих заявок приймемо 500 заявок;
- обсяг буфера обміну в маршрутизаторі змінюється в межах від 1 до 5 пакетів, тому що більший розмір буфера буде відповідно вимагати більше

часу для проведення експериментальних досліджень з моделлю за рахунок потреби перезапуску VM і зміни розміру буфера вручну.

ПЗ GNS3 – це графічний симулятор мережі, який дає змогу змоделювати віртуальну мережу з маршрутизаторів і віртуальних машин. Працює практично на всіх платформах, але на Linux подібних системах функціонал є значно ширшим .

Залежно від апаратної платформи, на якій буде використовуватися GNS3, можлива побудова комплексних проектів, що складаються з маршрутизаторів Cisco, Cisco ASA, Juniper, а також серверів під управлінням мережевих операційних систем [23].

Середовище містить у собі частину мережного обладнання, що є доступними за замовчуванням (наприклад, комутатор) і частину обладнання, яке необхідно встановлювати окремо, у залежності від структури мережі, що моделюється (наприклад, кінцеві пристрої, маршрутизатори, тощо) [23].

Можливості програми [23]:

- симуляція комутаторів технологій сімейства Ethernet;
- проектування мережних топологій різного ступеню складності;
- за необхідності є можливість з'єднання віртуальної мережі, що змодельована, та реальною мережею;
- побудова великих мереж на основі роутерів (відпрацювання різних протоколів динамічної маршрутизації).

Програмний пакет GNS3 включає до свого складу низьку програм (Dynamips, Dynagen, Qemu/Pemu, Putty, VPCS, WinPCAP і Wireshark). Наприклад, використання WireShark дає змогу провести моніторинг трафіку всередині топології, що була змодельована. Програмний модуль Dynamips є програмним емулятором, який позиціонується в якості серверної компоненти і працює у фоновому режимі. Фактично він є ядром GNS3 , оскільки здійснює емуляцію вказаної операційної системи. Сам по собі Dynamips не має інтерфейсу, тому застосовується модуль Dynagen, який є текстовою консоллю для Dynamips. Крім цих двох, пропонує [23].

Графічне середовище GNS3 поділяється на три області (рисунок 4.1) [22]:

- панель, що зліва, містить мережні пристрої, які можно вибрати для створення топології;
- по середині знаходиться робоча область (на початку вона є порожньою);
- панель, що справа, містить так зване резюме топології (Topology Summary);
- нижня частина – це консоль (Console) Dynagen.

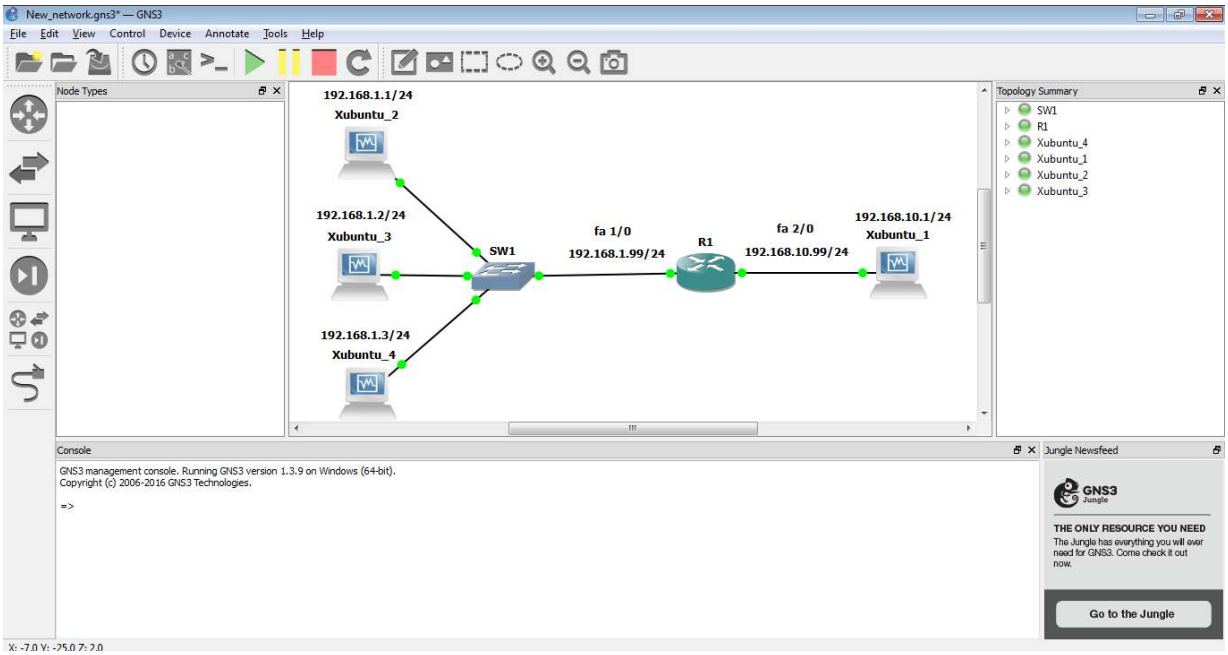


Рисунок 4.1 – Вікно графічного середовища GNS3

На панелі, що зліва, можна вибрати мережні обладнання, наприклад, маршрутизатори, комутатори Ethernet, сервери, хмари тощо. Це залежить від обраної операційної системи і додаткових компонентів, які також можуть бути додані. Після розміщення на панелі пристроїв та клацнувши правою кнопкою миші на мережному пристрої, можна вибрати параметри для його налаштування. Після створення мережної топології та здійснення налаштування вибраного обладнання, можна починати їх тестування, для цього використовується за замовчуванням для Windows або Linux клієнт Telnet [22].

Треба зазначити, що одним із недоліків ПЗ GNS3 є те, що емулятор додатків насправді не має жодного значення з точки зору простого та повного навантаження. Команди виконуються одна за одною протягом моделювання [23].

4.2 Створення моделі мережі

В якості основної (хостової) операційної системи (ОС) використовується Windows 10. Для емуляції кінцевих пристроїв будемо використовувати пакет ПЗ VirtualBox v.6.1.12 (Windows hosts), який завдяки створенню VM дозволяє встановлювати гостьові ОС на основі дистрибутива Linux Xubuntu 14.04 (рисунок 4.2).

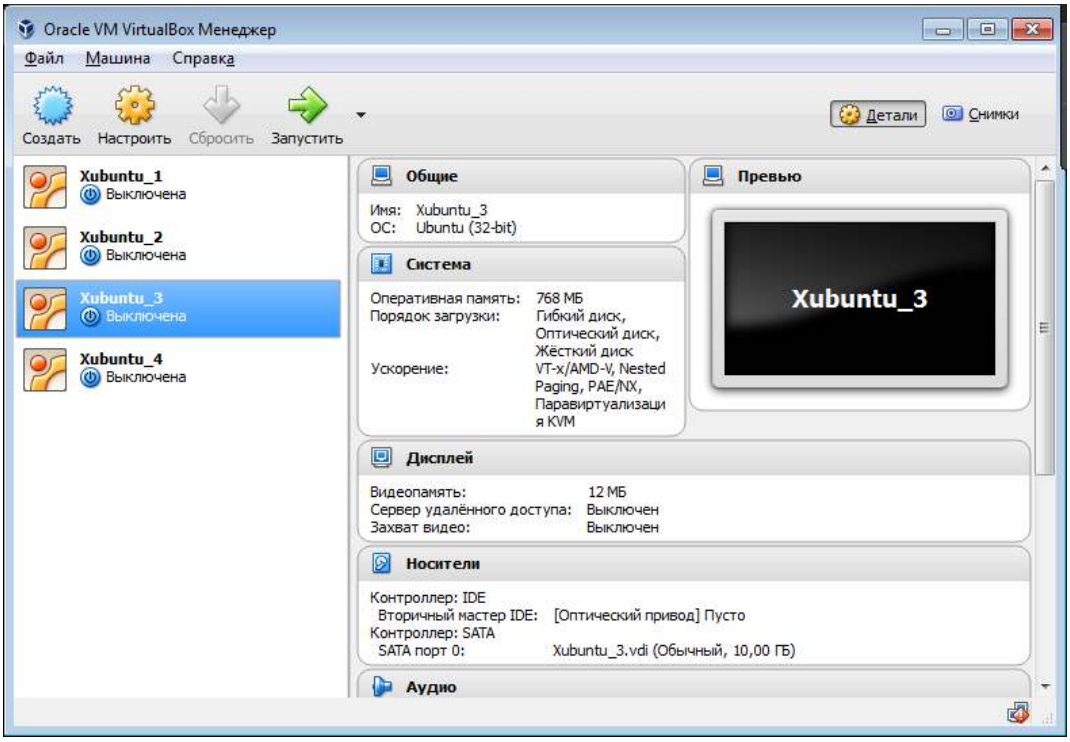


Рисунок 4.2 – Вікно менеджеракно для здійснення управління VM

Інсталяцію програми здійснюємо за допомогою стандартного інсталятора ОС Windows. Таким же чином інсталюємо GNS3 в якості середовища для подальшої організації віртуального середовища. Робимо

потрібні налаштування гостьових ОС для GNS3. У меню «Налаштування» вмикаємо мережний адаптер і встановлюємо тип підключення «Не підключено» для кожної гостьової ОС, як це показано на рисунку 4.3.

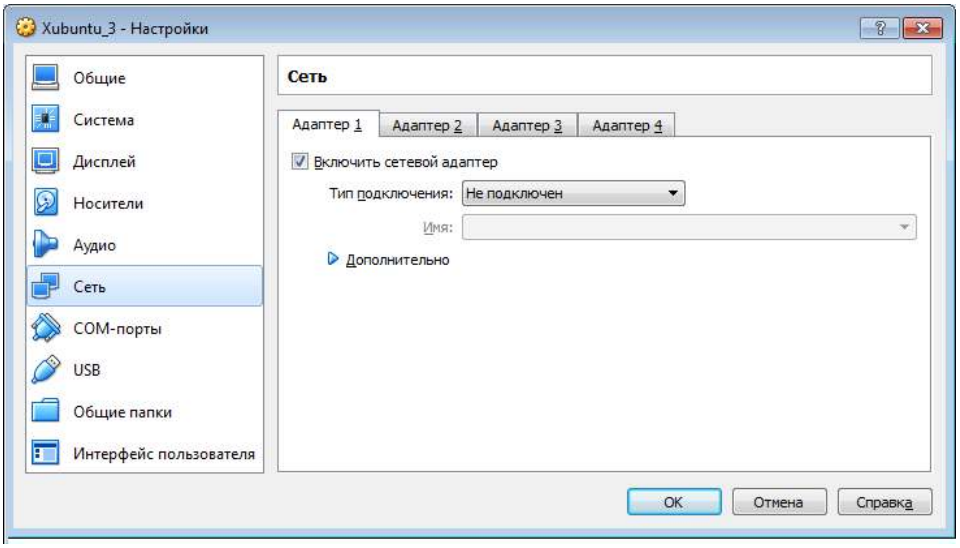


Рисунок 4.3 – Вікно налаштувань VM

Здійснюємо підключення менеджера VBoxManager із пакету ПЗ VirtualBox, як це можна бачити із рисунка 4.4.

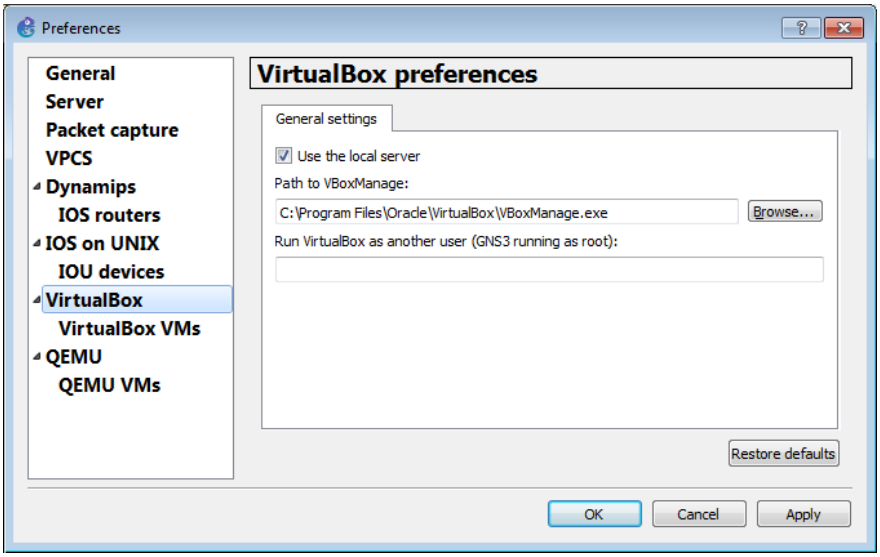


Рисунок 4.4 – Вікно підключення VBoxManager до GNS3

Це дає змогу додати віртуальні машини до програмного середовища GNS3, як це показано на рисунку 4.5.

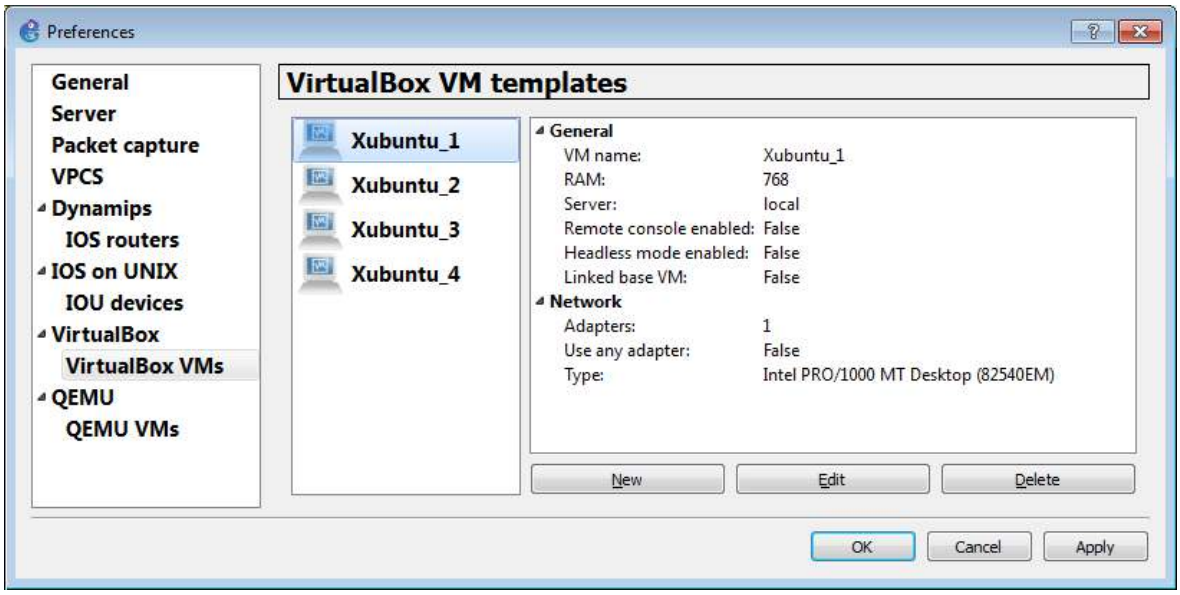


Рисунок 4.5 – Вікно, де здійснюється підключення VM

Далі для кожної віртуальної машини здійснюємо налаштування мережного інтерфейсу «eth0». Для цього потрібно відредагувати файл «/etc/networks/interfaces» (рисунок 4.6).

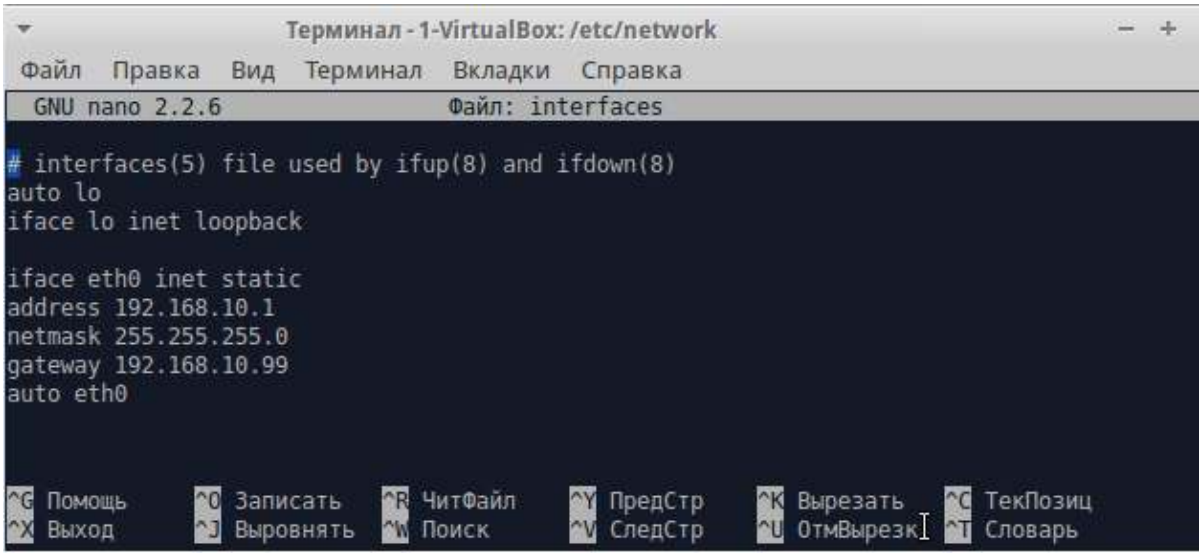


Рисунок 4.6 – Вікно здійснення конфігурації мережі

Робимо перевірку налаштувань інтерфейсу «eth0». Для цього використовуємо команду «ifconfig» (рисунок 4.7).

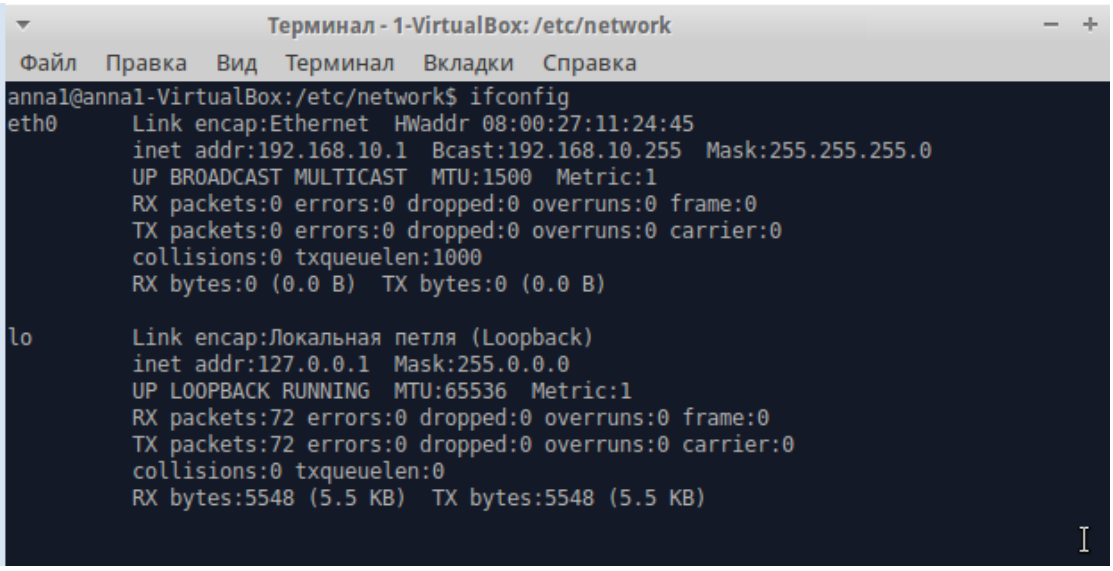


Рисунок 4.7 – Вікно здійснення перевірки налаштувань інтерфейсу

Робимо підключення образу ОС маршрутизатора Cisco C3725, як це показано на рисунку 4.8. Для створення віртуального маршрутизатора використовується ПЗ GNS3.

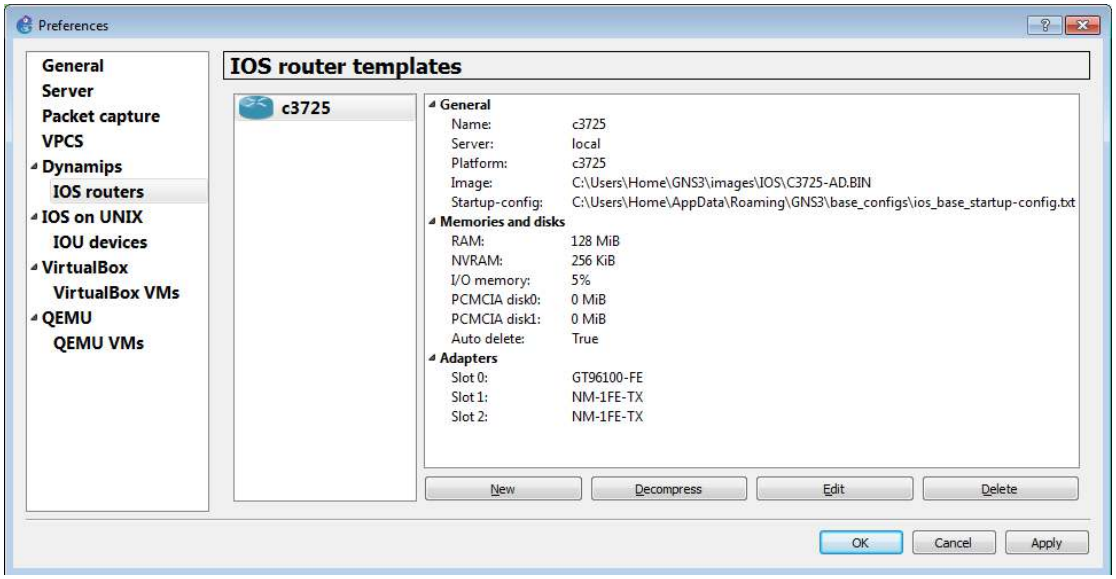


Рисунок 4.8 – Вікно здійснення підключення образу маршрутизатора

Далі проводимо тестування всіх підключень. Для цього використовуємо команду «ping» до кожного із вузлів мережі. В якості прикладу на рисунку 4.9 показаний процес тестування четвертого вузла створеної моделі мережі.

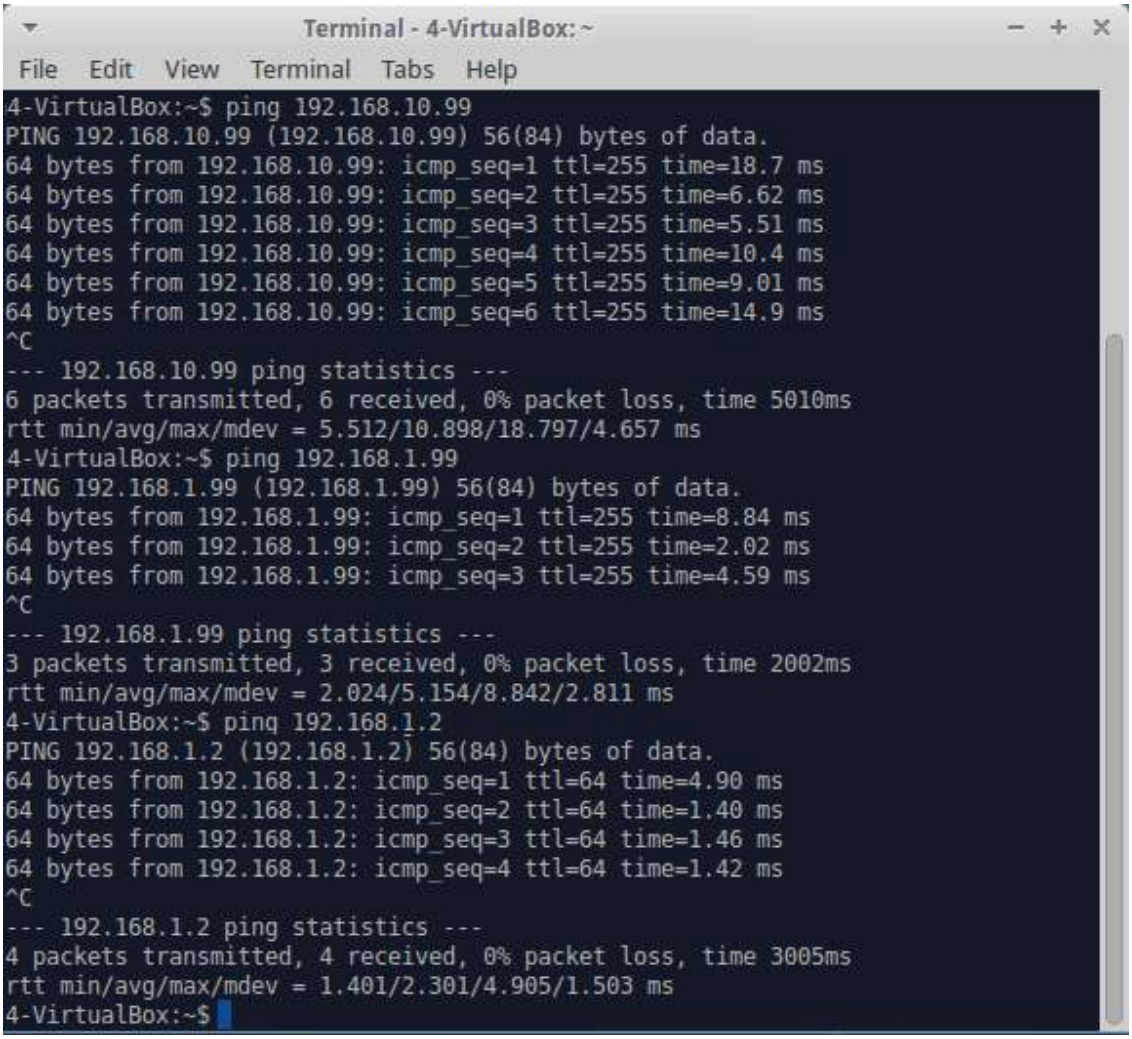


Рисунок 4.9 – Вікно процесу «пінгування» мережі

4.3 Моделювання та дослідження алгоритмів управління трафіком

Як зазначалося для дослідження алгоритмів управління трафіком створюється віртуальна модель IP-мережі, що складається із чотирьох VM, комутатора і маршрутизатора. Структура цієї мережі показана на рисунку 4.10.

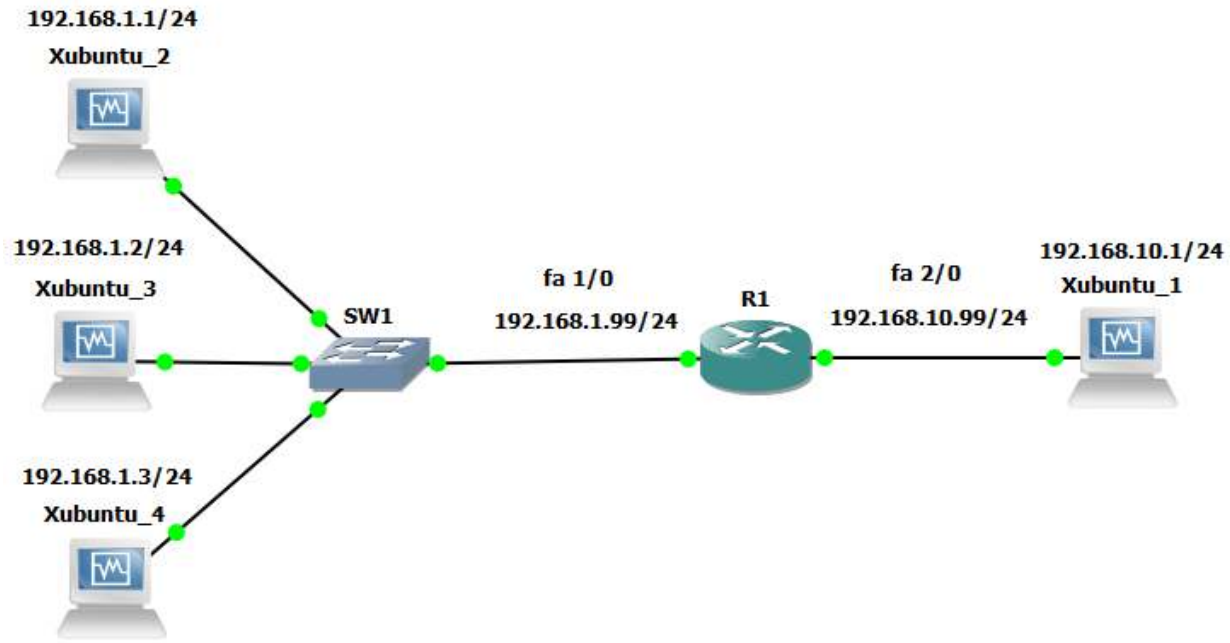


Рисунок 4.10 – Модель IP-мережі, що досліджується у середовищі GNS3

Для початку досліджень потрібно зайти до консолі здійснення налаштувань маршрутизатора і ввести команди, що дозволяють змінити обсяг буфера обміну пакетами. В якості прикладу на рисунку 4.11 показано вікно консолі де задається ліміт обсягу буфера рівним 1 пакету.

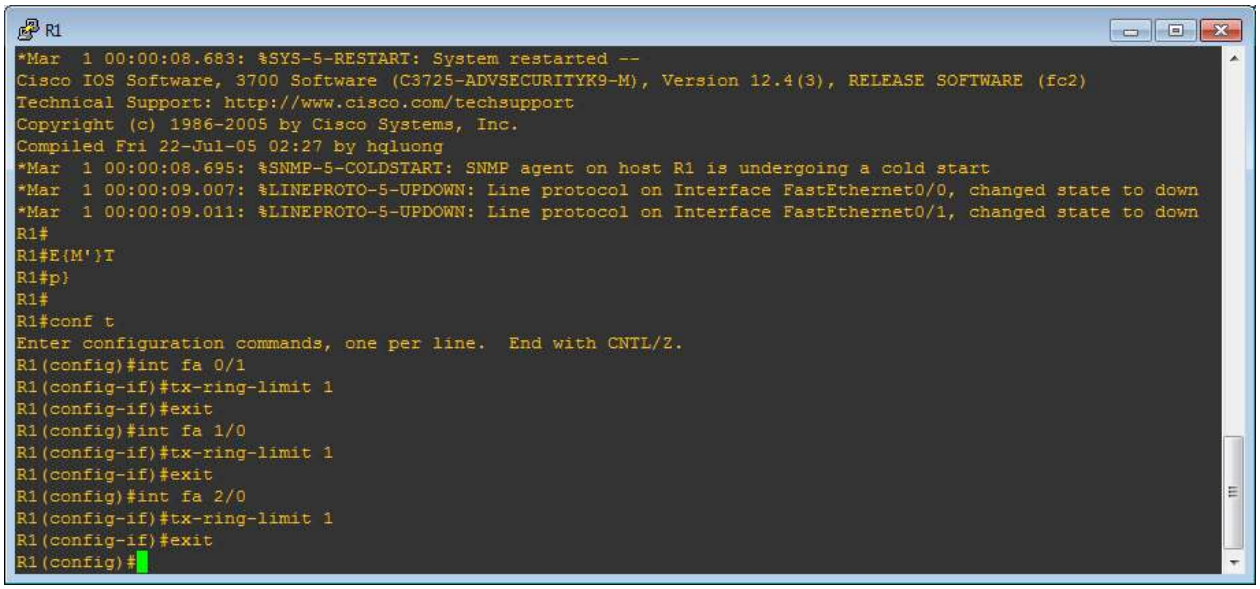


Рисунок 4.11 – Вікно консолі маршрутизатора, де задається обсяг буфера

Із використанням протоколу керуючих повідомлень Internet (Internet Control Message Protocol, ICMP) згенеруємо запити ICMP до вузла мережі Xubuntu_1 з IP-адресою 192.168.10.1. Цей протокол використовується якраз для того, щоб проводити діагностику мережі. Для того, щоб подивитися, як працюють вузли в мережі треба використовувати утиліту «ping», завдяки використанню якої перевіряється доступність вузлів у мережі. У відповідності до припущень, що були зроблені на початку моделювання, будемо мати на увазі, що генерується по 500 заявок із кожної VM з інтенсивністю 0,1 с і розміром в 1 Кбайт (рисунок 4.12).

Time	Source	Destination	Protocol	Length	Info
125 294.379382	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=7/1792, ttl=64 (reply in 132)
132 294.417887	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=7/1792, ttl=63 (request in 125)
139 294.497897	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=8/2048, ttl=64 (reply in 146)
146 294.546403	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=8/2048, ttl=63 (request in 139)
161 294.582908	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=1/256, ttl=64 (reply in 169)
162 294.582908	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=9/2304, ttl=64 (reply in 176)
169 294.636415	192.168.10.1	192.168.1.2	ICMP	1162	Echo (ping) reply id=0x0898, seq=1/256, ttl=63 (request in 161)
176 294.636915	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=9/2304, ttl=63 (request in 162)
183 294.681420	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=2/512, ttl=64 (reply in 197)
190 294.687421	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=10/2560, ttl=64 (reply in 204)
197 294.712424	192.168.10.1	192.168.1.2	ICMP	1162	Echo (ping) reply id=0x0898, seq=2/512, ttl=63 (request in 183)
204 294.722425	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=10/2560, ttl=63 (request in 190)
211 294.808436	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=3/768, ttl=64 (reply in 225)
218 294.810937	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=11/2816, ttl=64 (reply in 232)
225 294.828939	192.168.10.1	192.168.1.2	ICMP	1162	Echo (ping) reply id=0x0898, seq=3/768, ttl=63 (request in 211)
232 294.829439	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=11/2816, ttl=63 (request in 218)
239 294.887446	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=4/1024, ttl=64 (no response found!)
246 294.907949	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=12/3072, ttl=64 (reply in 253)
253 294.922451	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=12/3072, ttl=63 (request in 246)
260 295.003461	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=5/1280, ttl=64 (reply in 274)
267 295.011462	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=13/3328, ttl=64 (reply in 281)
274 295.019463	192.168.10.1	192.168.1.2	ICMP	1162	Echo (ping) reply id=0x0898, seq=5/1280, ttl=63 (request in 260)
281 295.029965	192.168.10.1	192.168.1.3	ICMP	1162	Echo (ping) reply id=0x08f9, seq=13/3328, ttl=63 (request in 267)
288 295.488023	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=6/1536, ttl=64 (reply in 334)
295 295.488523	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=14/3584, ttl=64 (reply in 341)
302 295.489523	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=7/1792, ttl=64 (no response found!)
309 295.490023	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=15/3840, ttl=64 (no response found!)
316 295.490523	192.168.1.2	192.168.10.1	ICMP	1162	Echo (ping) request id=0x0898, seq=8/2048, ttl=64 (no response found!)
323 295.492023	192.168.1.3	192.168.10.1	ICMP	1162	Echo (ping) request id=0x08f9, seq=16/4096, ttl=64 (no response found!)

Рисунок 4.12 – Трафік на інтерфейсі fa 1/0

Перевіряємо статистику пакетів, що отримані отриманих на VM. На рисунок 4.13 в якості прикладу показане статистика загального числа отриманих пакетів на вузлі Xubuntu_1.

Далі зробимо повторення експерименту, змінюючи при цьому обсяг буфера від 2-х до 5-ти. Отримані результати щодо втрачених пакетів у разі використання алгоритму TailDrop зведені до таблиці 4.1.

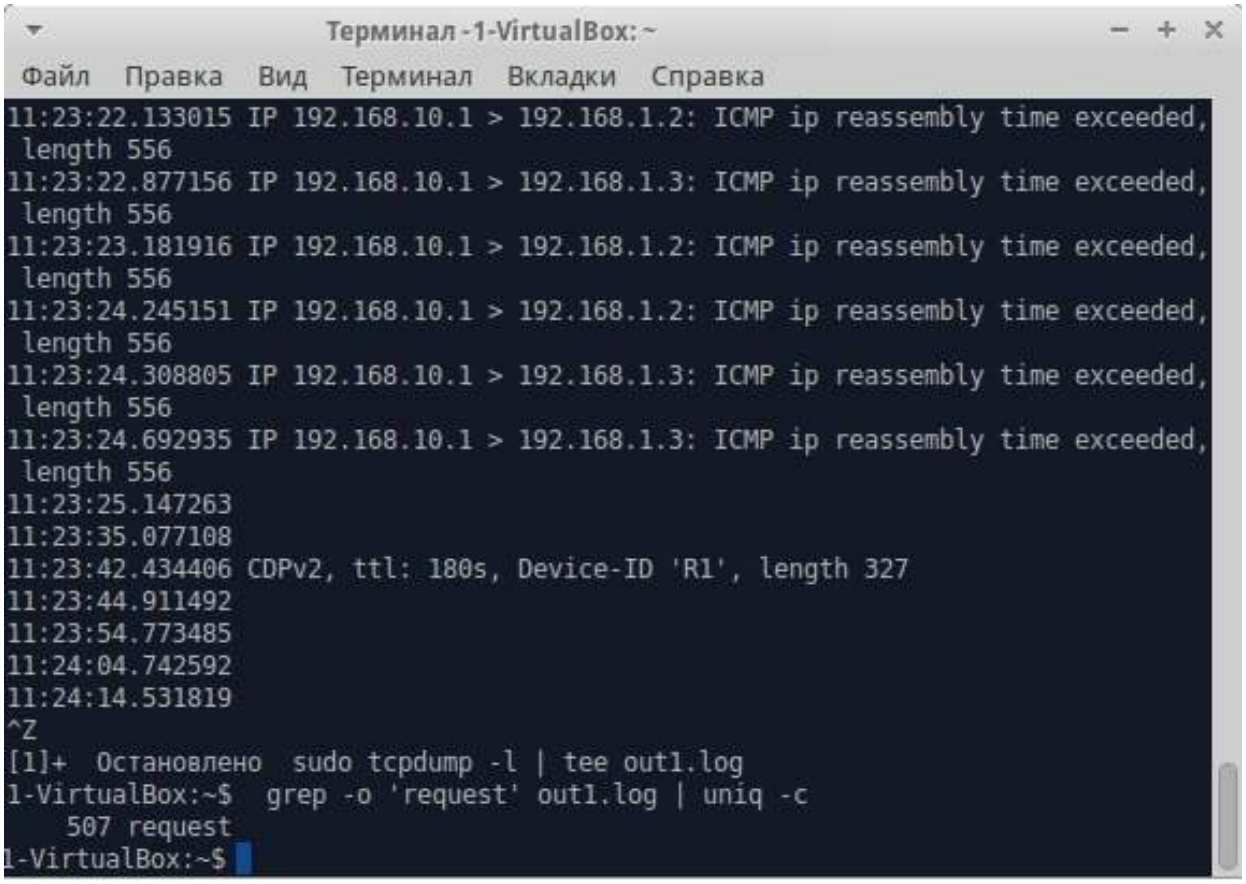


Рисунок 4.13 – Статистика загального числа пакетів, що отримані на вузлі Xubuntu_1

Таблиця 4.1 – Результати здійснення моделювання для алгоритму TailDrop

	Xubuntu2	Xubuntu3	Xubuntu4	Загальна кількість отриманих пакетів
1/1	170	166	171	507
2/2	242	227	244	713
3/3	359	333	311	1003
4/4	500	500	500	1500
5/5	500	500	500	1500

Далі у консолі налаштування маршрутизатора проводимо заміну алгоритму Tail Drop на WRED за допомогою команди «random-detect». І знову повторюємо експеримент, змінюючи при цьому обсяг буфера. Результати експерименту зафіксовані в таблиці 4.2.

Таблиця 4.2 – Результати здійснення моделювання для алгоритму WRED

	Xubuntu2	Xubuntu3	Xubuntu4	Загальна кількість отриманих пакетів
1/1	172	167	175	514
2/2	256	243	249	748
3/3	398	377	399	1174
4/4	500	500	500	1500
5/5	500	500	500	1500

Далі у консолі налаштування маршрутизатора проводимо заміну алгоритму WRED на алгоритм планувальника CBWFQ, при цьому встановлюємо найвищий пріоритет для передачі трафіку на вузлі Xubuntu_4, найменший – на вузлі Xubuntu_3. Повторюємо експеримент, не забуваючи про зміну обсягу буфера. Результати експерименту фіксуємо в таблиці 4.3.

Таблиця 4.3 – Результати здійснення моделювання для алгоритму CBWFQ

	Xubuntu2	Xubuntu3	Xubuntu4	Загальна кількість отриманих пакетів
1/1	167	159	208	534
2/2	227	198	341	766
3/3	368	333	500	1201
4/4	500	500	500	1500
5/5	500	500	500	1500

На основі отриманих результатів, що наведені в таблицях 4.1 - 4.3, побудуємо графіки залежності розміру буфера від втрачених пакетів. Для побудови графічних залежностей будемо використовувати дані, що отримані на VM Xubuntu_4. Ці графічні залежності показані на рисунку 4.14.

Таким чином можна бачити, що алгоритм TailDrop можна використовувати на IP-мережах, де створюються невеликі черги. Для цього алгоритму не має значення який тип трафіку обслуговується, він може функціонувати за будь-яких умов, але одним із основних його недоліків є те, що він може здійснювати відкидання пакетів, які містять важливу інформацію (тобто він не підтримує пріоритизацію трафіку) [4, 13].

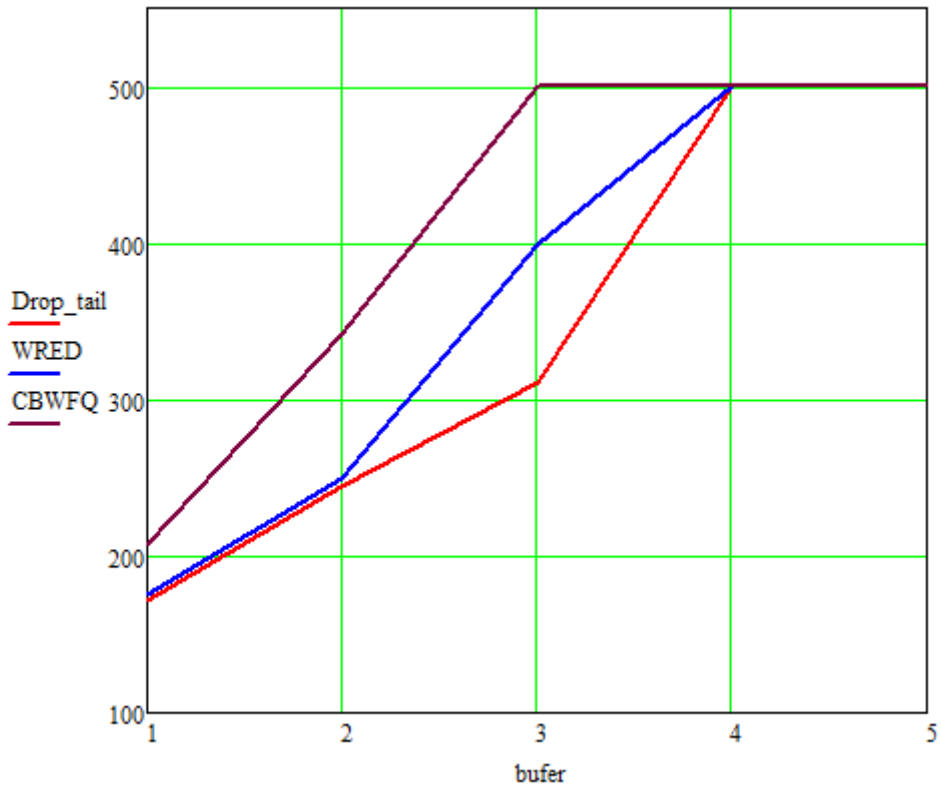


Рисунок 4.14 – Графіки залежності втрачених пакетів від обсягу буфера

Алгоритм WRED динамічно регулює швидкість передачі та відкидає пакети на підставі їх пріоритету, що у свою чергу призводить до можливості не втрачати важливу інформацію. Недоліком алгоритму є те, що він може обробляти тільки TCP трафік [4, 15].

Алгоритм планувальника CBWFQ забезпечує гарантовану передачу пакетів даних для певних користувачів, що є перевагою цього алгоритму. Внаслідок того, що для фільтрації використовуються адреси джерел – він може взаємодіяти з будь-якими типами і класами трафіку, але є суттєвий недолік, який полягає у складності реалізації [18].

ВИСНОВКИ

У цій кваліфікаційній роботі магістра був проведений аналіз алгоритмів управління потоками трафіку для унеможливлення виникнення перевантажень або їх запобіганню у IP-мережах.

В першому розділі кваліфікаційної роботи розглянута загальна концепція організації управління трафіком в IP мережах. Зокрема розглянуті загальні поняття та підходи щодо організації системи управління трафіком у мережах з комутацією пакетів. Показано вся СУ трафіком орієнтована на надання якісних умов роботи всім користувачам і на попередження можливих перевантажень у мережі. Визначені загальні особливості контролю характеристик трафіка для попередження перевантажень. Проаналізовані загальні особливості управління трафіком в мережах з КП, що базуються на алгоритмах планування і управління чергами.

Планування потоків трафіку здійснюється на основі інформації, що зберігається в описі процесів і потоків. У процесі планування трафіку можуть братися до уваги пріоритет потоків трафіку, час їх очікування в черзі та інші чинники. Показано, що основною задачею планувальника є визначення, з якої черги наступний пакет буде відправлений на обслуговування. Тому, саме завдяки планувальнику вирішується задача поділу смуги пропускання вихідного каналу і у відповідності до заданих параметрів QoS [7].

Черги та відповідні алгоритми управління і обробки цих черг становлять основу механізмів, що забезпечують необхідний рівень QoS у будь-якому мережному обладнанні, що працює на за методом КП. Наслідком виникнення черг є погіршення QoS трафіку перш за все за рахунок виникнення перевантажень. Звідси визначена основна задача функціонування алгоритмів управління чергами, яка полягає у запобіганні виникненню або збільшенню ступеня перевантаження за рахунок наявності можливості

скидання пакетів певних класів трафіку у певні моменти часу. Відповідно роботу таких алгоритмів оцінюють в аспекті їх можливостей щодо забезпечення ефективного контролю трафіку в період дії перевантаження. Показано, що такі алгоритми поділяються на пасивні (TailDrop) та активні (RED, WRED) [4].

У другому розділі кваліфікаційної роботи зроблений аналіз найпоширеніших алгоритмів управління трафіком, що застосовуються в IP-мережах, та які входять до групи алгоритмів, що направлена на управління і планування черг. Зокрема для подальших досліджень були проаналізовані найпростіший алгоритм пасивного управління чергами (TailDrop) і один із різновидів алгоритму RED – зважений RED (WRED), а також один із різновидів алгоритмів планувальника WFQ, який базується на класах (CBWFQ).

Аналізу управління трафіком з метою уникнення перевантажень на основі алгоритмів управління чергами та їх планування показав, що в основі їх роботи лежить поняття «дисципліна обслуговування», яка визначає порядок вибору пакетів (заявок) із числа тих, що надійшли, і порядок розподілу їх між вільними каналами (портами, пристроями). Оптимальний вибір такої дисципліни обслуговування, а отже, і конкретного алгоритму управління трафіком багато в чому залежить від реальних мережевих параметрів. Також була звернута увага на те, що оцінка цих параметрів за певних умов роботи IP-мережі та пошук параметрів, які будуть оптимальними за деякими окремими критеріями, базується, як правило, на раціональному виборі структури відповідної СМО.

Тому у третьому розділі кваліфікаційної роботи проведений аналіз механізмів управління трафіком з використанням моделей на основі СМО, що описані в теорії масового обслуговування.

Показано, що СМО являють собою сукупність кінцевої кількості вузлів, що обслуговують, у яких циркулюють заявки, що переходять у відповідності з маршрутною матрицею з одного вузла в інший. Зокрема,

якщо канали системи, що обслуговує, з'єднані паралельно, то має місце багатоканальне обслуговування, а якщо паралельні композиції каналів з'єднані послідовно, то має місце багатофазне обслуговування. На підставі такого відображення СМО під час реалізації моделювання показників ефективності роботи конкретної мережі на їх основі можна отримати досить виважений розрахунок характеристик об'єктів [19].

Зазначено, що при збільшенні розмірності та кількості параметрів СМО, що моделюються, точність обчислення може значно знизитися через, наприклад, обмеженість ресурсів обчислювальної системи. У цьому випадку для аналізу СМО необхідно застосовувати узагальнені алгоритми, що дасть змогу значно розширити сферу застосування точних алгоритмів. Також показано, що в разі, якщо розмірність СМО вимагає витрат пам'яті, що перевищують можливості обчислювальної системи при реалізації узагальнених алгоритмів, то єдиним інструментом аналізу будуть наближені методи [19].

Зазначено, що найпростішим і найбільш універсальним методом дослідження черг є метод імітаційного моделювання, що дає змогу описати логіку функціонування досліджуваних систем і взаємодію окремих її елементів у часі, що свідчить про доцільність використання цього методу, особливо в разі виникнення перевантажень [20].

У четвертому розділі кваліфікаційної роботи були досліджені принципи функціонування алгоритмів управління та планування черг у IP-мережах із застосуванням інструментарію віртуального моделювання пакетної мережі на основі ПЗ GNS3. Із проведених раніше аналізу і досліджень було зроблено висновок, що алгоритм TailDrop можна використовувати на IP-мережах, де створюються невеликі черги. Для нього не має значення тип трафіку, що обслуговується, він функціонує за будь-яких умов. Недоліком є відкидання пакетів, що містять важливу інформацію [4, 13]

Алгоритм WRED динамічно регулює швидкість передачі та відкидає пакети на підставі їх пріоритету, що у свою чергу призводить до можливості

не втрачати важливу інформацію. Недоліком алгоритму є те, що він може обробляти тільки TCP трафік [4, 15].

Алгоритм планувальника CBWFQ забезпечує гарантовану передачу пакетів даних для певних користувачів, що є перевагою цього алгоритму. Внаслідок того, що для фільтрації використовуються адреси джерел – він може взаємодіяти з будь-якими типами і класами трафіку, але є суттєвий недолік, який полягає у складності реалізації [18].

Як показав аналіз показників ефективності роботи алгоритмів TailDrop, WRED і CBWFQ, у IP-мережі у випадку виникнення перевантаження краще використовувати алгоритм CBWFQ, тому що він дає змогу кожному потоку трафіку мати свій пріоритет і відповідну смугу пропускання. У разі наявності перевантаження, це є значною перевагою, тому що забезпечує гарантовану передачу даних для користувачів, що мають найвищий пріоритет.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Росляков, А.В., Ванюшин С.В., Самсоном М.Ю. и др. Сети следующего поколения. – М.: Эко-Трендз, 2008. – 424 с.

2. Олифер В.Г. Компьютерные сети: принципы, технологии, протоколы / В.Г. Олифер, Н.А. Олифер. – 4-е изд. – СПб.: Питер, 2010. – 944 с.

3. Олифер В.Г. Введение в IP-сети [Електронний ресурс] / В.Г. Олифер, Н.А. Олифер. – Доступ здійснено 29.12.2023. – Режим доступу до ресурсу: http://citforum.ru/nets/ip/contents.shtml.

4. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет / Е.А. Кучерявый. − СПб. : Наука и Техника, 2004. – 336 с.

5. Классификация, мониторинг, допуск и управление нагрузкой [Електронний ресурс] // Программирование с использованием OpenGL. – Доступ здійснено 03.01.2024. – Режим доступу до ресурсу: https://opengl.org.ru/upravlenie-trafikom-i-kachestvo-obsluzhevaniya-v-seti/klassifikatsiya-monitoring-dopusk-i-upravlenie-nagruzkoi.html.

6. Бельков Д.В. Система управления трафиком информационной сети [Електронний ресурс] / Д.В. Бельков, Е.Н. Едемская, Т.А. Едемская // Донецкий национальный технический университет. – 2014. – Режим доступу до ресурсу: https://ea.donntu.edu.ua/bitstream/123456789/3844/1/s6_31_belkov.pdf.

7. Классы планировщиков [Електронний ресурс] // Программирование с использованием OpenGL. – Доступ здійснено 07.01.2024. – Режим доступу до ресурсу: https://opengl.org.ru/upravlenie-trafikom-i-kachestvo-obsluzhevaniya-v-seti/klassy-planirovshchikov.html.

8. Планирование потоков в ОС Windows. Величина кванта времени [Електронний ресурс] / Астраханский ГТУ. – 2019. – Режим доступу до ресурсу: https://studfile.net/preview/8594655/page:3/.

9. АЛГОРИТМ DEFICIT ROUND ROBIN [Електронний ресурс] // Программирование с использованием OpenGL. – Доступ здійснено 07.01.2024. – Режим доступу до ресурсу: https://opengl.org.ru/upravlenie-trafikom-i-kachestvo-obsluzhevaniya-v-seti/algoritm-deficit-round-robin.html.

10. Гольдштейн Б.С. IP-телефония / Б.С. Гольдштейн, А.В. Пинчук, А.Л. Суховицкий. – М.: Радио и связь, 2001. – 336 с.

11. Мануйлова Л.В. Исследование динамики процессов передачи данных в сетях TCP/IP и разработка методики оценки эффективности алгоритмов управления очередью маршрутизатора [Електронний ресурс] / Л.В. Мануйлова // Донецкий национальный технический университет. – 2013. – Режим доступу до ресурсу: http://masters.donntu.ru/2013/fkita/manuilova/diss/index.htm#p41.

12. Киреева Н.В. Изучение алгоритма RED в среде NS-2 [Електронний ресурс] / Н.В. Киреева, М.А. Буранова, И.С. Поздняк // МУ к выполнению лабораторной работы по специальностям: 210700, 090106. – Самара: Поволжский ГУ телекоммуникаций и информатики. – 2014. – 34 с. – Режим доступу до ресурсу: http://ib.psuti.ru/content/metod/Изучение%20алгоритма%20RED.pdf.

13. Вербицький Є.Р. Аналіз механізмів комплексного керування трафіком в IP-мережах при виникненні перенавантаження / Є.Р. Вербицький, Ю.М. Колтун // матеріали 11-ої міжнародної науково-технічної конференції «Проблеми інформатизації». Том 3. – Черкаси –Баку – Харків – Бельсько-Бяла. – 16 - 17 листопада, 2023 р. – С. 92.

14. Королькова А.В. К вопросу о классификации алгоритмов RED / А. В. Королькова, Д. С. Кулябов, А. И. Черноиванов // Вестник РУДН. Серия Математика, Информатика, Физика. – №3. – 2009. – С. 34 – 46.

15. Гончаров А.А. Исследование влияния параметров алгоритма WRED на качество обслуживания при передаче данных по перегруженным каналам [Електронний ресурс] / А.А. Гончаров, Ю.А Семенов. –2006. – Режим доступу до ресурсу: http://saturn.itep.ru/wred_2006.htm.

16. Федодеев Д. Алгоритмы управления очередями [Електронний ресурс] / Дмитрий Федодеев // Журнал сетевых решений. – №12. – 2017 – Режим доступу до ресурсу: http://www.osp.ru/lan/2017/12/4659316/.

17. Алгоритм Weighed Fair Queuing [Електронний ресурс] // Программирование с использованием OpenGL. – Доступ здійснено 10.01.2024. – Режим доступу до ресурсу: https://opengl.org.ru/upravlenie-trafikom-i-kachestvo-obsluzhevaniya-v-seti/algoritm-weighed-fair-queuing.html.

18. Сериков А. IP-телефония в компьютерных сетях: учебный Internet-курс: [Електронний ресурс] / Александр Сериков // Национальный Открытый Университет «ИНТУИТ» – 2008. – Режим доступу до ресурсу: https://intuit.ru/studies/professional_retraining/943/courses/8/info.

19. Вишневский В.М. Теоретические основы проектирования компьютерных сетей / В.М. Вишневский – Москва: Техносфера, 2003 – 512 с.

20. Никульский И.Е., Метод оценки задержек распространения при моделировании пакетных сетей / И.Е. Никульский, В.О. Пяттаев // Техника связи. – №6. – 2008. – С. 8 - 10.

21. Ложковский, А.Г. Теория массового обслуживания в телекоммуникациях: учебник / А.Г. Ложковский. – Одесса: ОНАС им. А. С. Попова, 2012. – 112 с.

22. Эмулятор GNS3: Характеристики, описание, преимущества. [Електронний ресурс] – 2013. – Режим доступу до ресурсу: http://www.it-world.pp.ua/gns3/.

23. Новая версия эмулятора GNS3 [Електронний ресурс] – Доступ здійснено 12.01.2024. – Режим доступу до ресурсу: http://habrahabr.ru/post/110805/.

24. The software that empowersnetwork professionals (інсталяційний пакет GNS3 та документація) [Електронний ресурс] / – Доступ здійснено 12.01.2024. – Режим доступу до ресурсу: https://www.gns3.com.