УДК 621.391
DOI https://doi.org/10.32782/2663-5941/2024.1.1/02

Близнюкова А.Д.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Близнюков Д.В.
Науково-дослідний гірничорудний інститут
Криворізького національного університету

Новіков В.І.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

ДОСЛІДЖЕННЯ МЕТОДУ ВИПАДКОВОГО РАННЬОГО ВИЯВЛЕННЯ (RED) ДЛЯ ЗАПОБІГАННЯ ПЕРЕВАНТАЖЕНЬ ТРАФІКУ В МУЛЬТИСЕРВІСНИХ IP-МЕРЕЖАХ

Перевантаження трафіку в мультисервісних IP-мережах виникає внаслідок інтенсивного використання мережевих ресурсів при передачі різних видів даних, таких як голос, відео, текстова інформація тощо. Збільшення обсягу мультимедійного контенту та високочутливість до затримок у реальному часі створює необхідність у вивченні та впровадженні методів, спрямованих на запобігання перевантажень та забезпечення стабільності мережі. У роботі проаналізовано метод випадкового раннього виявлення (RED) для запобігання перевантажень трафіку в мультисервісних IP-мережах. Визначено проблеми пасив нового керування чергами. Розглянута дисципліна черги випадкового раннього виявлення та її особливості застосування. Проведено огляд існуючих варіацій дисципліни черги RED, а саме Gentle RED, Nonlinear RED, Self Configuring RED та Adaptive RED Queue Discipline. Метод випадкового раннього виявлення (RED) вибраний для подальшого дослідження за такими критеріями, як стійкість, адаптивність до змін навантаження та ефективність управління трафіком. В роботі досліджується ефективність використання механізму активного керування чергами RED в умовах перевантаження трафіку в мультисервісній IP-мережі. При виконанні дослідження застосовується моделювання у середовищі NS2 для визначення впливу та ефективності роботи механізму RED на протидію перевантаження трафіку в мультисервісній IP-мережі. Це дозволяє отримати конкретні дані щодо ефективності запропонованого методу в реальних умовах мультисервісної IP-мережі. В результаті дослідження встановлено, що метод випадкового раннього виявлення (RED) виявляється оптимальним рішенням для запобігання перевантажень трафіку в мультисервісних IP-мережах, що відображено в отриманих результатах моделювання в середовищі NS2. Це дозволяє зробити висновок про практичну застосовність та ефективність запропонованого методу у реальних умовах мультисервісних IP-мереж.

Ключові слова: *перевантаження, мультисервісна IP-мережа, метод, черга, моделювання, випадкове раннє виявлення (RED).*

Постановка проблеми. В сучасному світі, де інформаційні технології займають ключову роль у розвитку суспільства, комп'ютерні мережі є невід'ємною частиною інфраструктури, що забезпечує передачу даних на великі відстані. З появою та широким поширенням мультисервісних IP-мереж, основний акцент у сфері телекомунікацій та інтернет-технологій ставиться на забезпеченні якості обслуговування та ефективному керуванні трафіком. Проблема перевантажень трафіку стає надзвичайно актуальною в умовах інтенсивного використання мультисервісних IP-мереж. В останні роки загальний обсяг передачі даних у мережах значно зріс, що призвело до навантаження на їхні ресурси. Ця проблема є особливо гострою у мультисервісних IP-мережах, де одночасно передаються різні типи даних, такі як голос, відео, та дані. Дослідження цієї проблеми має велике значення, оскільки відповідно до сучасних вимог до телекомунікаційних мереж, важливо розробляти та вдосконалювати методи та механізми, що дозволяють уникнути перевантажень та забезпечити ефективне управління трафіком в умовах інтенсивного використання мережевих ресурсів.

Аналіз останніх досліджень і публікацій. Аналіз досліджень і публікацій [1–7] вказує на

значущість та актуальність використання методу RED для забезпечення ефективного управління трафіком у сучасних мультисервісних IP-мережах. Огляд літературних джерел підтверджує, що зростання обсягу мультимедійного контенту та вимог до низьких затримок у реальному часі створює необхідність у вдосконаленні методів управління трафіком. Додатковий аналіз варіацій дисципліни черги RED, таких як Gentle RED, Nonlinear RED, Self Configuring RED та Adaptive RED Queue Discipline [6–9], підкреслює важливість вибору конкретної варіації в залежності від конкретних умов та вимог мережі.

Метою статті є дослідження методу випадкового раннього виявлення для запобігання перевантажень трафіку в мультисервісних IP-мережах.

Виклад основного матеріалу. Використання методу TailDrop для керування TCP трафіком призводить до ефекту, відомого як «глобальна синхронізація». Це відбувається, коли буфер маршрутизатора переповнюється, і всі пакети, що надходять, одночасно відкидаються. У відповідь до цього всі TCP передавачі зменшують розмір свого вікна одночасно, і потім одночасно його збільшують, що призводить до нового періоду високого навантаження. Крім того, ця ситуація призводить до збільшення кількості відкинутих пакетів UDP трафіку, які також проходять через той же буфер маршрутизатора [1]. Для подолання цих проблем був розроблений метод випадкового раннього виявлення перевантажень (RED), який визначає ймовірність скидання пакетів на основі середньої довжини черги. RED залишається найбільш поширеним методом у маршрутизаторах [2], хоча деякі дослідники вказують на його недоліки, оскільки він лише оцінює середню довжину черги, що може призводити до коливань миттєвої довжини черги, і використовує лінійний підхід до зміни ймовірності скидання пакетів, ігноруючи нелінійні аспекти цього процесу.

Методи RED та Adaptive RED

Метод RED [3] визначає середню довжину черги, використовуючи експоненційно згладжений ковзний середній показник α через рекурентну формулу:

$$a \leftarrow (1-w)\cdot a_n + w \cdot q,$$

де α_n – попереднє значення середньої довжини черги, q – поточна величина черги, w – ваговий коефіцієнт, рекомендований для використання зі значенням 0,002.

У випадку, коли черга є пустою, проводиться оцінка середньої величини черги α у відсутність пакетів за формулою:

$$a \leftarrow a_n \cdot (1-w)^m,$$

де m – ймовірна кількість пакетів, що надійшли в чергу. Без цього припущення, оцінка значення α буде некоректною. У випадку, коли пакет надходить в той момент, коли черга є порожньою, значення α розраховується за допомогою такої формули:

$$m = \frac{t - t_0}{s},$$

де t – представляє собою поточний час, а t_0 – це момент часу, коли черга стала порожньою, s – представляє собою середній розмір пакета.

Імовірність маркування або відкидання пакетів систематично змінюється у межах від 0 до max_ρ за наступною математичною залежністю:

$$P_{drop} = \frac{max_p\,(\alpha - min_{th})}{(max_{th} - min_{th})},$$

де min_{th} – вказує на найнижчий рівень середнього значення черги, нижче якого не відбувається відкидання, max_{th} - вказує на найвищий рівень середнього значення черги, після якого усі пакети скидаються.

Фактична ймовірність відкидання пакетів P_α розраховується на підставі лічильника пакетів, які надійшли після останнього відкидання:

$$P_a = \frac{P_{drop}}{(1 - n \cdot P_{drop})},$$

де n – кількість пакетів, які надійшли до черги після останнього відкидання.

Сучасні маршрутизатори використовують модифікацію RED (Random Early Detection) під назвою WRED (Weighted RED) [4]. Ця модифікація дозволяє враховувати різні пріоритети пакетів та підтримувати архітектуру диференційованого обслуговування DiffServ [5]. Досліджено вплив параметрів конфігурації RED, таких як ваговий коефіцієнт (w), максимальна ймовірність скидання (max_p), пороги виникнення осциляцій довжини черги та параметри якості обслуговування. Вибір відповідних параметрів RED є складним завданням, і одним із недоліків методу є висока залежність поведінки алгоритму від встановлених значень цих параметрів. Для розв'язання цієї проблеми було запропоновано механізм Adaptive RED [6, 7], який дозволяє динамічно змінювати значення параметрів, включаючи max_p, залежно від завантаження черги, а також автоматично розраховувати значення max_{th} і вагового коефіцієнта згладжування w за формулою:

$$w = 1 - e^{\frac{1}{C}},$$

де C – це швидкість каналу, що вимірюється в пакетах на секунду.

Для динамічного контролю параметра max_{th} використовують адитивний коефіцієнт α, який обчислюється α = $min(0.01, max_{\rho}/4)$ і мультиплікативний коефіцієнт β, що дорівнює 0.9. Періодично, зазначений інтервал виміру значення середньої довжини черги α порівнюється з певним пороговим рівнем *l*:

$$l = \left[min_{th} + 0.4 \cdot (max_{th} - min_{th}), min_{th} + 0.6 \cdot (max_{th} - min_{th}) \right],$$

при виконанні умови:

$(\alpha > l)$ *and* $(max_{\rho} \le 0.5)$, де α – це середнє значення. Тоді max_{ρ} буде збільшуватися за такою формулою:

$$max_p \leftarrow max_p + a,$$

де α – це адитивний коефіцієнт.

У випадку, якщо

$(\alpha < l)$ *and* $(max_{\rho} \ge 0.01)$,

тоді max_{ρ} буде зменшуватися за такою формулою:

$$max_p \leftarrow max_p \cdot \beta.$$

Отже, характеристики методу RED автоматично змінюються відповідно до обсягу та інтенсивності трафіку.

Типи дисциплін RED Queue

Алгоритм випадкового раннього виявлення (RED) є методом керування чергою у мережевому планувальнику, спрямованим на запобігання перевантаженням.

Існують різні варіації дисципліни черги RED, включаючи наступні [8]:

– Gentle RED;
– Nonlinear RED;
– Self Configuring RED;
– Adaptive RED Queue Discipline.

Gentle RED

Алгоритм випадкового раннього виявлення збільшує шанси на скидання пакетів з 0,05 до 0,50, коли середня довжина черги зростає лінійно від мінімального порогового значення до максимального порогового значення [9]. Проте, коли середня довжина черги незначно перевищує максимальний поріг, ймовірність скидання пакета раптово зростає з 0,50 до 1. Ця зміна не є плавною. Алгоритм gentle RED використовується для плавного регулювання цього раптового переходу. Ця динаміка призводить до інтенсивного відкидання пакетів, тобто, коли середня довжина черги перевищує максимальну, ймовірність відкидання різко зростає до 1. Gentle RED намагається згладити цю криву з кутом, подібним до початкового RED, коли середня довжина черги знаходиться в діапазоні від max_{th} до подвоєного max_{th}.

Nonlinear RED і Self Configuring RED

Замість збільшення P_d лінійно, може бути ефективніше, якщо P_d збільшується повільно, коли він наближається до min_{th}, і різко зростає, коли він наближається до max_{th}. Якщо ймовірність відкидання збільшується лінійно між мінімальним і максимальним порогами, існує значна ймовірність, що вхідний пакет може бути відкинутий, навіть якщо середній розмір черги невеликий [8]. Тому для досягнення високої ймовірності відкидання пакета, коли середня довжина черги наближається до максимального порогу, було розроблено квадратне рівняння для розрахунку ймовірності відкидання, коли середній розмір черги перевищує мінімальний поріг, але залишається меншим за максимальний поріг.

Робота Nonlinear RED:

$$\Rightarrow P_d = 0, \text{ when newavg} <= min_{th}$$

$$\Rightarrow P_d = 1, \text{ when newavg} < max_{th}$$

$$\Rightarrow P_d = (max'_p)2 * [(newavg - min_{th})/(max_{th} - min_{th})]$$

де $max'_p = 1.5 * max_p = 0.75$

Робота Self Configuring RED:

Self Configuring RED адаптує max_p, як показано в коді нижче: При кожному оновленні «newavg»:

```
if (min_th < newavg < max_th)
    status = between;
else if (newavg < min_th && status != below)
    status = below;
    max_p = max_p ÷ ?
else if (newavg > max_th && status != above)
    status = above;
    max_p = max_p x ?
```

Дисципліна Adaptive RED Queue

Мета Adaptive RED схожа на ту, що і у Self-configuring RED. Self-configuring RED старається підтримувати середній розмір черги в межах мінімальних і максимальних порогових значень. Крім того, Adaptive RED вилучає необхідність ручного втручання та автоматично налаштовує параметри. Максимальна ймовірність відкидання адаптується в залежності від доступності мережі, і більше не вимагає ручних налаштувань, як у попередніх версіях RED.

Основні внески в Adaptive RED [8]:

– автоматичне встановлення нижнього порогу (min_{th}) проводиться на підставі пропускної здатності каналу (C) та бажаної затримки в черзі;

– автоматичне визначення верхнього порогу (max_{th}) залежить від встановленого значення min_{th};

– саморегулювання параметру w_q автоматично відповідає пропускній здатності каналу;

– адаптивне налаштування максимальної ймовірності відкидання (max_p) змінюється відповідно до поточної середньої довжини черги.

Перейдемо до дослідження ефективності роботи RED для запобігання перевантажень

трафіку в модельованій мережі та проаналізовано залежність роботи RED від параметрів налаштування мережі.

У роботі досліджується використання механізму RED для запобігання перевантаження трафіку шляхом моделювання в середовищі NS2.

Для вирішення поставлених завдань дослідження використовується середовище NS2. Щоб змоделювати ділянку мережі потрібно створити сценарій моделювання, для написання якого використовується tcl-скрипт [10]. Але спочатку потрібно визначити топологію модельованої ділянки мультисервісної IP-мережі.

Топологія імітаційної моделі мережі має відповідати таким вимогам:

– необхідність забезпечити, щоб протоколи TCP були наявні для відправників і одержувачів трафіку;

– пропускна здатність каналів зв'язку має бути використана на максимум;

– потрібно створити проблемну ділянку, для того щоб штучного створювати перевантаження;

– можливість зміни дисциплін обслуговування черги має бути наявною для проміжних маршрутизаторів;

– одиничність наявного маршруту, який здійснюється від відправників до одержувачів.

Було створено топологію мережі, яка має відповідати наведеним вище вимогам, вона наведена на рисунку 1.

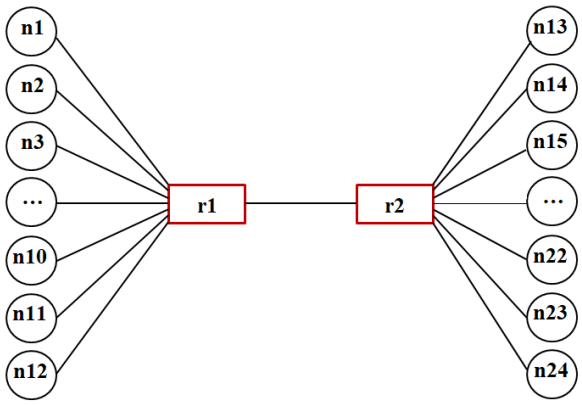

Рис. 1. Створена топологія мережі

На рисунку 1 можна побачити, що n1 – n12 – представляють собою відправників даних, а n13 – n24 – представляють собою одержувачів даних. Проміжними маршрутизаторами на рисунку зображені r1 і r2. Лінії зв'язку між відправниками та маршрутизатором мають пропускну здатність 5 Гб/с, в якості як фізичного, так і канального рівнів стека протоколів.

Для передачі повідомлень одержувачу використовується транспортний протокол TCP із реалізаціями Reno та Vegas, над яким функціонує протокол прикладного рівня FTP, що спрямований на використання наявної пропускної здатності максимально можливою мірою.

Маршрутизатор виконує функцію маршрутизації для отримуваних пакетів та передає їх по лінії зв'язку до одержувача. Лінія зв'язку, в свою чергу, має обмежену пропускну здатність в 0.5 Гб/с. Ця пропускна здатність значно нижча за загальну пропускну здатність каналів, які з'єднують маршрутизатор із відправниками. Змінюючи значення смуги пропускання, моделюється різна пропускна здатність маршрутизатора, а також різна частота відкидання пакетів.

Важливо зазначити, що під час моделювання в середовищі NS2 питання, які стосуються фізичної реалізації передачі даних по лінії зв'язку, можуть враховуватись лише обмежено, тобто можна встановити лише пропускну здатність та затримку. Це забезпечує незалежність функціонування протоколів, що мають більш високий рівень. Саме через це детальний аналіз організації цього рівня не доцільно розглядати в даній роботі. Маршрутизатор використовує реалізовані в середовищі NS2 механізми обслуговування черги, такі як алгоритми RED.

Після встановлення топології мультисервісної мережі, необхідно розробити відповідний сценарій. Зважаючи на унікальні характеристики побудованої топології та поставлені завдання для дослідження, використовуються такі ключові параметри для сценарію, як:

– пропускна здатність ліній зв'язку між маршрутизатором та відправниками/одержувачами даних складає 5 Гб/с;

– затримка в лініях зв'язку між маршрутизатора та відправниками/одержувачами дорівнює 3 мілісекундам;

– максимальна довжина черги маршрутизатора буде змінюватися від 50 до 500 сегментів;

– протокол передачі даних TCP використовує реалізацію «Reno» або «Vegas»;

– розмір сегменту даних становить 1500 байт;

– для обслуговування черги маршрутизатора використовується алгоритм «RED» (зі стандартними параметрами, які визначені в скрипті «ns-default.tcl»);

– загальний час моделювання складає 240 секунд;

– початок передачі даних відправниками розпочинається від 0 до 220 секунд для першого та дванадцятого відправників, відповідно;

– основним навантаженням у мережі є потоки даних за допомогою протоколу FTP, який передає файли нескінченної довжини.

З самого початку створюється файл для сценарію моделювання «simulation.tcl». Відповідно до особливостей роботи з середовищем NS2, а саме створення сценаріїв моделювання, вміст файлу «simulation.tcl» сформований на мові Tcl.

Далі створюється новий об'єкт моделювання, використовуючи команду «set ns [new Simulator]». Також визначається процедура «finish». Вона виконує дії, що необхідні при завершенні моделювання, включаючи закриття файлів з даними, очищення буфера трасування та виклик програми «xgraph», яка буде використовуватись саме для побудови графіків отриманих значень. Текст процедури «finish» наведений на рисунку 2.

```
# Define 'finish' procedure (include post-simulation processes)
proc finish {} {
    global tchan_
    set awkCode {
        {
            if ($1 == "Q" && NF>2) {
                print $2, $3 >> "temp.q";
                set end $2
            }
            else if ($1 == "a" && NF>2)
            print $2, $3 >> "temp.a";
        }
    }
    set f [open temp.queue w]
    puts $f "TitleText: red"
    puts $f "Device: Postscript"

    if { [info exists tchan_] } {
        close $tchan_
    }
    exec rm -f temp.q temp.a
    exec touch temp.a temp.q

    exec awk $awkCode all.q

    puts $f \"queue
    exec cat temp.q >@ $f
    puts $f \n\"ave_queue
    exec cat temp.a >@ $f
    close $f
    exec /home/nastya/Desktop/ns2/xgraph_4.38_linux64/XGraph4.38_linux64/bin/xgraph -bb -tk
 -x time -y queue temp.queue &
    exit 0
}
```

Рис. 2. Текст кінцевої процедури «finish»

Розглянемо опис топології мережі, що задається для дослідження. У запропонованій моделі мережі запрограмовані 24 вузли, з яких n1 – n12 це відправники даних, n13 – n24 це одержувачі даних. Також створюється ще два вузли r1 та r2, вони є маршрутизаторами. Створення та опис ліній зв'язку між вузлами наведено на рисунку 3 та 4 відповідно.

```
set node_(s1) [$ns node]
set node_(s2) [$ns node]
set node_(r1) [$ns node]
set node_(r2) [$ns node]
```

Рис. 3. Створення вузлів топології

```
$ns duplex-link $node_(s1) $node_(r1) 5Gb 3ms DropTail
$ns duplex-link $node_(s2) $node_(r1) 5Gb 3ms DropTail
```

Рис. 4. Опис ліній зв'язку

На рис. 5 показано конфігурацію максимальної довжини черги маршрутизатора.

```
$ns duplex-link $node_(r1) $node_(r2) 0.5Gb 6ms RED

#Set Queue Size of link (r1-r2) to 500
$ns queue-limit $node_(r1) $node_(r2) 500
$ns queue-limit $node_(r2) $node_(r1) 500
```

Рис. 5. Завдання максимальної довжини черги маршрутизатора

Необхідне створення та налаштування 12 з'єднань протоколом TCP. Для досягнення цієї мети визначаються агенти та адресати трафіку TCP, які зв'язуються з необхідними для них вузлами. Встановлюється з'єднання між агентами та адресатами, та задається максимальний розмір для плаваючого вікна протоколу TCP. Відповідні конфігурації зображені на рисунку 6.

```
set tcp1 [$ns create-connection TCP/Vegas $node_(s1) TCPSink $node_(s13) 0]
$tcp1 set window_ 1500
set tcp2 [$ns create-connection TCP/Reno $node_(s2) TCPSink $node_(s14) 1]
$tcp2 set window_ 1500
```

Рис. 6. З'єднання TCP

Для генерації надісланих даних через протокол транспортного рівня TCP поверх нього використовується протокол прикладного рівня FTP. Процедура створення агентів FTP та їх зв'язок з наявними з'єднаннями TCP наведена на рисунку 7.

```
set ftp1 [$tcp1 attach-source FTP]
set ftp2 [$tcp2 attach-source FTP]
```

Рис. 7. Створення агентів FTP

Конфігурація з'єднань між двома маршрутизаторами, що трасують чергу наведена на рисунку 8.

```
# Tracing a queue
set redq [[$ns link $node_(r1) $node_(r2)] queue]
set tchan_ [open all.q w]
$redq trace curq_
$redq trace ave_
$redq attach $tchan_
```

Рис. 8. Створення з'єднання між маршрутизаторами r1 та r2

Розглянемо розроблення розкладу моделювання, який автоматизує запуск необхідних процедур у визначений момент часу. Відповідні операції наведені на рисунку 9.

```
#Schedule events for the FTP agents
$ns at 0 "$ftp1 start"
$ns at 20 "$ftp2 start"
$ns at 40 "$ftp3 start"
$ns at 60 "$ftp4 start"
$ns at 80 "$ftp5 start"
$ns at 100 "$ftp6 start"
$ns at 120 "$ftp7 start"
$ns at 140 "$ftp8 start"
$ns at 160 "$ftp9 start"
$ns at 180 "$ftp10 start"
$ns at 200 "$ftp11 start"
$ns at 220 "$ftp12 start"
$ns at 240 "finish"
```

Рис. 9. Розклад моделювання

Процедура безпосереднього запуску моделювання виконується за допомогою команди «$ ns run».

Створений та описаний попередньо сценарій запускається для моделювання в середовищі NS2, використовуючи командний рядок. Запуск виконується за допомогою команди «ns simulation.tcl». При запуску середовище NS2 запускає інтерпретатор мови Tcl, а він, в свою чергу, виконує по черзі кожен рядок скрипту, а також перевіряє скрипт на коректність та наявність синтаксичних помилок.

Для детального дослідження роботи механізму RED було п'ять разів модифіковано вихідний код створеного сценарію. А саме змінювались значення розміру черги, кількість джерел протоколу TCP, розмір смуги пропускання ліній зв'язку маршрутизатора з одержувачем, пропускна здатність ділянки, що є проблемною та час моделювання.

У наступних графіках показані результати моделювання для використовуваного механізму при різних варіантах роботи алгоритму RED. Головним параметром виконаного моделювання є розмір черги маршрутизаторів «queue».

Результати моделювання при першій модифікації параметрів наведені на рисунку 10. Розмір черги становить 500 сегментів, використовуються шість джерел протоколу TCP, три з яких використовують реалізацію TCP Vegas, а інші три – TCP Reno. Розмір смуги пропускання лінії зв'язку маршрутизатора з одержувачем становить 5 Гб/с, а RED є механізмом обслуговування черги маршрутизатора. Пропускна здатність проблемної ділянки становить 0.5 Гб/с.

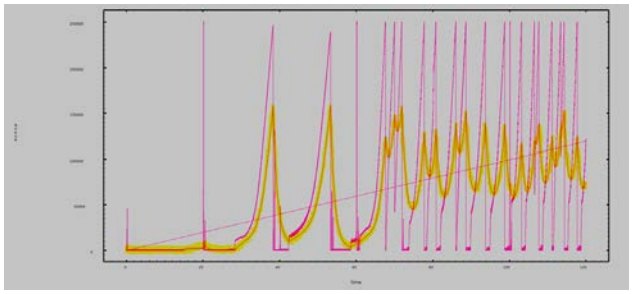

Рис. 10. Залежність довжини черги від часу при першій модифікації параметрів

Результати моделювання при другій модифікації параметрів наведені на рисунку 11. Розмір черги становить 250 сегментів, використовуються шість джерел протоколу TCP, три з яких використовують реалізацію TCP Vegas, а інші три – TCP Reno. Розмір смуги пропускання лінії зв'язку маршрутизатора з одержувачем становить 5 Гб/с, а RED є механізмом обслуговування черги маршрутизатора. Пропускна здатність проблемної ділянки становить 0.5 Гб/с.

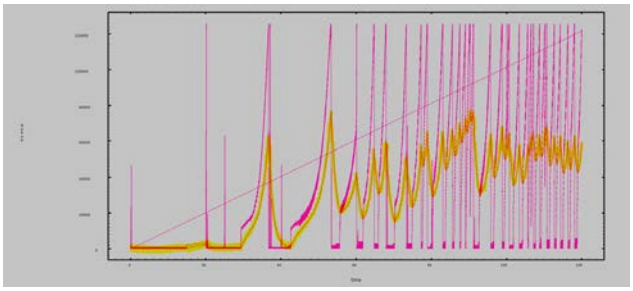

Рис. 11. Залежність довжини черги від часу при другій модифікації параметрів

Результати моделювання при третій модифікації параметрів наведені на рисунку 12. Розмір черги становить 100 сегментів, використовуються шість джерел протоколу TCP, три з яких використовують реалізацію TCP Vegas, а інші три – TCP Reno. Розмір смуги пропускання лінії зв'язку маршрутизатора з одержувачем становить 0.5 Гб/с, а RED є механізмом обслуговування черги маршрутизатора. Пропускна здатність проблемної ділянки становить 0.1 Гб/с.

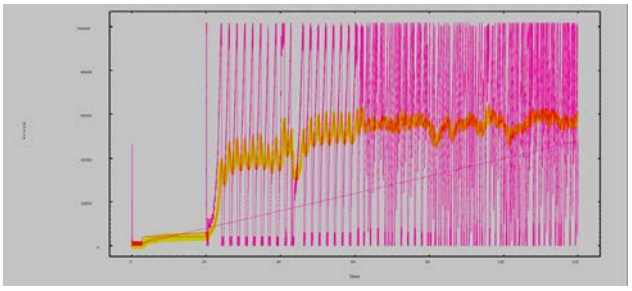

Рис. 12. Залежність довжини черги від часу при третій модифікації параметрів

Результати моделювання при четвертій модифікації параметрів наведені на рисунку 13. Розмір черги становить 50 сегментів, використовуються шість джерел протоколу TCP, три з яких використовують реалізацію TCP Vegas, а інші три – TCP Reno. Розмір смуги пропускання лінії зв'язку маршрутизатора з одержувачем становить 0.5 Гб/с, а RED є механізмом обслуговування черги маршрутизатора. Пропускна здатність проблемної ділянки становить 0.1 Гб/с.

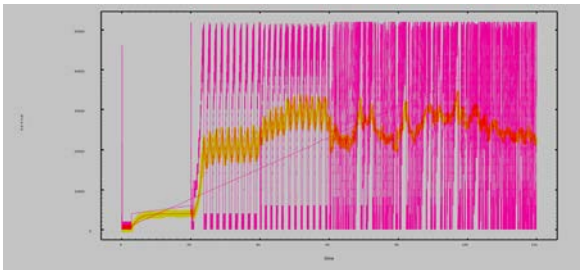

Рис. 13. Залежність довжини черги від часу при четвертій модифікації параметрів

Результати моделювання при п'ятій модифікації параметрів наведені на рисунку 14. Розмір черги становить 200 сегментів, використовуються дванадцять джерел протоколу TCP, шість з яких використовують реалізацію TCP Vegas, а інші шість – TCP Reno. Розмір смуги пропускання лінії зв'язку маршрутизатора з одержувачем становить 0.5 Гб/с із затримкою в 3мс, а RED є механізмом обслуговування черги маршрутизатора. Пропускна здатність вузького місця становить 0.1 Гб/с та має затримку в 6мс.

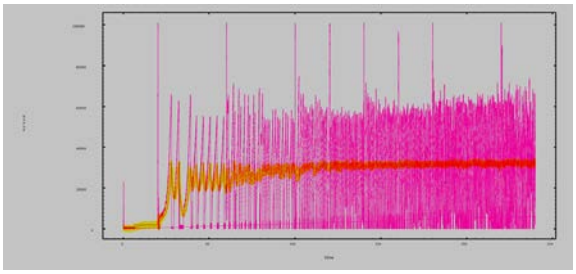

Рис. 14. Залежність довжини черги від часу при п'ятій модифікації параметрів

У випадку з розміром черги, який становив 500 сегментів та реалізацією протоколів TCP Vegas і TCP Reno, було помічено, що сплески в черзі спостерігалися при використанні протоколу TCP Vegas, і черга швидко заповнювалася. Однак, середнє значення заповнення черги залишалося на прийнятному рівні, коливаючись між 70 і 140 сегментами, що свідчить про те, що механізм RED ефективно контролює розмір черги та знижує ймовірність втрат даних. Зменшення розміру черги до 250 сегментів призвело до збільшення частоти сплесків, що може свідчити про більшу чутливість черги до змін обсягу даних. Однак, середнє значення заповнення черги залишалося на прийнятному рівні, коливаючись між 30 і 72 сегментами, показуючи, що механізм RED впорався зі змінами в умовах мережі. З розміром черги, становлячи 50 сегментів, відбувалася значна втрата даних через велику кількість випадкових відкидань пакетів. Черга в цьому випадку не була здатна витримати такий великий поток даних, і механізм RED не можна було б вважати ефективним при такому розмірі черги. У випадку з чергою розміром 200 сегментів і зміною певних параметрів мережі, черга заповнювалася лише наполовину завдяки ефективності механізму RED у врахуванні часу затримки. Цей результат показує, що правильний підбір параметрів черги і розумне використання механізму RED можуть сприяти збалансуванню навантаження в мережі та зменшенню втрат даних. У п'ятому сценарії з чергою розміром 200 сегментів і використанням дванадцяти джерел протоколу TCP Vegas і TCP Reno, відзначалася значна резервна місткість черги завдяки ефективному управлінню чергою механізмом RED. Це може свідчити про те, що правильне використання механізму RED може допомогти збалансувати навантаження та забезпечити оптимальне використання доступної смуги пропускання.

На підставі проведеного дослідження можна рекомендувати використання методу RED у таких випадках:

– Рекомендації щодо розміру черги: RED може бути ефективним при розмірах черги, які можуть варіюватися від 200 до 500 сегментів, забезпечуючи оптимальний баланс між заповненістю черги та втратами даних.

– Чутливість до змін в мережі: RED може успішно адаптуватися до змін обсягу даних та параметрів мережі, таких як час затримки.

– Використання з різними протоколами: RED показав ефективність при використанні різних реалізацій протоколів TCP (Vegas і Reno) та змінних числах джерел даних.

Важливо також відзначити, що оптимальні параметри RED можуть варіюватися в залежності від конкретних умов мережі, тому рекомендується проводити додаткові експерименти та налаштування параметрів RED для конкретних сценаріїв та мережевих умов.

Для подальшого вдосконалення та розширення дослідження вирішення завдання перевантаження трафіку можна розглянути наступні ідеї та аспекти.

Оптимізація параметрів RED:

– Використання оптимізаційних методів або автоматизованих алгоритмів для знаходження оптимальних значень параметрів.

Дослідження впливу типів трафіку та протоколів:

– Розширення дослідження на різні типи трафіку та протоколи для оцінки ефективності RED в різних сценаріях.

Адаптація до змінних умов мережі:

– Врахування динамічних змін у мережі та розробка механізмів адаптації RED до змін обсягу даних та трафіку.

Розгляд сумісності з іншими AQM методами:

– Дослідження можливостей комбінації RED з іншими методами AQM для покращення загальної ефективності в різноманітних умовах.

Використання машинного навчання та аналізу даних:

– Застосування методів машинного навчання для прогнозування та автоматичного налаштування параметрів RED з урахуванням динаміки мережі.

Врахування цих аспектів може допомогти у подальших дослідженнях та поліпшити ефективність методу RED для запобігання перевантаження трафіку в мультисервісних IP-мережах.

Висновки. На підставі аналізу результатів дослідження можна рекомендувати використання методу Random Early Detection (RED) у таких випадках: RED може бути ефективним при розмірах

черги, від 200 до 500 сегментів, забезпечуючи оптимальний баланс між заповненістю черги та втратами даних; RED успішно адаптується до змін обсягу даних та параметрів мережі, таких як час затримки; RED демонструє ефективність при використанні різних реалізацій протоколів TCP (Vegas і Reno) та змінних числах джерел даних. З великим розміром черги та ефективним виявленням перевантажень RED допомагає уникнути деградації якості обслуговування. RED виявляється ефективним в умовах змінного обсягу трафіку, де зміни в розмірі трафіку можуть виникати швидко. Важливо зазначити, що з розміром черги 50 сегментів, відбувалася значна втрата даних через велику кількість випадкових відкидань пакетів. Черга в цьому випадку не була здатна витримати такий великий потік даних, і механізм RED не можна було б вважати ефективним при такому розмірі черги. Це відбувається через випадкове відкидання великої кількості пакетів даних, і робить майже неможливим передачу файлів, які мають великий розмір. Також через це практично неможливе проведення відеозв'язку, оскільки присутні значні затримки та великі втрати пакетів.

Список літератури:

1. Sawashima H., Hori Y., Sunahara H. Characteristics of UDP packet loss: effect of TCP traffic. *Internet Society's seventh annual conference, INET'97*. 1997. P. 6.
2. May M., Bolot J., Diot C., Lyles B. Reasons not to deploy RED. *Seventh International Workshop on Quality of service (IWQoS '99)*. 1999. P. 40– 46. DOI: 10.1109/IWQOS.1999.766502.
3. Floyd S., Jacobson V. Random Early Detection gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*. 1993. No. 1(4), P. 397–413. DOI: 10.1109/90.251892.
4. Class-Based Weighted Fair Queueing. *Cisco.com*. URL: https://www.cisco.com/en/US/docs/ios/12_0t/12_0t5/feature/guide/cbwfq.html (дата звернення 18.10.2023).
5. Blake S., Black D., Carlson M., Davies E., Wang Z., Weiss W. An Architecture for Differentiated Services. *RFC–2475*. 1998
6. Floyd S., Gummadi R., Shenker S. Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management. *Rep. ICSI; Executor*. 2001. P.518-522.
7. Feng W.-C., Kandlur D., Saha D., Shin K. A Self-Configuring RED Gateway. *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now.* 1999. No. 3(99CH36320), P. 1320–1328. DOI: 10.1109/INFCOM.1999.752150.
8. Types of RED Queue Disciplines. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/types-of-red-queue-disciplines/ (дата звернення 06.11.2023).
9. Random Early Detection (RED) Queue Discipline. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/random-early-detection-red-queue-discipline/ (дата звернення 01.11.2023).
10. Harju K., Korventausta, S. Network Simulation and Protocol Implementation Using Network Simulator 2. *TTKK, Tech. Rep.* 2001. URL: https://silo.tips/download/network-simulation-and-protocol-implementation-using-network-simulator-2 (дата звернення 07.10.2023).

Blyzniukova A.D., Blyzniukov D.V., Novikov V.I. STUDY OF RANDOM EARLY DETECTION (RED) METHOD FOR PREVENTING TRAFFIC OVERLOAD IN MULTISERVICE IP NETWORKS

Traffic congestion in multi-service IP networks occurs as a result of intensive use of network resources when transmitting various types of data, such as voice, video, text information, etc. The increase in the volume of multimedia content and high sensitivity to delays in real time creates the need to study and implement methods aimed at preventing overloads and ensuring network stability. The paper analyzes the random early detection (RED) method for preventing traffic congestion in multiservice IP networks. The problems of passive new queue management have been identified. The discipline of the random early detection queue and its application features are considered. The existing variations of RED queue discipline, namely Gentle RED, Nonlinear RED, Self-Configuring RED and Adaptive RED Queue Discipline, are reviewed. The Random Early Detection (RED) method is selected for further investigation based on criteria such as robustness, adaptability to load changes, and traffic management efficiency. The work investigates the effectiveness of using the active RED queue management mechanism in conditions of traffic congestion in a multi-service IP network. When performing the research, modeling in the NS2 environment is used to determine the impact and effectiveness of the RED mechanism on countering traffic congestion in a multi-service IP network. This makes it possible to obtain specific data on the effectiveness of the proposed method in the real conditions of a multi-service IP network. As a result of the study, it was found that the random early detection (RED) method is the optimal solution for preventing traffic congestion in multi-service IP networks, which is reflected in the obtained simulation results in the NS2 environment. This allows us to draw a conclusion about the practical applicability and effectiveness of the proposed method in the real conditions of multiservice IP-networks.

Key words: *congestion, multiservice IP network, method, queuing, simulation, random early detection (RED).*