УДК 004.715

К.т.н., доцент Орлова М.М., магістрант Ковшун Д.В.

Національний технічний університет України
«Київський політехнічний інститут»

ОПТИМІЗАЦІЯ МАРШРУТИЗАЦІЇ У РОУТЕРАХ НА БАЗІ ЯДРА LINUX

Вступ

Сьогодні виробники вбудованих систем використовують відкрите програмне забезпечення на базі операційних систем Linux і FreeBSD. Одним з найбільш поширених комунікаційних модулів (промислового та домашнього використання,) є маршрутизатори.

Постановка задачі

Мета роботи – дослідження продуктивності маршрутизатора на базі стандартного та оптимізованого ядер, аналіз впливу інтенсивності вхідного трафіку з різним розміром пакету на функціональні можливості маршрутизатора. Для цього вирішуються наступні задачі:

1. Дослідження поведінки системи при різному розмірі вхідного пакету.
2. Оптимізація структури ядра за критерієм підвищення продуктивності та пропускної спроможності маршрутизатора.

Теоретичні відомості

Всі функції маршрутизації знаходяться в просторі ядра Linux, тоді як більшість операцій з контролю і моніторингу (протоколи управління і маршрутизації) – демони (додатки) - запускаються в просторі, призначеному для користувача. Мережева архітектура Linux фактично побудована на механізмі переривань: мережева плата інформує ядро про прийнятий пакет через апаратне переривання. Кожне апаратне переривання обробляється обробником, так швидко, як це можливо, зупиняючи поточне завдання, яке оброблялося процесором. Поки обробник апаратного переривання не виконається до кінця, його обробка не буде перервана нічим, навіть іншим апаратним перериванням. Саме тому код обробника переривання роблять дуже коротким, тоді як

вимогливі до часу завдання, створюють з «програмними перериваннями». Програмне переривання - це штучне переривання, викликане дією ядра, і так само як і апаратне переривання, планується для обробки ядром. Обробка програмного переривання може бути перервана тільки апаратним. NET_TX_SOFTIRQ і NET_RX_SOFTIRQ це два найбільш важливі програмні переривання, які розроблені для управління операціями прийому і передачі [1].

Розглянемо докладніше процес маршрутизації.

Процес маршрутизації ініціалізувався апаратним перериванням, яке генерується мережевим пристроєм, що сигналізує про прийнятий пакет, потім обробник апаратного переривання здійснює деякі базові перевірки, і запускає відповідне програмне переривання, яке активується планувальником ядра, по можливості максимально швидко. Коли запустити програмне переривання, воно виконає всі операції з маршрутизації пакету. На рис. 1 представлена схема процесу маршрутизації. Умовно схему можна розділити на 3 частини: блок прийому, модуль, який відповідає за IP рівень і блок передачі.

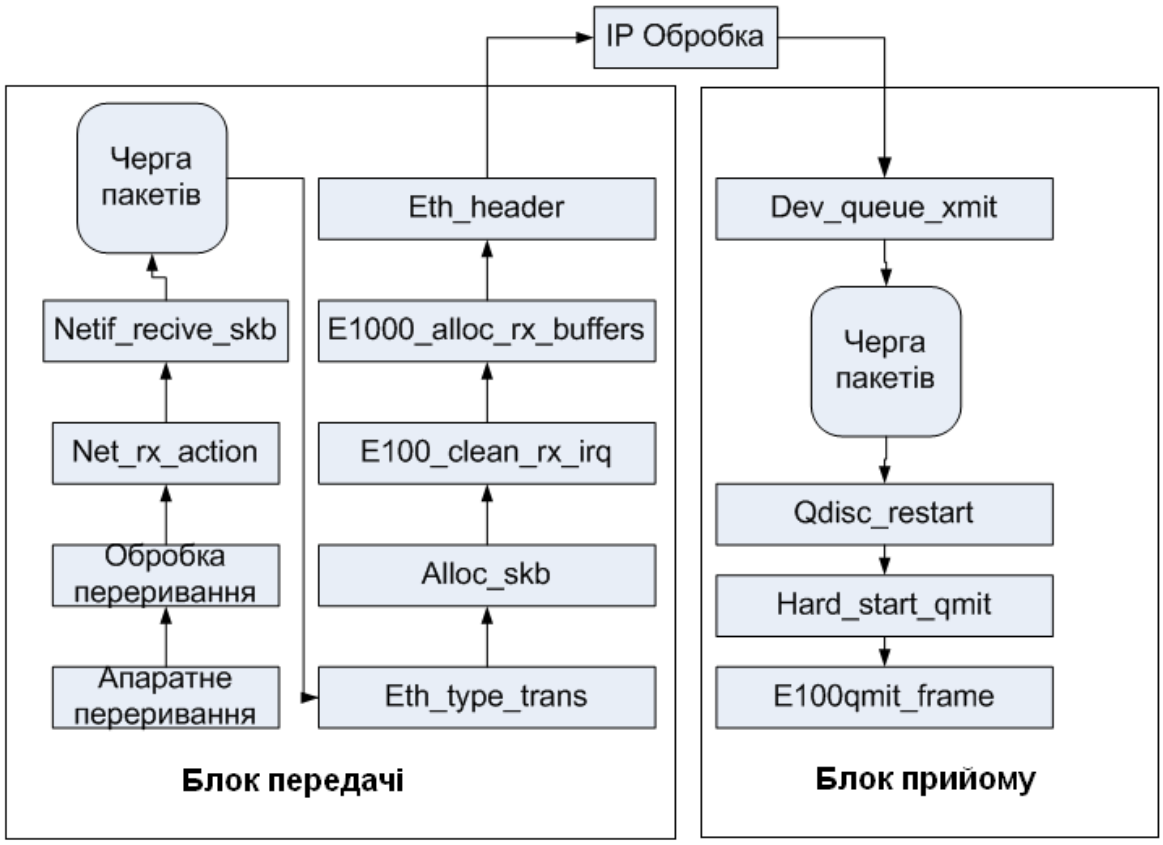

Рис. 1. Схема викликів функцій по маршрутизації в ядрі Linux

Фактично блоки прийому і передачі - це найнижчі модулі, які активізуються апаратним перериванням і плануються програмним. Вони

відповідають за мережеві інтерфейси і забезпечують деяку функціональність на другому рівні мережевого стека. Для кожного прийнятого пакету використовується дескриптор skbuf. Цей дескриптор містить ряд вказівників (pointers) на ключові поля і заголовки, які використовуватимуться при обробці на другому і третьому рівнях. У статті [3] для e1000 драйвера описаний метод оптимізації структури skbuf шляхом повторного використання цієї структури для вхідних пакетів, замість нового розподілу пам'яті. Було проаналізовано запропонований варіант оптимізації і ухвалено рішення протестувати його використання у випадку 2-ядерного роутера для змішаного голосового трафіку і трафіку даних.

Тестова конфігурація

При тестуванні повинна бути враховані як апаратна, так і програмна конфігурації, які були використані під час тесту. Окрім цього, в тестуванні важливо було використовувати такий самий реальний трафік, як і в мережах, де можуть бути присутні маршрутизатори даного типу. Отже, трафік перенаправлявся з LAN(RTL8305SC-LF) в WAN (MAX3221ECAE), для вимірювання чого було використано пристрій Agilent N2X Router Tester. На рис. 2 представлена тестова конфігурація. Генератором трафіку виступає програма tcpreply, що дозволяє використовувати трафік, зібраний у файл, дамп для якої був захоплений в реальній мережі. Для глибшого аналізу було додано декілька внутрішніх вимірювань використання процесора за допомогою oprofile.

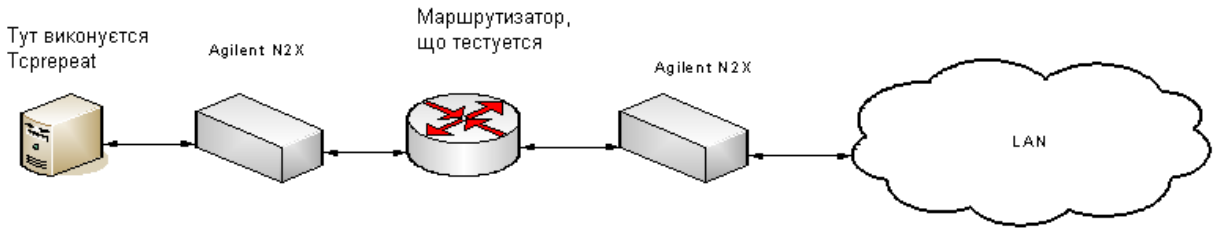

Рис. 2. Схема тестові конфігурації

Результати досліджень

Нижче представлено графік залежності розміру пакету від пропускної спроможності стандартного і оптимізованого ядер. Як видно, при малому розмірі пакету (до 200 байт) можна збільшити продуктивність на 20 %

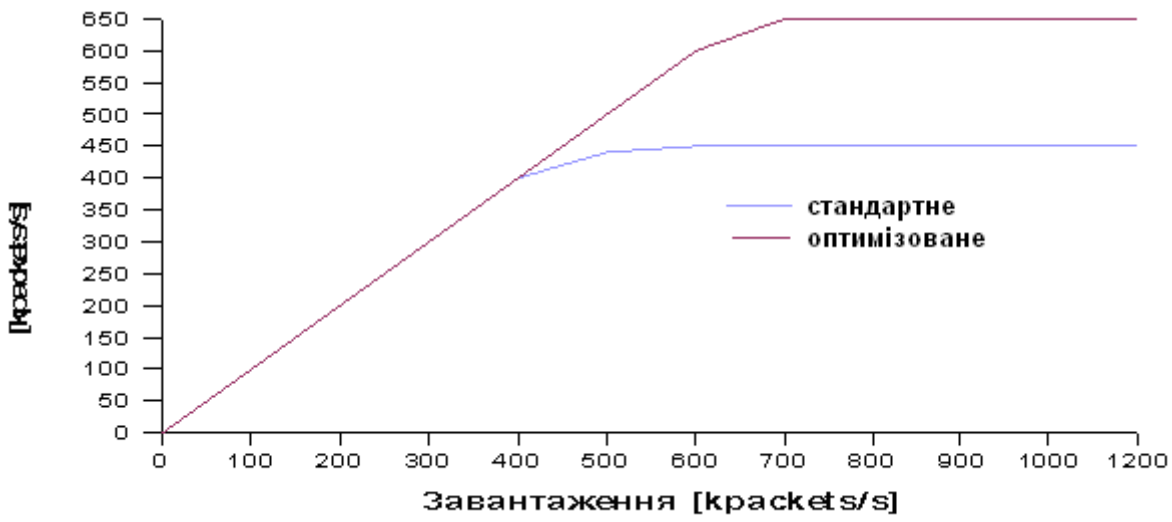

Рис. 3. Графік відноження завантаження до пропускної спроможності

Як видно з графіка, система дозволяє обробляти гігабітний трафік, отже забезпечується максимальна пропускна спроможність при розмірі пакету понад 800 байт, але при малому розмірі пакету відбувається значне зниження продуктивності. Це вузьке місце добре видно на наступному графіку, що показує відношення завантаження до пропускної спроможності при найбільш критичному розмірі IP пакету в 46 байт. На графіку розглянуті обидва випадки: при стандартному ядрі версії 2.6.21 і оптимізованому. Зрозуміло, що зменшення обсягу обчислень (деякі функції планувальника) і перерозподіл пам'яті (повторне використання skbuf для вхідних пакетів) призведе до збільшення пропускної спроможності. І дійсно, максимальна пропускна спроможність зростає з 450 Кпакетів за секунду до 660 Кпакетів за секунду. Розглянемо завантаження системи залежно від характеристик вхідного трафіку (рис. 4 та рис. 5). Перший графік показує розподіл завантаження по процесам в стандартному ядрі версії 2.6.21, другий – в оптимізованому.

Як видно з представлених графіків, завантаження процесора операціями з управління пам'яттю (як от: виділення пам'яті під структуру skbuf, її звільнення) в звичайному (стандартному) ядрі зростає лінійно, досягаючи у максимумі 25%, тоді як в оптимізованій версії вдалося зменшити завантаження процесора до менш ніж 1%. Зменшення частоти таймера планувальника завдань зберігає не більше 3% використання процесорного часу, що дає виграш близько 30К пакетів за секунду. Однак, при виконанні операцій з управління перериваннями завантаження процесора зменшується із збільшенням вхідного

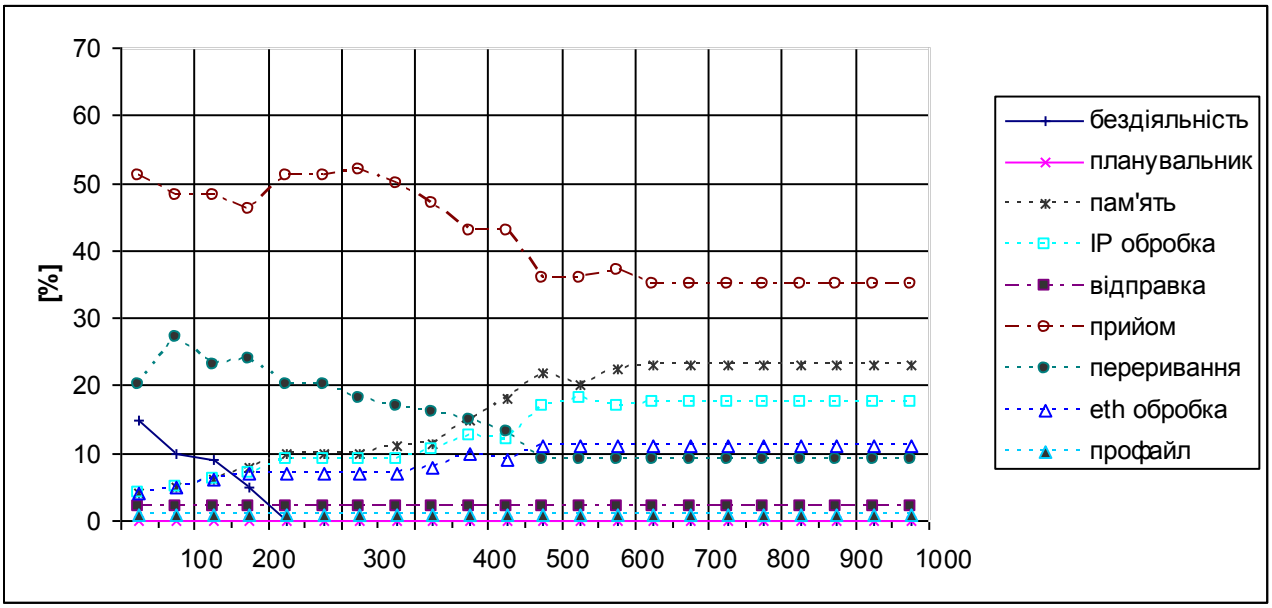

Рис. 4. Розподілення по процессам відношеннязавантаження системи до завантаження трафіком в стандартному ядрі

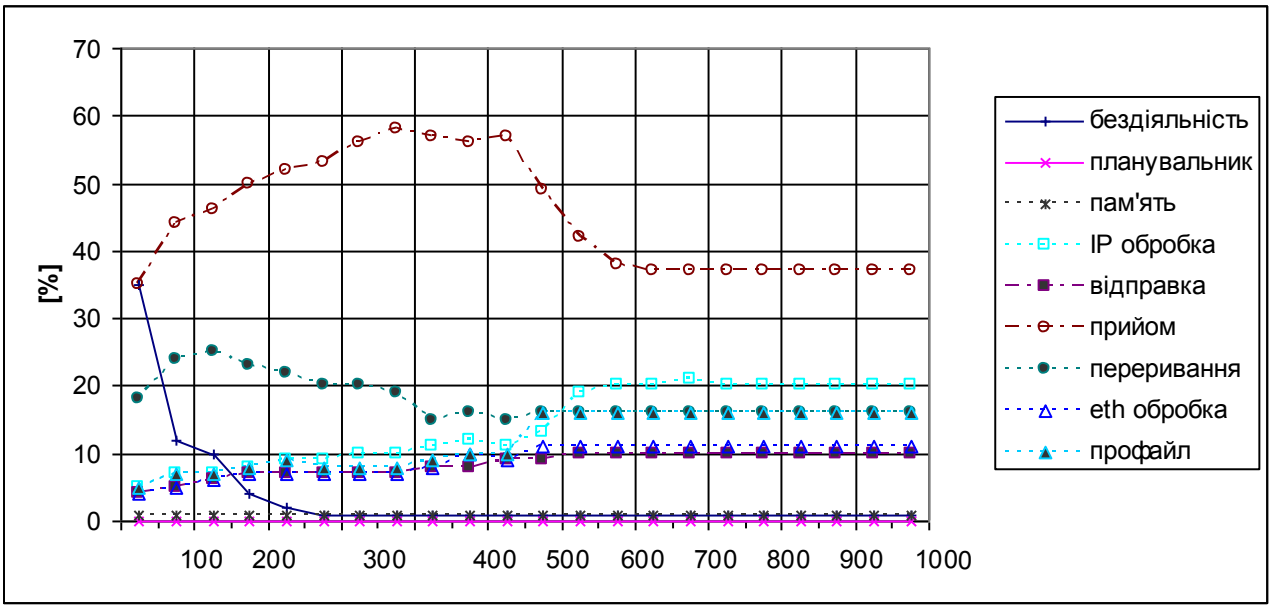

Рис. 5. Розподілення по процессам відношеннязавантаження системи до завантаження трафіком в оптимізованому ядрі

потоку. Цей ефект, в основному, пояснюється двома різними аспектами, пов'язаними з блоком прийому і блоком передачі. Коли зростає кількість вхідних пакетів блок прийому прагне зменшити кількість переривань, тоді

як блок передачі за таких самих умов використовує механізм групування пакетів шляхом пересилання декількох пакетів одразу (за одне переривання). Таким чином, кількість переривань для успішного відправлення пакетів зменшується. Всі ці ефекти можуть бути легко досліджені на представлених графіках (відношення вхідного трафіку до завантаження процесора для прийому і передачі). Відносно операцій з обробки IP і Ethernet відзначимо, що їх характеристика завантаження центрального процесора має лінійний характер і відповідає кількості відправлених пакетів за секунду. Інший аспект – завантаження процесора операціями з «блоку прийому». Як видно з графіка, ці завдання виходять найбільш обтяжливими для процесу маршрутизації. На обох графіках вони, як мінімум, на 35% завантажують процесор.

Висновки

В роботі досліджена поведінка роутерів при різному ступені завантаження трафіку, використані відомі методи для оптимізації процесу маршрутизації і проведено їх порівняння зі стандартними. Дослідження проведені на 2-процесорному маршрутизаторі, де один з процесорів працював під управлінням Linux (ядро 2.6.21), а другий - використовував спеціальну систему перекодування трафіку. Фактично в маршрутизації брав участь один процесор.

Отримані результати показали наступне:

- При малому розмірі пакету має місце значне зниження пропускної спроможності.
- Оптимізоване ядро дозволяє досягти значного підвищення продуктивності і збільшити пропускну спроможність для пакетів з малим розміром.
- Обмеження продуктивності викликані завантаженням процесора.

Подальші дослідження мають бути спрямовані на пошук інших можливостей щодо збільшення продуктивності як в зміні апаратної конфігурації (використання DMA, часткового завантаження другого процесора), так і в програмних засобах.

Література

1. *С. Сэтчел, Х. Клифорд*. Linux IP Stacks. Diasoft, 2001. – P. 673.
2. *A. Cox*. Network Buffers and Memory Management // Linux Journal, 1996. http://www2.linuxjournal.com/lj-issues/issue30/1312.html.
3. *J. A. Ronciak, J. Brandeburg, G. Venkatesan, M. Williams*. Networking Driver Performance and Measurement – e1000 A Case Study // Proc. of the 2005 Linux Symposium, Ottawa, Ontario, Canada, July 2005. – P. 154- 167.