Іван Давиденко, Ірина Пастернак
Національний університет "Львівська політехніка"
кафедра електронних обчислювальних машин

Технологія XDP: аналіз та порівняння технології обробки мережевих пакетів в Linux

© Іван Давиденко, Ірина Пастернак, 2020

Проаналізовано XDP - технологію ядра Linux з обробки мережевих пакетів - як гнучку та швидкісну альтернативу системам обходу обробки пакету мережевим стеком.

Ключові слова: мережеві технології Linux, швидкісна обробка трафіку, XDP

Analysis of XDP – Linux kernel feature of network packet processing technology – as fast flexible alternative of kernel-bypass systems.

Keywords: Linux networking, fast packet processing, XPD

Високопродуктивна обробка пакетів у програмному забезпеченні вимагає дуже жорстких обмежень часу, витраченого на обробку кожного пакета. Мережевий стек в операційних системах загального призначення, як правило, оптимізований для гнучкості, а це означає, що він виконує занадто багато операцій для кожного отриманого пакету, і відповідно не може забезпечити максимальну швидкість обробки вхідного трафіку. Це призвело до зростання популярності спеціальних наборів інструментів для обробки програмних пакетів, таких як Data Plane Development Kit (DPDK) [1]. Такі набори інструментів, як правило, повністю підміняють мережевий стек операційної системи, замість цього передають управління мережевим обладнанням безпосередньо програмі.

Технологія XDP (англ. eXpress Data Path) з'являється в ядрі ОС Linux з версії v4.8[2] разом з першою підтримкою на рівні драйвера мережевої карти (драйвер mlx4). Як альтернатива дизайну обходу мережного стеку (DPDK) ядра, вона привносить можливість програмно-керованої обробки кадрів безпосередньо в мережевий стек операційної системи. Це дає означає високошвидкісну обробку пакетів, яка легко інтегрується з існуючими системами.

XDP надає можливість програмувати невеликі модулі обробників пакетів, обов'язковим результатом роботи яких є рішення про пропускання пакету в мережевий стек ОС, скидання пакету (припинення подальшої обробки) або перенаправлення пакету в інший мережний інтерфейс. Дані обробники пишуться на C, компілюються в eBPF програму та завантажуються в ядро Linux. Завантажений модуль можна закріпити на мережевих інтерфейсах, після чого він буде виконуватись для кожного вхідного пакета. За необхідністю, модуль може бути відкріплений від одного або всіх мережевих інтерфейсів або заміненим іншим модулем обробки пакетів. Для обміну даними, статистикою та керуванням XDP-програмою доступні для використання eBPF-map - таблиці різного типу для спілкування з юзерспейсом[3]. На рис. 1. показаний потік виконання типової XDP-програми.

XDP має три режими роботи:

1. режим прямого виконання (Native);
2. режим вивантаження (Offloaded);
3. загальний режим (Generic).

В режимі прямого виконання XDP-програма запускається безпосередньо з мережевого драйвера. Найбільш поширені мережні адаптери для 10G і вище підтримують режим прямого виконання XDP.

У режимі вивантаження XDP-програма передається для виконання безпосередньо в мережеву карту замість того, щоб виконуватися на центральному процесорі. Таким чином досягається найменша затримка обробки пакетів через відсутність пересилання даних з мережевої карти до процесора, забезпечуючи більш високу продуктивність, ніж запуск у режимі прямого виконання XDP. Цей режим реалізований в драйверах SmartNIC, що містять вбудовані процесори. JIT-компілятор в ядрі транслює BPF-код XDP-модуля у інструкції мережевої карти.

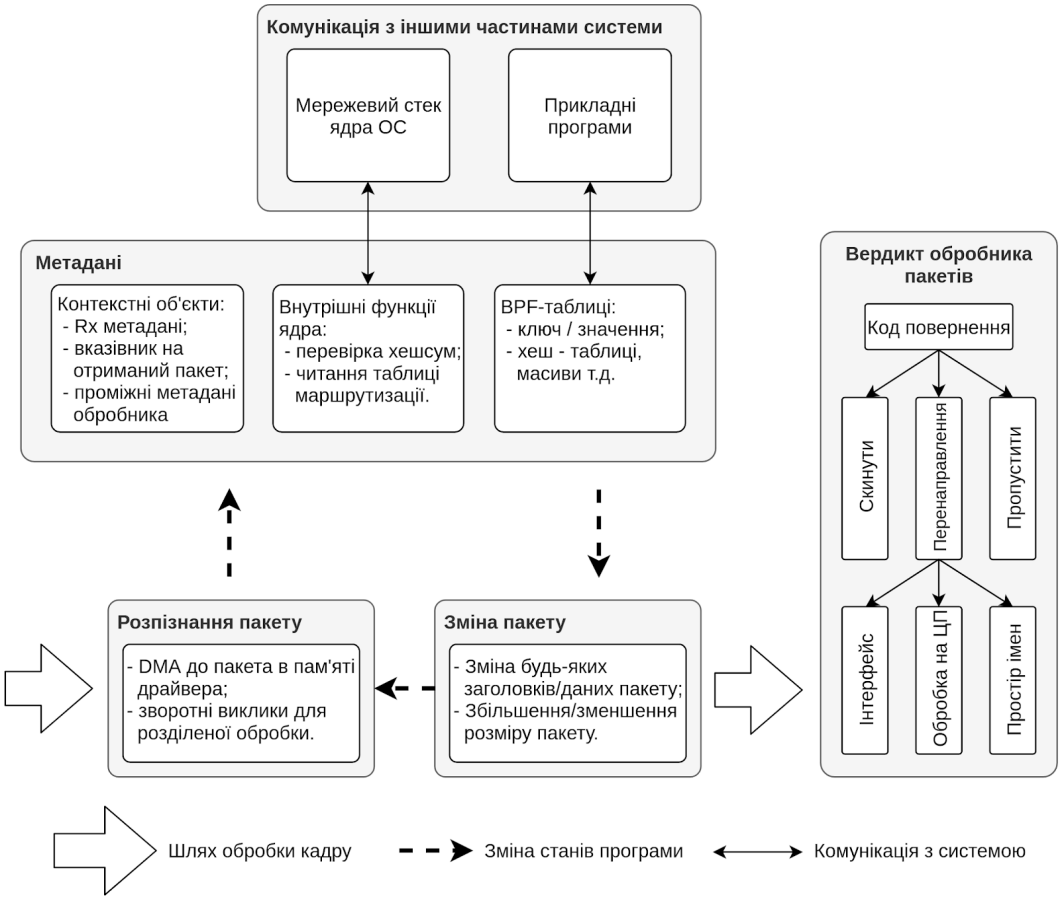

Рис. 1. Потік виконання XDP-програми.

Загальний режим виконання застосовується у випадку роботи з драйверами, які ще не підтримують інші режими запуску XDP-програми. Linux ядро надає опцію для загального режиму XDP, який не вимагає будь-яких змін в драйвері пристрою, оскільки запускається набагато пізніше, з мережного стека. Цей режим, насамперед, створений для розробників, які хочуть писати та тестувати програми на основі XDP, і не працюватимуть зі швидкістю продуктивності у власних або розвантажених режимах.

Технологія XDP включає в себе чотири основні компоненти:

1. **XDP-hook** - це основна точка входу для XDP-програми, залежить від виду
2. **Віртуальна машина eBPF -** виконує байт-код XDP-програми за допомогою JIT-компілятора.
3. **BPF-таблиці** - це структура даних типу ключ / значення, що використовується в ролі основного метода комунікації з рештою системи.
4. **Верифікатор eBPF -** система, що статично перевіряє код програмного модуля перед завантаженням в ядро ОС, щоб переконатися в її безпечності для працюючого ядра.

Разом, ці чотири компоненти утворюють потужне середовище для написання власних модулів обробників пакетів, які можуть позитивно вплинути на ефективність обробки пакетів, інтегруючись з ядром і повністю використовуючи існуючі можливості.

Оскільки модулі програм перед завантаженням в ядро ОС (в вигляді віртуальної машини eBPF) аналізуються eBPF - верифікатором, модулі мають відповідати деяким обмеженням, для впевненості в безпеці роботи ядра[4,5]. Дві основні причини накладання цих обмежень:

1. забезпечення завершення виконання програми забороною циклів та обмеженням розміру програми;
2. гарантія безпечних звернень до пам’яті.

Через ці причини, верифікатор не пропускає eBPF-програму, поки не зможе впевнитися в відповідності цим стандартам. Іноді це призводить до помилкової відмови завантажувати модуль в ядро ОС. Ще одним обмеженням є те, що до кожного мережного інтерфейсу можна приєднати лише одну програму XDP.

Головною перевагою технології є висока швидкодія. Оцінка продуктивності програм на технології XDP показує швидкість обробки пакетів до 24 мільйонів пакетів на секунду[6,7]. Порівняльна характеристика швидкості прийняття рішення про скидання обробки пакета наведена на рис. 2.

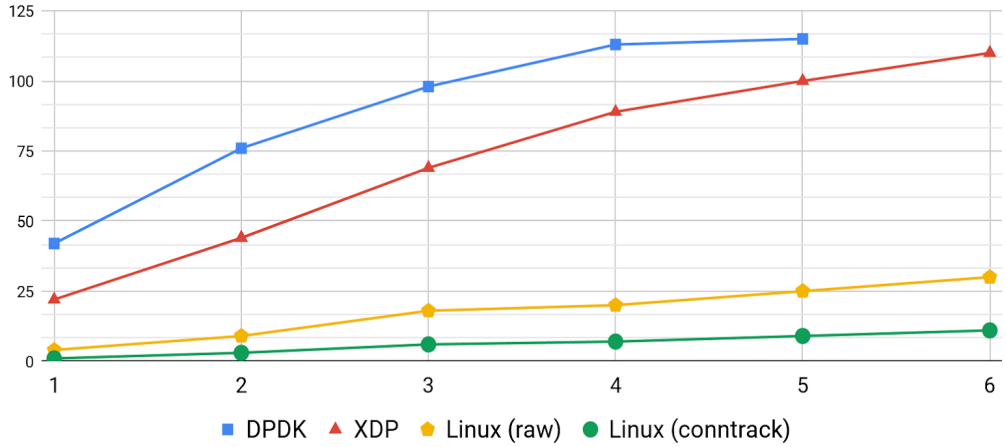

Рис. 2. Залежність продуктивності скидання трафіку (в мільйонах пакетів на секунду) від кількості логічних процесорів для різних технологій.

Ці результати співставні з піковими швидкостями DPDK-додатку[8] на тому ж апаратному забезпеченні. При цьому, система XDP має декілька переваг над DPDK та іншими технологіями, що базуються на ідеї обходу мережного стеку ядра.

Переваги технології XDP над аналолгічними системами швидкісної обробки пакетів в ОС:

1. Інтегрується з стандартним ядром, зберігаючи керування апаратурою в ядрі ОС.
2. Не потребує змін у конфігурації мережі та інструментах керування. Крім того, будь-який мережний адаптер з драйвером Linux може підтримувати XDP. Для базової роботи XDP-модуля розробникам драйверів мережевих карт не потрібно реалізовувати додаткові функції. Є можливість модифікації існуючих драйверів для більш раннього виконання XDP-модуля.
3. Дозволяє використовувати функції мережного стека ОС, такі як таблиці маршрутизації і стек TCP разом зі швидкою обробкою пакетів.
4. Аналізатор коду гарантує стабільність роботи як eBPF-програми (XDP-модуля), що розміщується в ядрі ОС.
5. Не вимагає повторних, затратних за ресурсами, ін'єкцій пакетів з простору користувача в простір ядра для роботи з трафіком через інтерфейси стандартного мережевого стеку ОС.
6. Є прозорим для прикладних програм, запущених на ОС, що дозволяє динамічно розгортати та модифікувати системи на базі XDP без жодного впливу на роботу прикладних програм.
7. Має можливість динамічного перепрограмування без будь-якого переривання роботи мережевого стеку. Це означає що функціонал системи на базі XDP може бути доданий або видалений (коли він більше не потрібен), без переривання мережного трафіку і що система обробки пакетів може динамічно реагувати на зовнішні умови.

Висновки. Технологія XDP пропонує функціонал ОС Linux, що може використовуватися для реалізації застосунків вищого рівня, що потребують швидкої обробки трафіку, такі як системи балансування навантаження, брандмауери[9], програмні маршрутизатори та програмовані фільтри мережевого трафіку.

Література

1. Jesper Dangaard Brouer, Red Hat; XDP - eXpress Data Path Used for DDoS protection; 2017.
2. Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann; The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel – 2018; CoNEXT '18: Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies ст. 54–66.
3. David Ahern, Cumulus Networks; Leveraging Kernel Tables with XDP, 2018.
4. М. В. Широков, Facebook; XDP: 1.5 years in production. Evolution and lessons learned, 2018.
5. Довідник ядра ОС Linux: документація до бібліотеки libbpf [Електронний ресурс] / 2018 - Режим доступу до ресурсу: https://kernel.org/doc/readme/tools-lib-bpf-README.rst.
6. Björn Töpel, Magnus Karlsson; Документація до мережної частини ядра ОС Linux [Електронний ресурс]; 2018 - Режим доступу до ресурсу: https://kernel.org/doc/html/v4.18/networking/af_xdp.html
7. Magnus Karlsson, Bjorn Topel, Intel; The Path to DPDK Speeds for AF XDP; 2018.
8. Anant Deepak, Shankaran Gnanashanmugam, Richard Huang, Puneet Mehra, Facebook; eBPF / XDP based firewall and packet filtering; 2018.
9. Jesper Dangaard Brouer, Red Hat; XDP – challenges and future work - 2018.