УДК 681.3, 004.728

Л. О. Березко, Я. П. Гурик
Національний університет “Львівська політехніка”,
кафедра електронних обчислювальних машин

РЕАЛІЗАЦІЯ МЕРЕЖЕВОГО СТЕКА ПРОТОКОЛІВ ДЛЯ ОС Linux З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ DPDK

© Березко Л .О., Гурик Я. П., 2017

Розглянуто завдання підвищення швидкодії наявних мережевих стеків протоколів. Проаналізовано засоби високошвидкісної обробки пакетів. Запропоновано архітектуру мережевого стека протоколів.

Ключові слова: мережевий стек, мережевий адаптер, DPDK.

L. Berezko, Y. Huryk
Lviv Polytechnic National University,
Computer Engineering Department

IMPLEMENTATION OF DPDK BASED TCP/IP PROTOCOLS STACK FOR OS Linux

© Berezko L. O., Huryk Y. P., 2017

The performance problem of network stacks currently implemented in operating systems. Analysis of frameworks for high-performance packet IO. An architectural diagram of the custom network stack.

Key words: network stack, NIC, DPDK.

Вступ

Сучасні мережеві стеки протоколів операційних систем не можуть повністю використати можливості 10-гігабітних мережевих адаптерів [1, 2] що зумовлено значними накладними витратами, спричиненими архітектурою мережевих стеків, орієнтованою насамперед на універсальність. Процесор швидко стає тим вузьким місцем, який не забезпечує навіть базового ефективного оброблення пакетів невеликого обсягу, не кажучи вже про маршрутизацію або ACL (Access Control List). Для вирішення цієї проблеми створено різні технології [3, 4]. Всі вони мають схожий механізм роботи, основною ідеєю якого є надання програмному засобу користувача прямого доступу до мережевого адаптера повз засоби операційної системи. Функції мережевого рівня стека протоколів TCP/IP виконує той самий програмний засіб, тоді як у стандартній ситуації цим займається операційна система. Пропонується варіант використання таких технологій.

Стан проблеми

Поява 10-гігабітних мережевих адаптерів та багатоядерних процесорів зумовила потребу в технологіях обходу мережевого стека операційної системи [5]. Найпопулярнішими такими технологіями тепер є: DPDK (Data Plane Development Kit) [7], netmap [6] та PF_RING ZC. DPDK – набір бібліотек та драйверів мережевих адаптерів, який не тільки містить базові засоби відправляння та отримання пакетів, але й надає багато додаткових функцій та засобів типу менеджерів черг, буферів та пам’яті. Ще однією важливою відмінністю DPDK від вищезгаданих альтернатив є те, що його драйвери повністю відключають мережевий стек операційної системи

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

незалежно від того, використовує якийсь програмний засіб користувача його API чи ні. Хорошим прикладом того, що може дати його використання, є спеціальна версія DPDK Open vSwitch, яка демонструє істотний приріст продуктивності.

Незважаючи на великі функціональні можливості, DPDK [7, 8] є лише інструментом, на основі якого потрібно побудувати повнофункціональне рішення. В сучасному розумінні мережевий стек повинен мати такі компоненти: протокол TCP [9], протокол IP (забезпечення маршрутизації), протокол ARP (протокол пошуку адреси)). Для досягнення необхідного рівня функціональності також треба реалізувати хоча б часткову підтримку стандартних системних викликів для роботи з мережевою підсистемою: socket, bind, listen, accept, connect, shutdown. Корисною також є підтримка протоколу UDP. Для цього треба реалізувати такі системні виклики: recvfrom, sendto. Це дасть змогу відправляти на порти транспортних протоколів будь-який програмний продукт, який використовує вищезгадані системні виклики. Вже зроблено спроби реалізувати мережевий стек на базі DPDK: (F-Stack, ANS). У них усіх є проблемні особливості, зокрема значна складність реалізації з погляду пересічних розробників.

Постановка задачі

Спроектувати та реалізувати простий мережевий стек протоколів TCP/IP на базі DPDK. Стек повинен забезпечити більшу перепускну здатність для основних сценаріїв використання, ніж стандартний мережевий стек ОС Linux, гнучкість та прості розширення порівняно з альтернативами.

Розв’язання задачі

Першим кроком у розв’язанні задачі є належне налаштування DPDK. Для цього необхідний мережевий адаптер, який підтримує цей фреймворк або засоби віртуалізації (Virtual Box). Другим кроком є налаштування сторінок пам’яті достатньо великого обсягу, що необхідно для зниження навантаження на буфер асоціативної трансляції центрального процесора.

Основні етапи оброблення пакетів з використанням DPDK такі:

1. Вхідні пакети потрапляють у кільцевий буфер.
2. Якщо в буфері є нові дескриптори пакетів, додаток звертається до буферів DPDK, що містяться у спеціально виділеному пулі пам’яті, через вказівники в дескрипторах пакетів.
3. Якщо в кільцевому буфері немає пакетів, то додаток опитує мережеві адаптери, які перебувають під управлінням DPDK, а потім знову звертається до кільцевого буфера.

Основними компонентами мережевого стека TCP/IP є протоколи ARP, ICMP, IP та TCP. Структурна схема архітектури вдосконаленого мережевого стека показана на рис. 1.

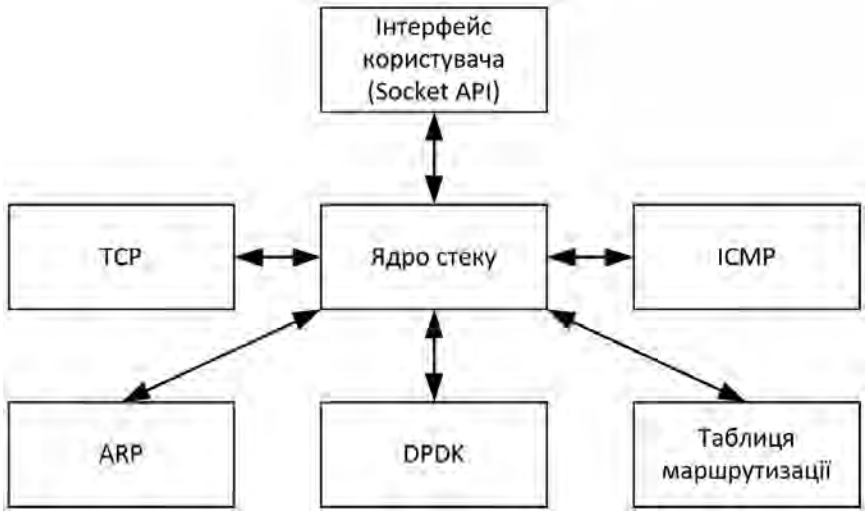

Рис. 1. Структурна схема архітектури мережевого стека

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

Найскладнішою частиною реалізації є забезпечення станів з'єднання протоколу TCP [9]. Алгоритм роботи протоколу TCP показано на рис. 2. Основним завданням TCP є забезпечення контролю надійного зв'язку між двома вузлами мережі. Ініціалізація з'єднання відбувається відправленням пакета з прапорцем SYN, що переводить сервер зі стану LISTEN в SYN_RCVD. Після цього сервер та клієнт обмінюються ще декількома пакетами і сервер переходить в стан ESTABLISHED. Процедура завершення з'єднання полягає в так званому чотирифазному handshake ("потисканні рук"). Ініціатор з'єднання відправляє пакет з прапорцем FIN, що переводить отримувача у стан CLOSE_WAIT. Під час наступних фаз handshake ініціатор і отримувач послідовно проходять стани FIN_WAIT_2, TIME_WAIT, LAST_ACK і, на завершення, – CLOSED.

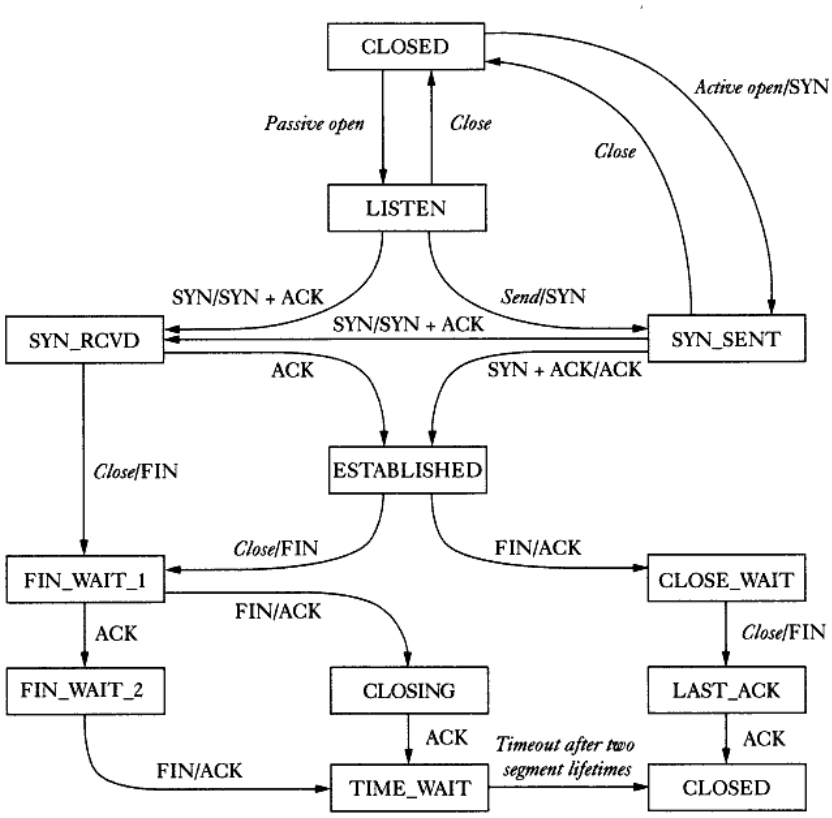

Рис. 2. Алгоритм роботи протоколу TCP

Опис Ethernet-фрейму мовою C, необхідного для реалізації мережевого стека, показано на рис. 3. Кадр Ethernet містить MAC-адреси отримувача та відправника, тип протоколу наступного рівня та власне дані.

```
struct eth_hdr
{
    unsigned char dmac[6];
    unsigned char smac[6];
    uint16_t ethertype;
    unsigned char payload[];
} __attribute__((packed));
```

Рис. 3. Опис Ethernet-фрейму мовою C

Протокол ARP [10] потрібен для отримання MAC-адреси за заданою IP-адресою. Для опису ARP-пакетів пропонується використати структури, показані на рис. 4. Основний заголовок містить

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua

тип протоколу передавання (1 байт для Ethernet), тип мережевого протоколу, довжину адреси протоколу передавання (6 байт для MAC-адреси), довжину адреси мережевого протоколу (4 байти для IP), код операції та власне дані, що залежать від вищеописаних значень. Для стандарту IPv4 це будуть MAC- та IP-адреси відправника та отримувача відповідно.

```
struct arp_hdr
{
    uint16_t hwtype;
    uint16_t protype;
    unsigned char hwsize;
    unsigned char prosize;
    uint16_t opcode;
    unsigned char data[];
} __attribute__((packed));

struct arp_ipv4
{
    unsigned char smac[6];
    uint32_t sip;
    unsigned char dmac[6];
    uint32_t dip;
} __attribute__((packed));
```

Рис. 4. Опис заголовка ARP-пакета та його тіла для запиту в стандарті IPv4

Висновки

У роботі розглянуто проектування та розроблення мережевого стека протоколів TCP/IP на базі DPDK. Запропоновано основні компоненти стека та описано їх взаємодію. Розроблено структурну схему архітектури стека та алгоритм програмної реалізації. Цей варіант мережевого стека TCP/IP не забезпечує такої універсальності, як стандартний в ОС Linux, але його швидкодія значно вища за рахунок прямого запису пакетів у простір користувача та спрощеної логіки роботи, що може бути істотною перевагою під час його використання.

1. Rizzo L. Netmap: a novel framework for fast packet I/O // in USENIX Annual Technical Conference, April 2012. 2. Impressive Packet Processing Performance Enables Greater Workload Consolidation // in Intel Solution Brief. Intel Corporation, 2013, Whitepaper. 3. Fusco F. and Deri L. High Speed Network Tranc Analysis with Commodity Multi-core Systems // in Internet Measurement Conference, November 2010, pp. 218–224. 4. Dobrescu M., Argyraki K., and Ratnasamy S. Toward predictable performance in software packet-processing platforms // in Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, ser. NSDI'12. Berkeley, CA, USA: USENIX Association, 2012. 5. Bolla R. and Bruschi R. Linux Software Router: Data Plane Optimization and Performance Evaluation // Journal of Networks, vol. 2, no. 3, June 2007. 6. García-Dorado J. L., Mata F., Ramos J., Santiago del Río P. M., Moreno V., and Aracil J. Data Tranc Monitoring and Analysis / E. Biersack, C. Callegari, and M. Matijasevic, Eds. Berlin, Heidelberg: Springer-Verlag, 2013, ch. High-Performance Network Tranc Processing Systems Using Commodity Hardware. 7. Deri L. nCap: Wire-speed Packet Capture and Transmission // in End-to-End Monitoring Techniques and Services. IEEE, 2005. 8. Data Plane Development Kit: Programmer's Guide, Revision 6. Intel Corporation, 2014. 9. Han S., Jang K., Park K. and Moon S. PacketShader: a GPU-accelerated Software Router // SIGCOMM Computer Communication Review, vol. 40, no. 4, 2010. 10. Bonelli N., Pietro A. Di, Giordano S., and Procissi G. On Multi–Gigabit Packet Capturing With Multi–Core Commodity Hardware // in Passive and Active Measurement. Springer, 2012.

Lviv Polytechnic National University Institutional Repository http://ena.lp.edu.ua