Chair of Decentralized Systems Engineering
Department of Informatics

Fast User-space Packet Processing

General information	**Advisor**	Simon Ellmann
	Email	simon.ellmann@tum.de
	Date	14.02.2022
Type	Bachelor's Thesis / Master's Thesis / Guided Research	
Description	In the past, high-speed packet processing was performed in kernel-space due to the high overheads of Linux's network APIs. However, the kernel environment poses restrictions on programming (e.g., APIs are not guaranteed to be stable between releases, no freedom of programming language choice) and debugging of code, and software errors in kernel code may have far-reaching consequences. To overcome the limitations of in-kernel code while maintaining packet processing speeds, network functions were shifted from kernel to user-space via custom frameworks. This happened in two steps: First, frameworks consisting of kernel modules and user-space libraries (e.g., PF_RING, netmap) emerged. Later, full user-space driver frameworks (e.g., DPDK, Snabb) moved packet processing entirely into user-space. In recent years, however, new APIs were added to the Linux kernel to improve I/O performance of user-space applications. One of these APIs is AF_XDP, a new socket type built on XDP designed to pass packets from kernel to user-space as fast as possible, omitting the kernel TCP/IP stack. Another one is io_uring, an interface for asynchronous I/O via queues that eliminates the system call overhead of read() and write() on file descriptors (e.g., sockets).	

With these new APIs and the ongoing efforts to improve performance of user-space I/O on top of the kernel network stack (e.g., zero-copy added to AF_XDP) the question arises whether the tides are slowly turning – away from user-space driver frameworks like DPDK back to packet processing on top of Linux's network APIs.

Previous research shows promising results for APIs like AF_XDP and io_uring but lacks a general evaluation of current approaches to fast user-space packet processing. Besides, it does not reflect ongoing changes[1] in the Linux kernel as well as in packet processing frameworks that have incorporated some of the new APIs.

Therefore, the overall goal of this project is to evaluate current approaches to fast user-space packet processing in terms of their architectural differences and their performance. For the performance evaluation, one or multiple applications (e.g., a packet forwarder) may be implemented in C, C++ or Rust.

Keywords

Linux, packet processing, DPDK, AF_XDP, io_uring, zero-copy

Goals

Concrete outcomes

1. Understand overheads of the Linux kernel network stack
2. Understand what approaches different packet processing frameworks follow
3. Evaluate the packet processing frameworks and Linux's APIs from an architectural point of view (i.e., complexity, usability, software/hardware support, …)
4. Evaluate the performance of packet processing frameworks and Linux's APIs

Bonus points

5. Submit a patch to one of the frameworks or the Linux kernel (e.g., fix a bug, implement a missing feature, …)
6. Propose a new approach to fast user-space packet processing

[1] https://git.kernel.org/pub/scm/linux/kernel/git/bpf/bpf-next.git/commit/?id=a23b3f5697e6cf8affa7adf3e967e5ab569ea757

Prerequisites	**Preferred** • Course "Basic Principles: Operating Systems and System Software (IN0009)" • Course "Introduction to Computer Networking and Distributed Systems (IN0010)" • Knowledge of a system programming language (C, C++, Rust, …) • Knowledge of packet processing frameworks (e.g., DPDK)
References	1. The Path to DPDK Speeds for AF_XDP. LPC '18. http://vger.kernel.org/lpc_net2018_talks/lpc18_paper_af_xdp_perf-v2.pdf 2. Accelerating networking with AF_XDP. https://lwn.net/Articles/750845/ 3. Revisiting the Open vSwitch Dataplane Ten Years Later. SIGCOMM '21. https://dl.acm.org/doi/10.1145/3452296.3472914 4. Understanding Host Network Stack Overheads. SIGCOMM '21. https://dl.acm.org/doi/abs/10.1145/3452296.3472888
Application process	If you are interested send me an e-mail or drop by my office.