Adding XDP support to Open Nic Driver

Alexandru Gabriel Bradatan, Marco Molè

April 11, 2024

Contents

1 Project data

- Project supervisor(s): Gianni Antichi & Sebastiano Miano
- Describe in this table the group that is delivering this project:

Last and first name	Person code	Email address
Alexandru Gabriel Bradatan	10658858	alexandrugabriel.bradatan@mail.polimi.it
Marco Molè	10676087	marco.mole@mail.polimi.it

- Describe here how development tasks have been subdivided among members of the group:
  - Most of the code was written in pair programming sessions.
    - We found the pair programming approach really useful to delve into the XDP API, which sometimes is poorly documented.
  - Bradatan worked on a eBPF testing suite that was not deemed necessary at the end.
    - Link to the tests: `https://github.com/alexbradd/ebpf-xdp-test-suite`
- Links to the project source code: `https://github.com/marcomole00/open-nic-driver/tree/xdp-support`
  Note: our work is in the `xdp-support` branch

2 Project description

2.1 Project goals

The goal of this project is to add support for *eXpress Data Path* (XDP) for the driver of the **AMD OpenNIC project**.

XDP is a high-performance data path for network packets present in the Linux kernel since version 4.8. It enables users to install packet processing programs in the kernel that are executed before the invocation of the standard network stack. These programs are written and executed in eBPF, a subset of C that can be verified to terminate and to not corrupt kernel memory making it safe to use for extending the kernel's functionality.

2.2 Importance to the AOS course

This project is relevant to the Advanced Operating System course because it requires extending the functionality of a Linux driver. Doing so requires understanding:

- How to write code that correctly runs in kernel context
- How to Linux kernel modules are written and compiled

Moreover, implementing XDP requires knowledge of the Linux networking stack, which even though is not a topic of the AOS course but of the supervisors' Network Computing course, it is still an important part of the Linux kernel.

2.3 Design and implementation

We based our implementation based on some previous work done on the Intel driver [1] and some documentation material provided by RedHat [2].

Our work is mainly focused in `onic_netdev.c`, where we added all the `onic_xdp_` functions, the most relevant of which are:

- `onic_xdp()`: entry-point of for the XDP subsystem, does some initialization
- `onic_run_xdp()`: main function that executes the XDP program on each received packet and handles the return value of the program; invoked for each packet by the main driver polling loop in `onic_poll()`
- `onic_xdp_xmit()`: used to submit XDP packets for transmit

We have also added support for querying XDP statistics through `ethtool`. Those changes have been done mainly to the `onic_ethtool.c` file.

3 Project outcomes

3.1 Concrete outcomes

We have produced a series of commits on a fork of the OpenNIC repository implementing XDP and a repo containing some tests for the various return values of XDP programs.

Links:

- Commits implementing XDP: `https://github.com/marcomole00/open-nic-driver/commits/xdp-support/`
- Diff w.r.t. upstream: `https://github.com/Xilinx/open-nic-driver/compare/main...marcomole00:open-nic-driver:xdp-support`
- Testing repo: `https://github.com/alexbradd/ebpf-xdp-test-suite`

[1] `https://patchwork.ozlabs.org/project/intel-wired-lan/patch/20200902203222.185141-1-anthony.l.nguyen@intel.com/#2535008`

[2] `https://people.redhat.com/lbiancon/conference/NetDevConf2020-0x14/add-xdp-on-driver.html`

3.2 Learning outcomes

- We gained insight on the inner workings of the Linux networking API and how to write drivers that utilize it
- We gained insights on how the linux kernel code is structured and how to navigate it effectively when searching for a specific function. We made extensive use of the *Elixir Cross Referencer* [3].
- We learned to search in commit messages or mailing list archives when in doubt about certain pieces of code.
- We learned how to correctly setup the environment for kernel module programming.

3.3 Existing knowledge

Bradatan followed profs. Antichi and Miano's Network Computing course last year, Molè is following it this semester. The course provides insights on how the Linux network stack is implemented.

This project focused on some topics taught in the Network Computing course. To avoid increasing too much the scope of the AOS course and to avoid overlap with the Network Computing course we do not feel like recommending adding networking topics to the AOS course. Moreover, profs. Antichi and Miano were very available to answering any questions and providing course materials one the topics needed for the project.

3.4 Problems encountered

The main problem we encountered was the lack of (or when present poor quality of) documentation on some parts of the linux networking stack and on OpenNIC driver code. This required some hunting around sources and cross-referencing with other driver's implementations to get resolved.

Another thing we had to consider is that the XDP API and the network stack data structures changed over time and that led to some inconsistency in kernel versions previous to 5.3.0 that hindered us from implementing the *onic_xdp_xmit_frame* function, and thus the support for `XDP_TX` and `XDP_REDIRECT` in those older versions.

4 Honor Pledge

(**This part cannot be modified and it is mandatory to sign it**)

I/We pledge that this work was fully and wholly completed within the criteria established for academic integrity by Politecnico di Milano (Code of Ethics and Conduct) and represents my/our original production, unless otherwise cited.

I/We also understand that this project, if successfully graded, will fulfill part B requirement of the Advanced Operating System course and that it will be considered valid up until the AOS exam of Sept. 2024.

Group Students' signatures

Alexandru Gabriel Bradatan

Marco Molè

[3] `https://elixir.bootlin.com`