cesnet

Accelerating Suricata with DPDK prefilters

Lukas Sismis

Agenda

- Introduction to packet acquisition modules of Suricata
- Experiments with DPDK packet capture method
- Concept of DPDK Prefilter and Proxy

Intro

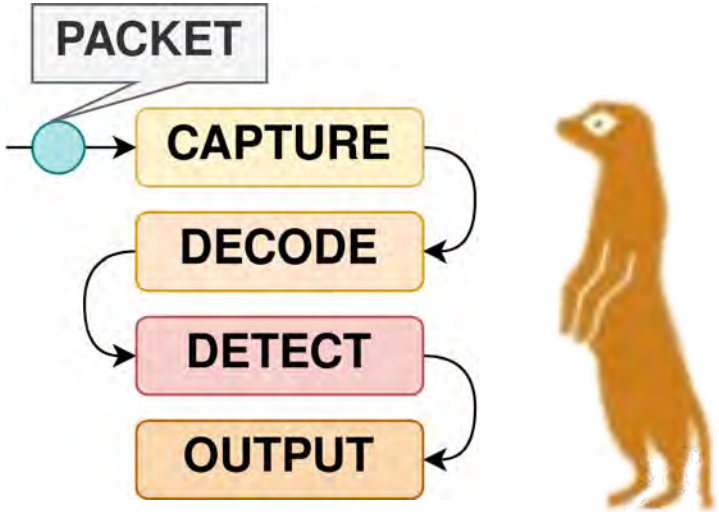
- Name > Lukas Sismis
- Role#1 > Researcher at CESNET
- Role#2 > Phd Student at Brno University of Technology, Faculty of Information Technology

CESNET does:

- tools for network monitoring and security
- high-speed real-time encrypted traffic analysis
- DDoS detection and mitigation

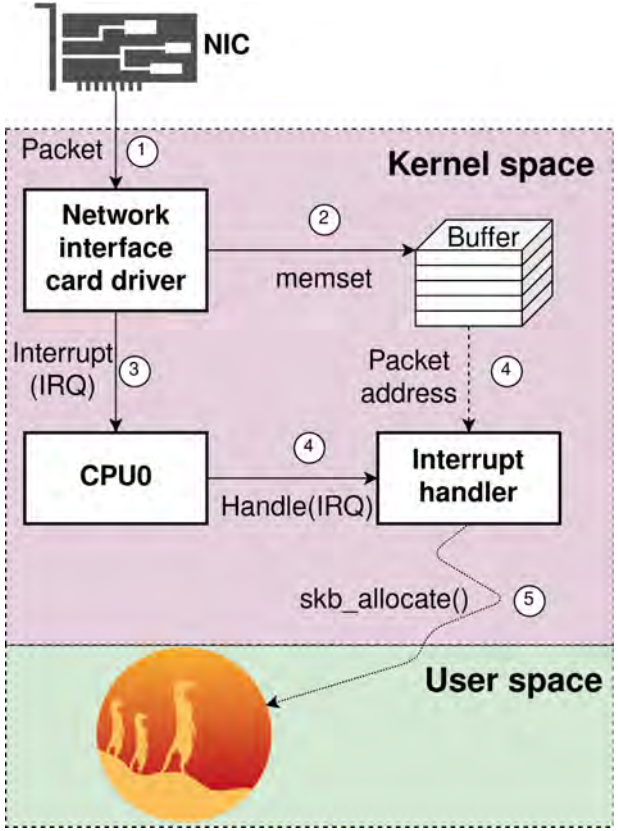
Getting more out of Suricata

- Suricata consists of 4 modules
- Each module takes a toll on performance and throughput
- Optimizing one module helps to increase the overall performance



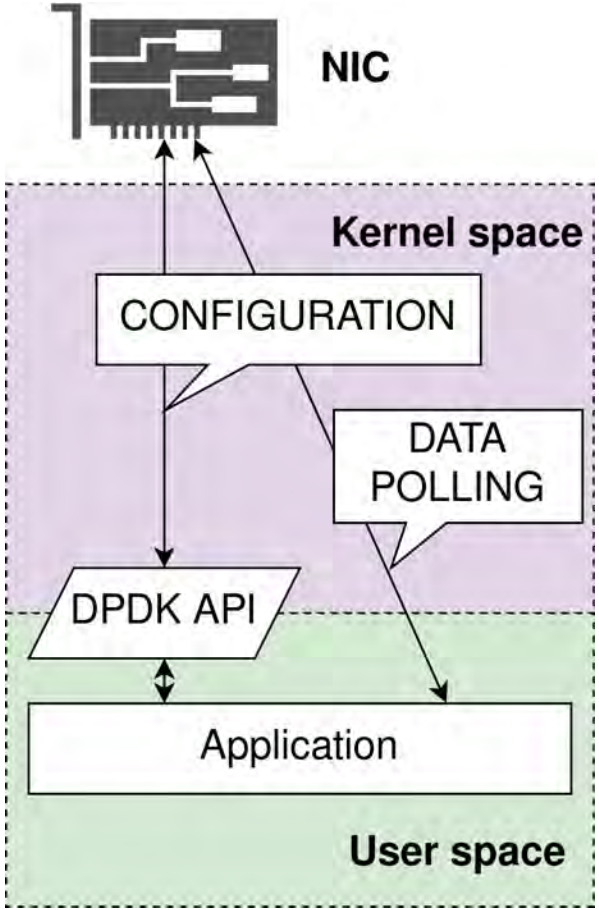
Suricata packet acquisition module

- Captures packets from the NIC
- The module connects Suricata with packet capture libraries
- Examples of Suricata PA modules: AF_PACKET, NFQUEUE, PF_RING or others
- Modules vary in performance



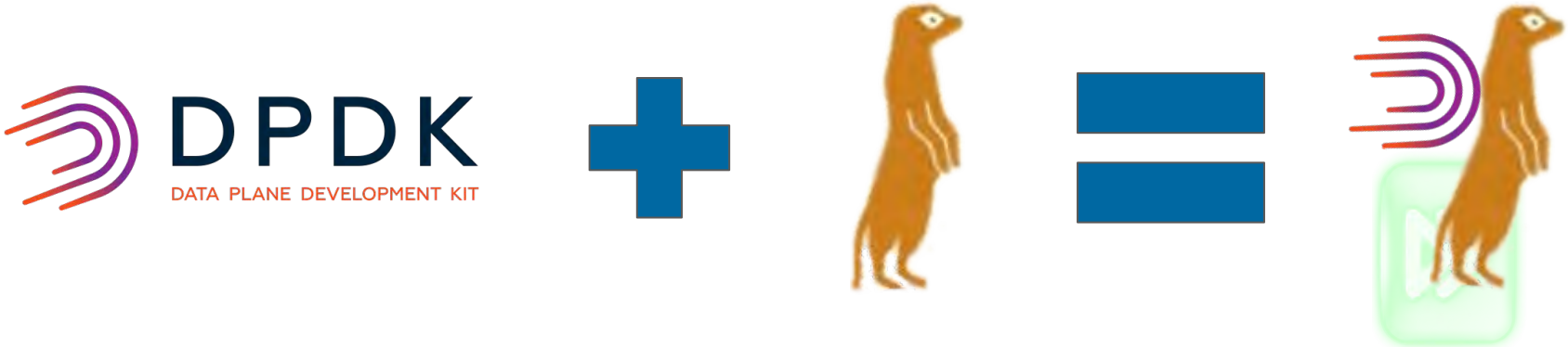
DPDK - Data Plane Development Kit

- Set of network data plane libraries and network interface controller
- Moves packet processing from the kernel to processes running in the user space
- Uses polling for packet acquisition
- Includes support for HW offloads from vendors like Intel, Mellanox, Xilinx
- Support virtual interfaces, Docker
- Standardized approach supported by many NIC vendors



DPDK as a Suricata capture method

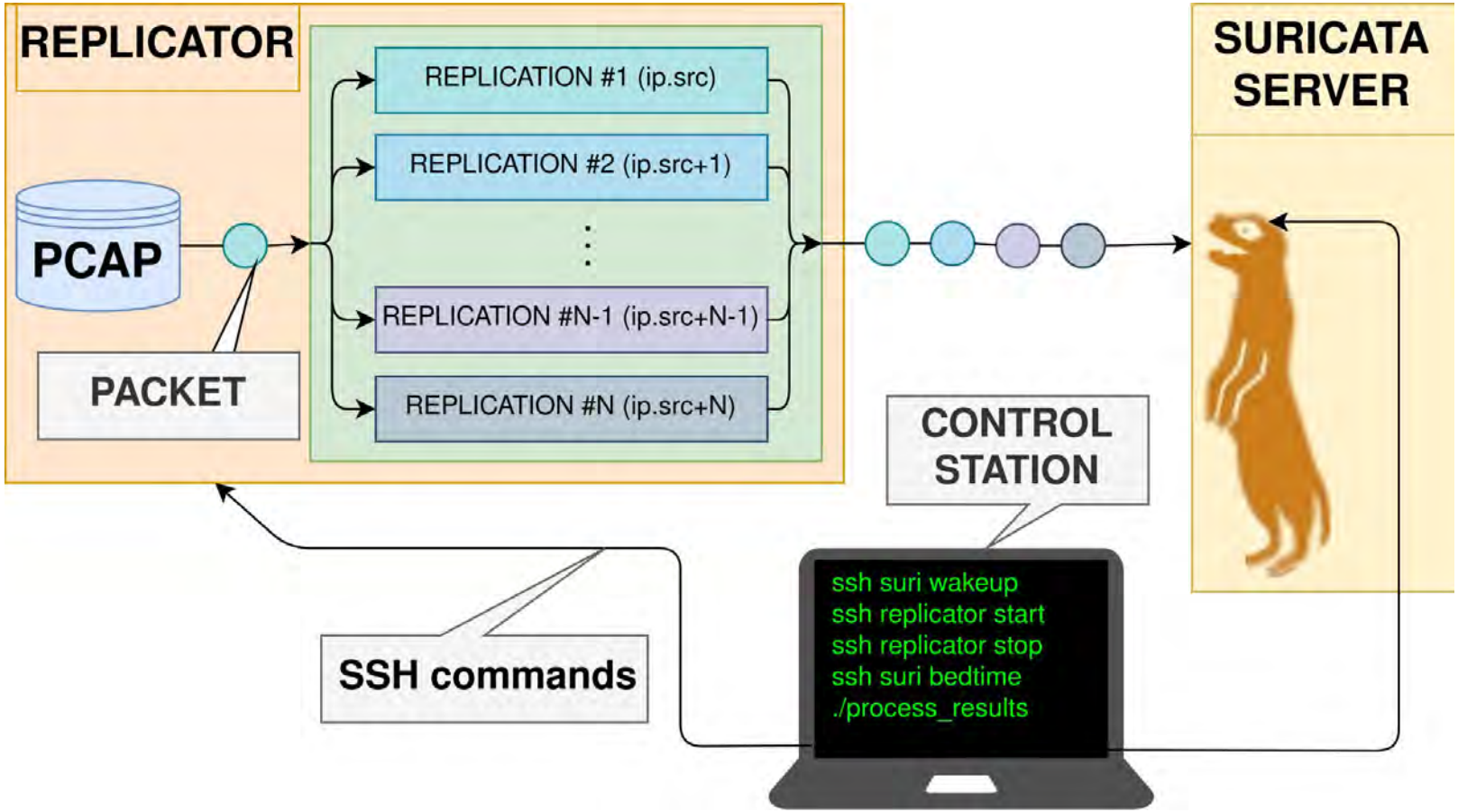
- Implemented using the standard packet acquisition API
- Implemented for worker runmode in both IPS and IDS
- Focused on supporting physical NICs running on physical machines
- Runs on DPDK 19.11+



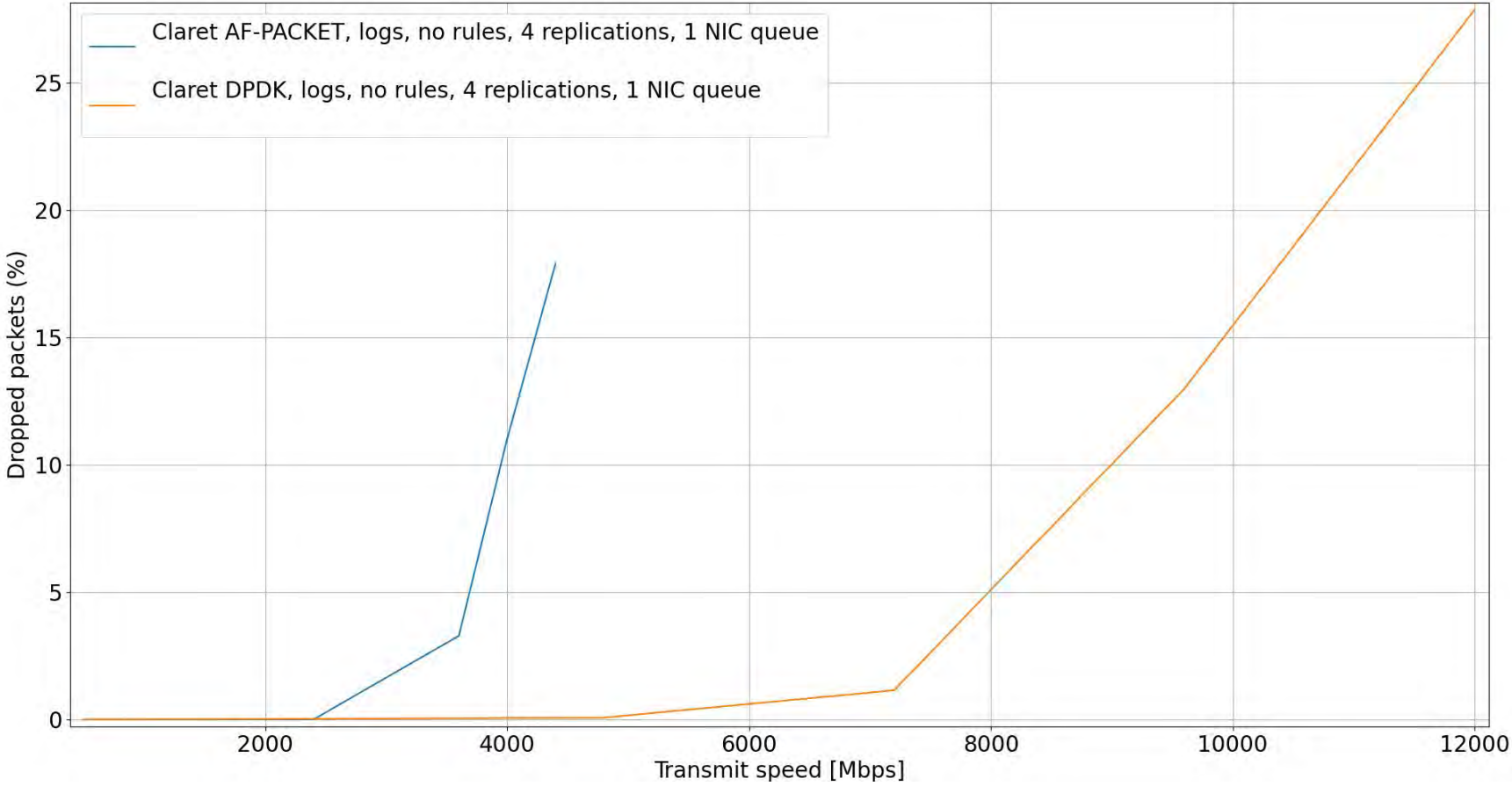
Experimental setup

- Tested performance of tuned AF_PACKET and DPDK capture interfaces
- Worker threads mapped to the NIC queues in 1:1 ratio
- Tested with rules ET Open (21314 rules enabled) in IDS mode
- Suricata machine specifications:
  - OS: CentOS 8.1 (kernel version 4.18)
  - Suricata: version 6.0.3-dev
  - CPU: 2x Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz, Hyperthreading enabled, 10 physical cores
  - Memory: 64GB
  - NIC: 2 port Mellanox ConnectX-5 MT27800 card rated for 100 Gbps
  - The whole setup based on 1 NUMA node

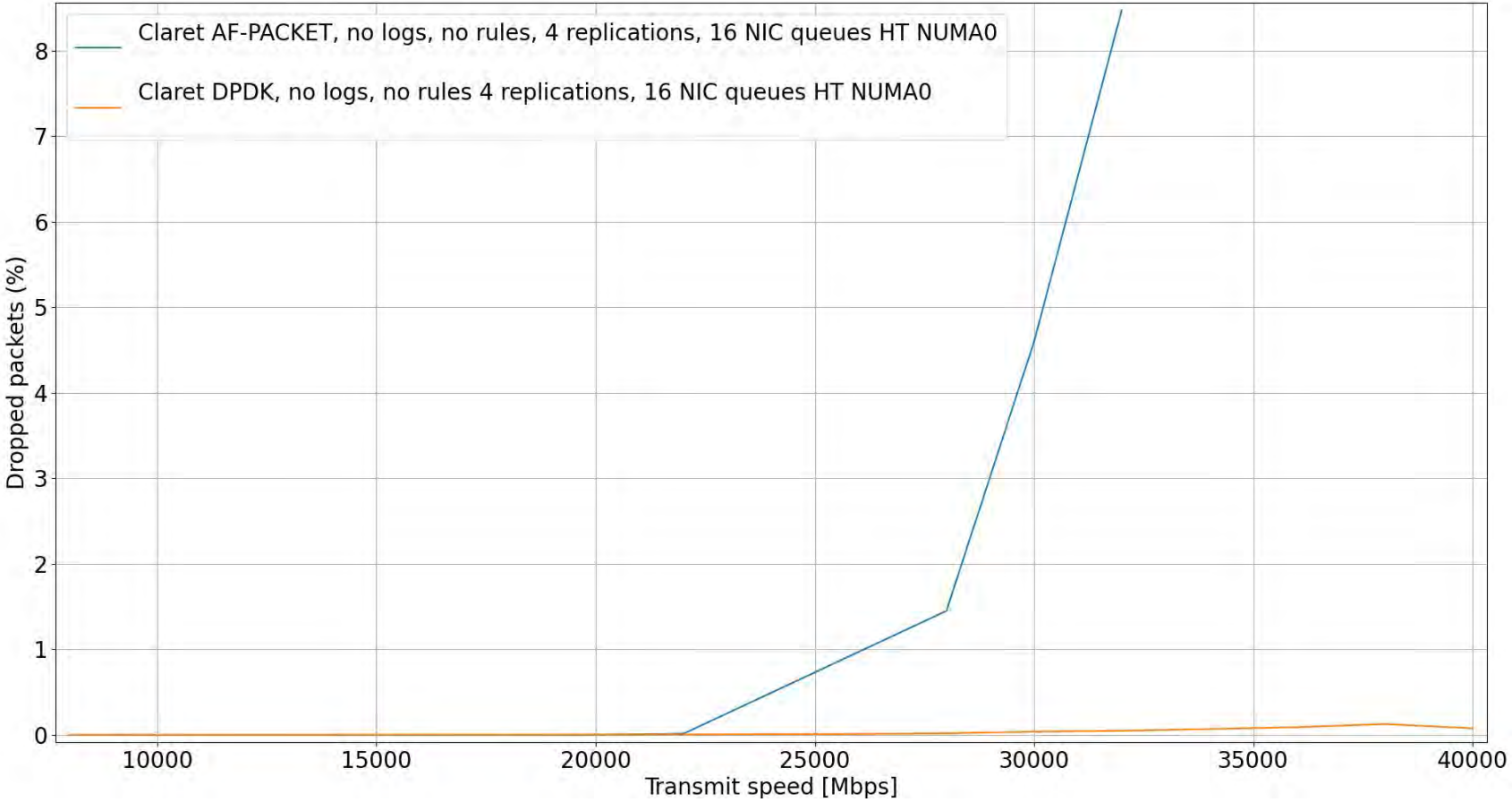
DPDK Testbed



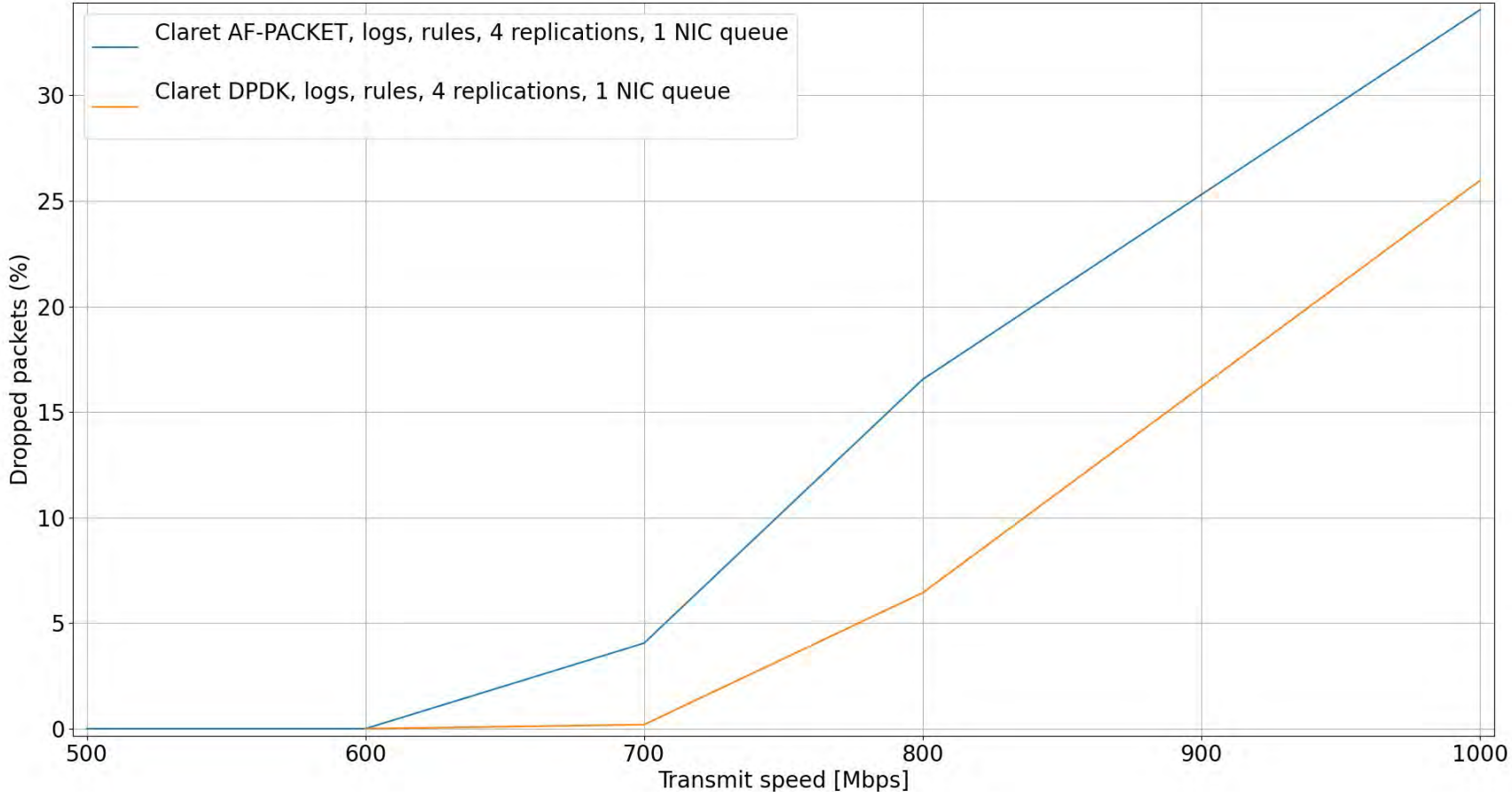
Experiments



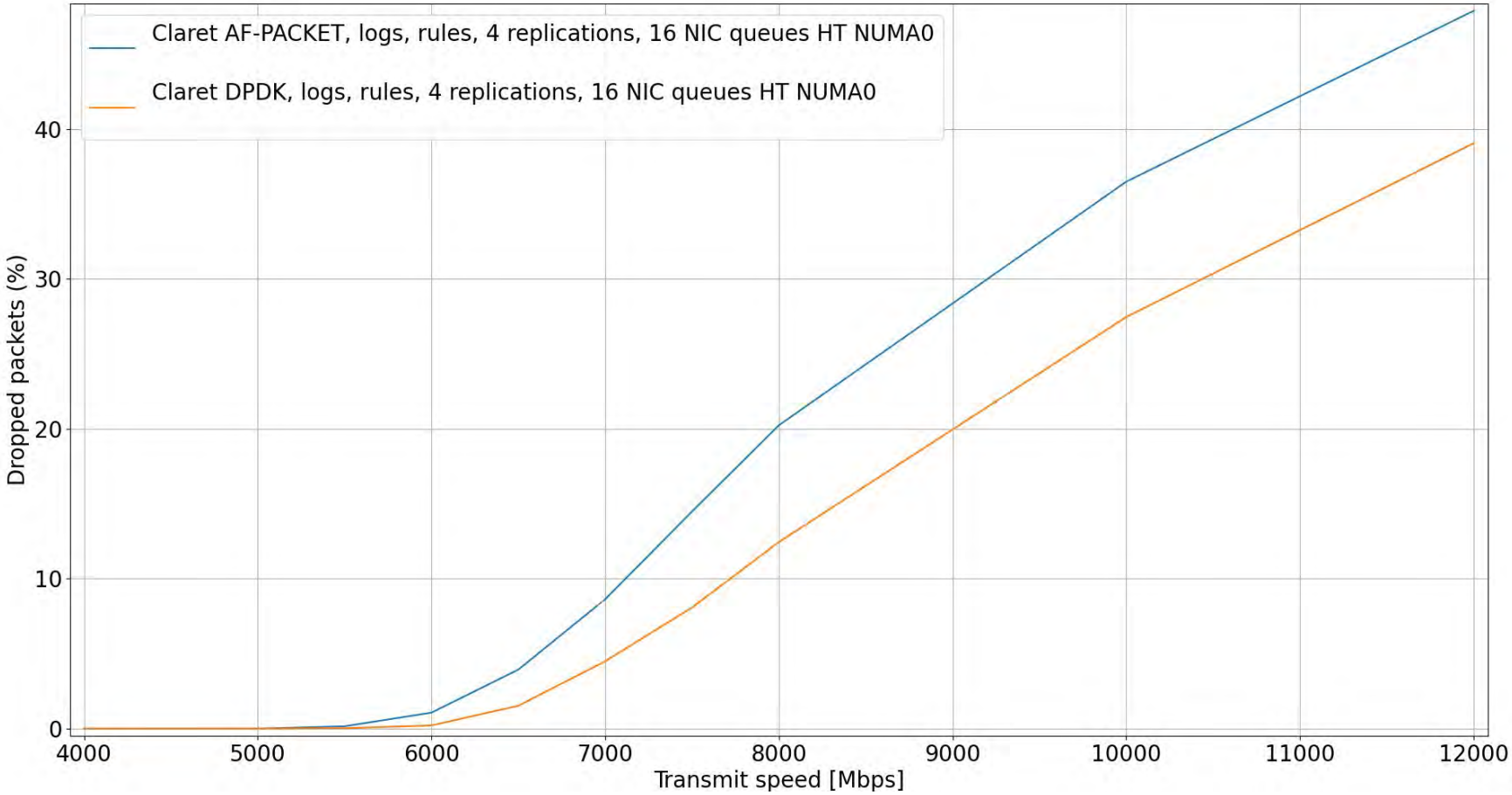
Experiments



Experiments



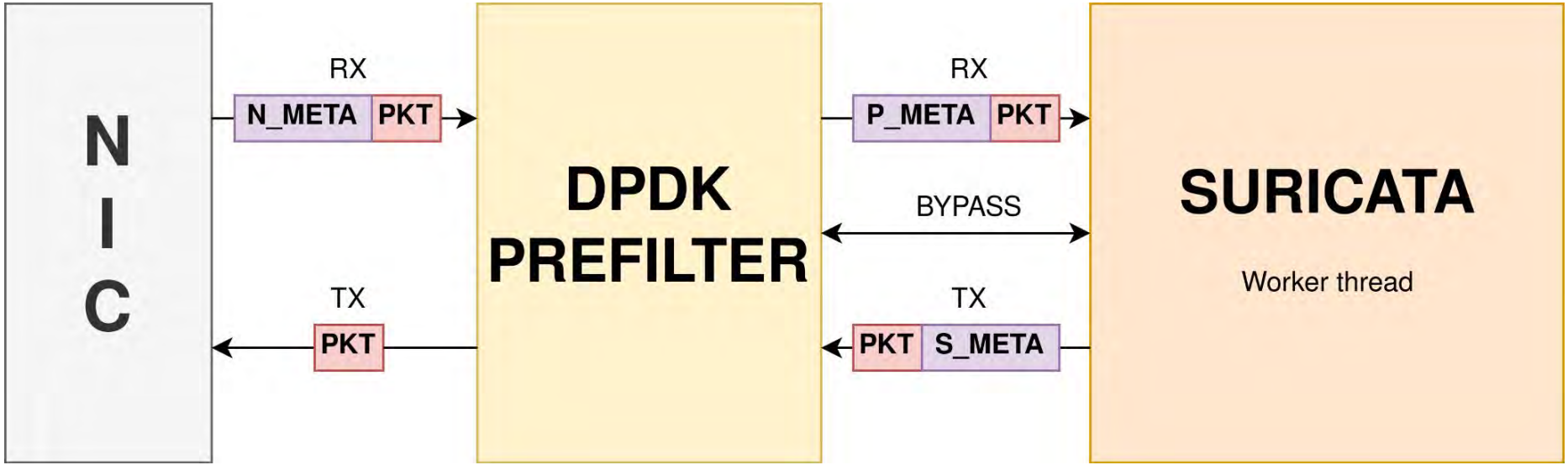
Experiments



DPDK capture method conclusion

- Massive performance increase in regards of packet acquisition modules
- DPDK capture interface provide overall performance benefits for Suricata
- How do we further improve the performance?
  - To analyze more we need to prefilter the traffic

DPDK Prefilters concept



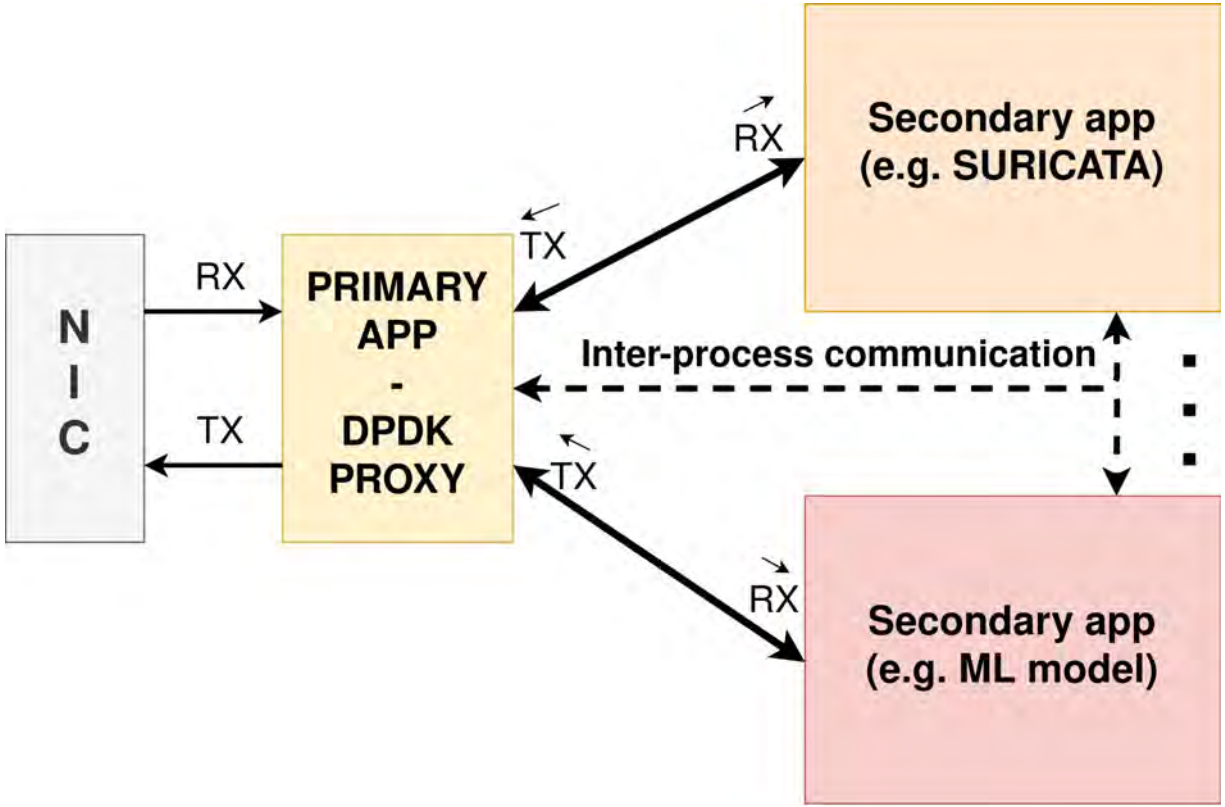
DPDK Prefilters

- Implement bypass functionality
- Divides operation into DPDK primary and secondary process
- Can employ various strategies to redirect flow even before Suricata tells it to
- Strategy can include e.g. encrypted traffic analysis or feed machine learning model with Suricata metadata
- Creates opportunity for quick prototyping of HW functions in SW

DPDK Proxy

- Analyze the same traffic by multiple apps on one server at the same time
- Packets can be shared even between multiple applications
- Allows to split the traffic between multiple apps
- Provide bypass functionality for the subscribed apps

DPDK Proxy concept



Conclusion

- Implemented new packet capture interface based on DPDK
  - Performance doubled with no rules enabled
  - Performance increased ~16% with ET Open enabled
  - Opportunity for new possibilities within DPDK libraries
- DPDK Prefilter
  - Designed for an advanced prefiltering of packets
  - Fast HW prototyping
- DPDK Proxy
  - Provides possibility to inspect the traffic by multiple apps at the same time on one server