

Using (Suricata over) PF_RING for NIC-Independent Acceleration

Luca Deri <deri@ntop.org>

Alfredo Cardigliano <cardigliano@ntop.org>

Outlook

- About ntop.
- Introduction to PF_RING.
- Integrating PF_RING with Suricata.
- Using PF_RING in real-life scenarios.

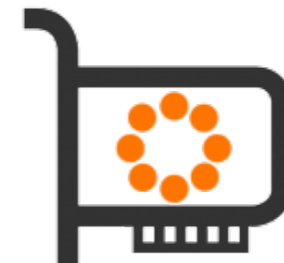
About ntop

- ntop develops open source network traffic monitoring applications.
- ntop (circa 1998) is the first app we released and it is a web-based network monitoring application.
- Today our products range from traffic monitoring, high-speed packet processing, deep-packet inspection, and IDS/IPS acceleration (Suricata, Snort, Bro).

ntop's Approach to Traffic Processing

- Ability to capture, process and (optionally) transmit traffic at line rate, any packet size.
- Leverage on modern multi-core/NUMA architectures in order to promote scalability.
- Use commodity hardware for producing affordable, long-living (no vendor lock), scalable (use new hardware by the time it is becoming available) monitoring solutions.
- Use open source to spread the software, and let the community test it on unchartered places.

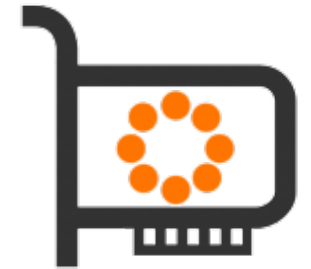
PF_RING



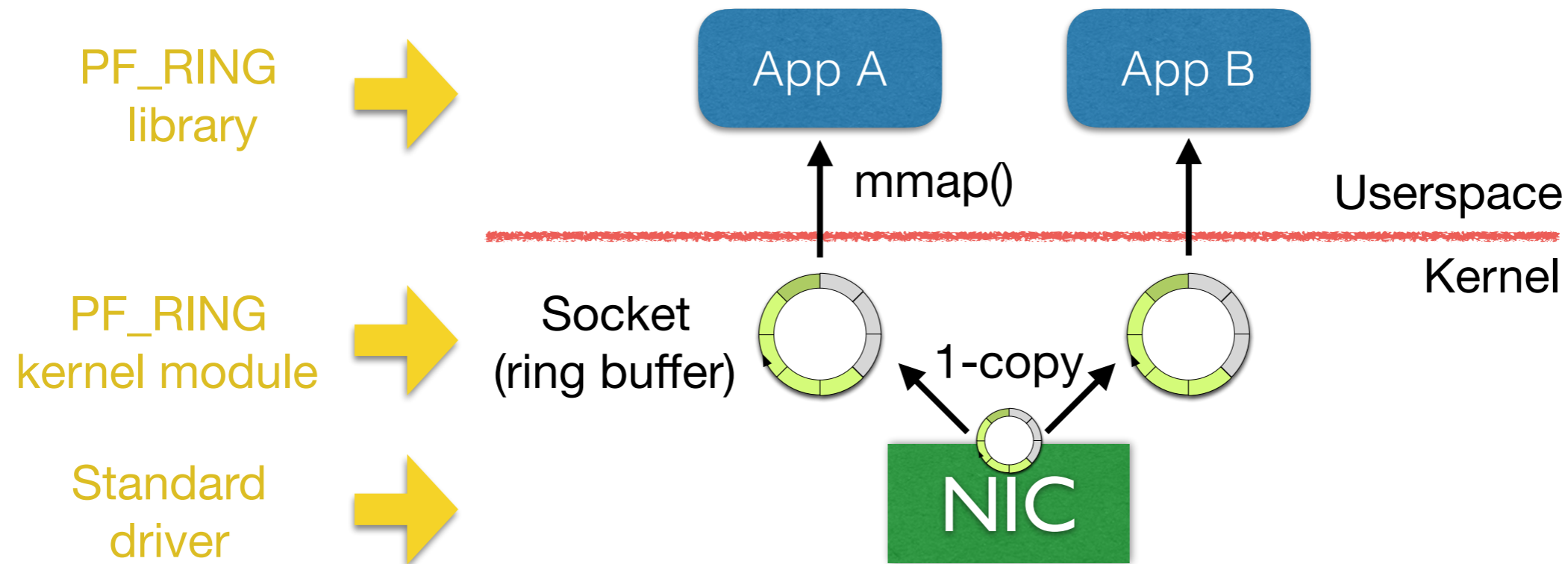
PF_RING History

- PF_RING is a home-grown open source packet processing framework for Linux.
- When it was born (2003), PF_RING was designed to accelerate packet capture on commodity hardware without using FPGA-based network adapters.
- Later (2005/9) PF_RING introduced DNA (Direct NIC Access) featuring kernel bypass for line rate RX/TX packet processing.
- In PF_RING ZC (2013) we have focused on efficient packet processing on multi-vendor NICs, as line rate was already granted by PF_RING.

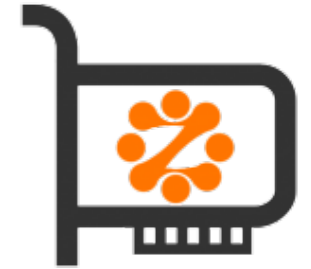
PF_RING Flavours [1/2]



- Standard, in-kernel packet capture (1-copy mode, no kernel bypass): packets are received and processed inside the Linux kernel, then are dispatched using memory-map to user-space. This operating mode is good for IG interfaces.



PF_RING Flavours [2/2]



- PF_RING ZC (zero-copy, kernel bypass): once the device is open, packets are ready directly by user-space applications without passing through the kernel. This is the preferred solution for traffic over 1 Gbit.

PF_RING library
(userspace driver)



App

Userspace

Kernel

Kernel Bypass

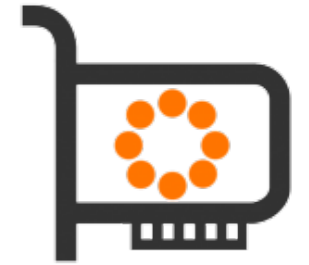
0-copy



ZC
drivers

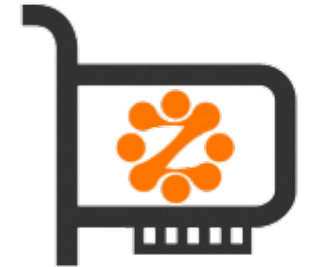


PF_RING API



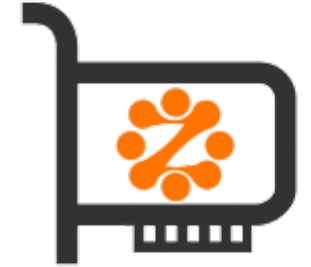
- A single API, regardless of:
 - Interface vendor and speed.
 - ZC and non-ZC.
- If you code your application on top of the PF_RING API, your development investment will be preserved as you won't need to change a single line of code when moving from one adapter to another (e.g. prototype on a cheap on-board ethernet and then deploy 40 Gbit interface: just change the device name).


PF_RING ZC [1/2]



- The idea behind ZC is to create a playground for processing information (and in particular network packets) in zero-copy.
- ZC comes with 0-copy user-space drivers (for 1 and 10G Intel NICs) that allow packets to be read in 0-copy.
- 1-copy packets (e.g. received on non-Intel NICs or WiFi/Bluetooth devices) can be injected in ZC and from that time onwards, be used in 0-copy.

PF_RING ZC [2/2]



- Support of legacy pcap-based applications.
- ZC has simple yet powerful components (no complex patterns, queue/consumer/balancer).
- KVM support: ability to setup Intra-VM clustering.
-  Support
- Native PF_RING ZC support in many open-source applications such as Snort, Suricata, Bro, Wireshark.
- Ability to operate on top of sysdig.org for dispatching system events to PF_RING applications.

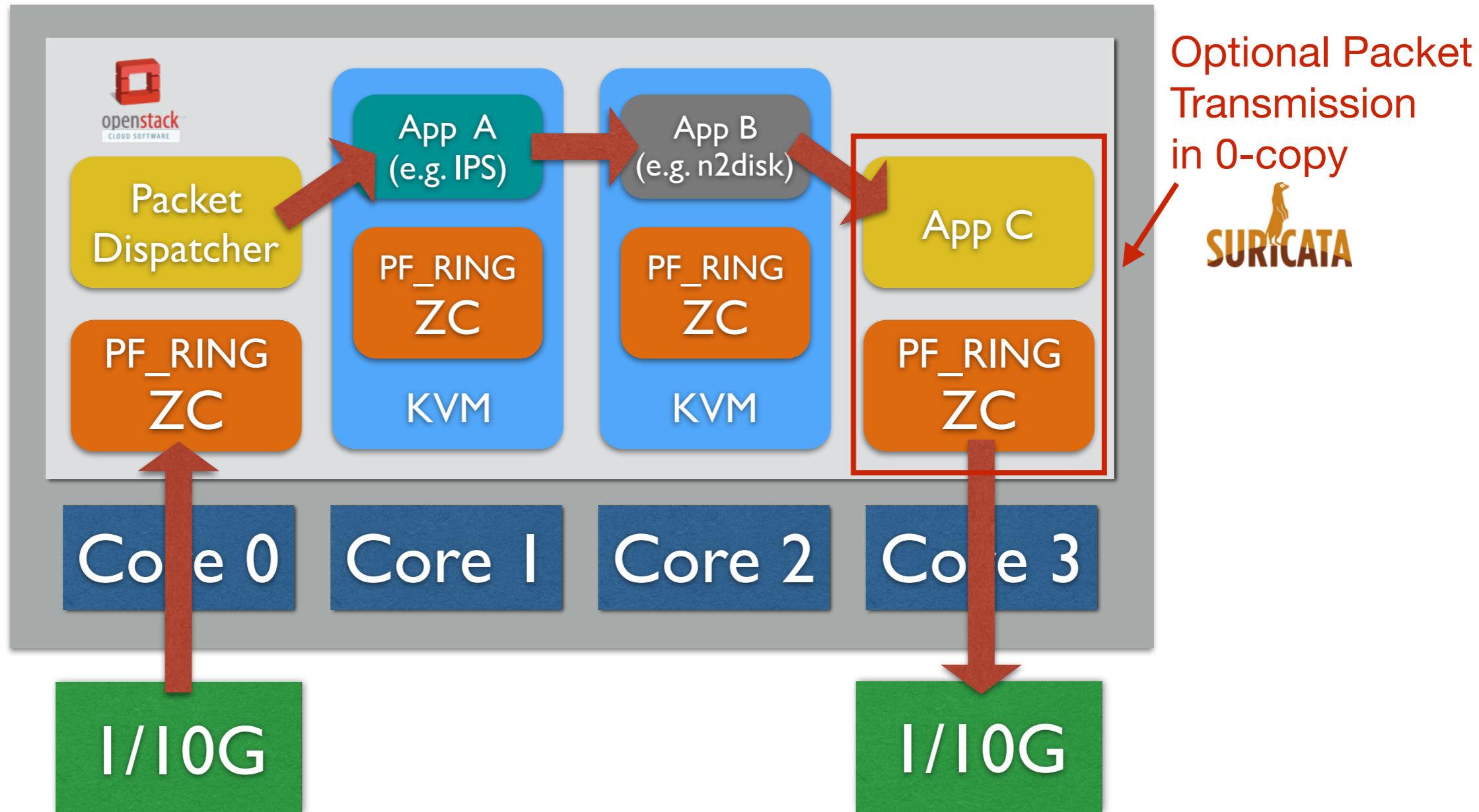
PF_RING ZC on KVM

- With ZC, packets are captured in 0-copy from network adapters and deployed in 0-copy to VMs.
- ZC packets are deployed on the VM using virtual adapters dynamically attached to the VM through PCI hot-plug.
- When an application running inside the VM wants to open a ZC queue, via this virtual adapter, the application is attached in 0-copy with the packet producer.

PF_RING ZC and OpenStack [1/2]

- Most companies operating in OpenStack, focus on Open vSwitch acceleration.
- In ntop we have privileged instead the ability to:
 - Deliver packets in 0-copy at line rate to VMs.
 - Create arbitrary packet processing topologies in VMs, processes and threads.
- The idea is to create a framework that is able to run applications on a VM at the same speed.

PF_RING ZC and OpenStack [2/2]



```
(Host) $ ./zpipeline_ipc -i zc:eth2,0 -o zc:eth3,1 -n 2 -c 99 -r 1 -t 2 -Q /tmp/qmp0
(VM)   $ ./zbounce_ipc -c 99 -i 0 -o 1 -g 3
```

Multi-Vendor Support in PF_RING [1/2]

- The PF_RING user-space library is logically divided into modules, to support multivendor NICs.
- PF_RING-based applications transparently select the correct module name, by means of the interface name to use. Example:
 - `pfcount -i eth1` [Vanilla Linux adapter]
 - `pfcount -i zc:eth1` [Intel ZC drivers]
 - `pfcount -i anic:1` [Accolade Technology]

Multi-Vendor Support in PF_RING [2/2]

- Currently PF_RING natively supports the following vendors (1/10/40/100 Gbit)



* Beta version.

What Vendor Shall I Choose?

- ntop is an independent software company that is not endorsing any vendor.
- Users have to decide based on their requirements, budget, and preferences.
- Most modern adapters can now support 10G line-rate RX/TX and hardware packet time-stamping, so you should choose based on other features such as in-hardware port aggregation, packet filtering/balancing, memory buffers on the NIC (to adsorb traffic spikes), pattern matching...

ASIC vs FPGA-based NICs [1/3]

- Most commodity NICs use an ASIC chip to implement networking: no programmability, simple RX/TX operations, cheap.
- As commodity NICs are used for basic networking they work per-packet so you have PCIe transactions and interrupts as packet rate increases.
- Packet memory is allocated per-packet in the host and its pointer passed to the NIC where it will place incoming RX packets.

ASIC vs FPGA-based NICs [2/3]

- In FPGA-based NICs, packets are moved to the host in blocks (1 MB or more) instead of per-packet and thus the pressure on the PCIe bus and memory subsystem is greatly reduced.
- Packet memory is managed by the FPGA that passes packets in zero-copy to user-space applications.

ASIC vs FPGA-based NICs [3/3]

- (+) Moving packets in blocks reduces the load on the system and thus frees CPU cycles for other tasks.
- (-) FPGA memory management limitations: Packets must be processed in order, and if not possible be copied (say goodbye to zero-copy).
- Bottom line: FPGA-based adapters are great if you use them the way the manufacturer has designed them, otherwise you will jeopardise the advantages (both in complexity and memory copies) you have paid when purchasing the NIC.

PF_RING Advantages for NIC Vendors

- If you provide PF_RING support for your NIC, you will immediately be able to run ntop, Suricata, Snort, Bro, Argus, Wireshark... users take advantage of your NIC without you individually supporting all these applications.
- You can provide a simple way to benchmark your NIC and show its advantages with respect to competitors using a single API.

PF_RING Advantages for Developers

Develop on top of PF_RING so:

- Your time investment will be preserved overtime as ntop will make sure that future vendor changes will be supported transparently by PF_RING.
- No need to learn yet another API, and sign NDAs.
- PF_RING will take care of all differences between NICs such as hardware timestamp format etc.
- PF_RING does not add any overhead (beside API wrapping) with respect to the native vendor API, but it offers you many advantages in terms of traffic processing facilities.

Accelerating Suricata with PF_RING

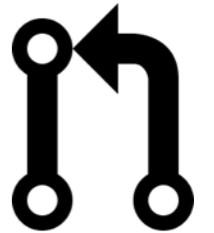
PF_RING Support In Suricata

- Maintained by William Metcalf and Eric Leblond

```
/**  
 * \file  
 *  
 * \author William Metcalf <william.metcalf@gmail.com>  
 * \author Eric Leblond <eric@regit.org>  
 *  
 * PF_RING packet acquisition support  
 *
```

- ntop is contributing for improving packet acquisition based on latest PF_RING technologies

Latest Contributions to Suricata [1/2]



- PF_RING IPS/TAP Support [#1587](#)
- new PktAcqBreakLoop callback in TmModule [#1696](#)
- workers runmode: allow multiple input devices [#1701](#)
- pfring pkt acq: keep running on 'pfring_set_cluster' failure when cluster is not required [#1713](#)
- pfring pkt acq: use zero-copy recv in workers runmode [#1706](#)
- pfring pkt acq: removed reentrant flag [#1707](#)
- pfring pkt acq: capture loop optimisation [#1708](#)

} Enabling support for non-Intel/ZC kernel-bypass

} Performance improvements

Latest Contributions to Suricata [2/2]

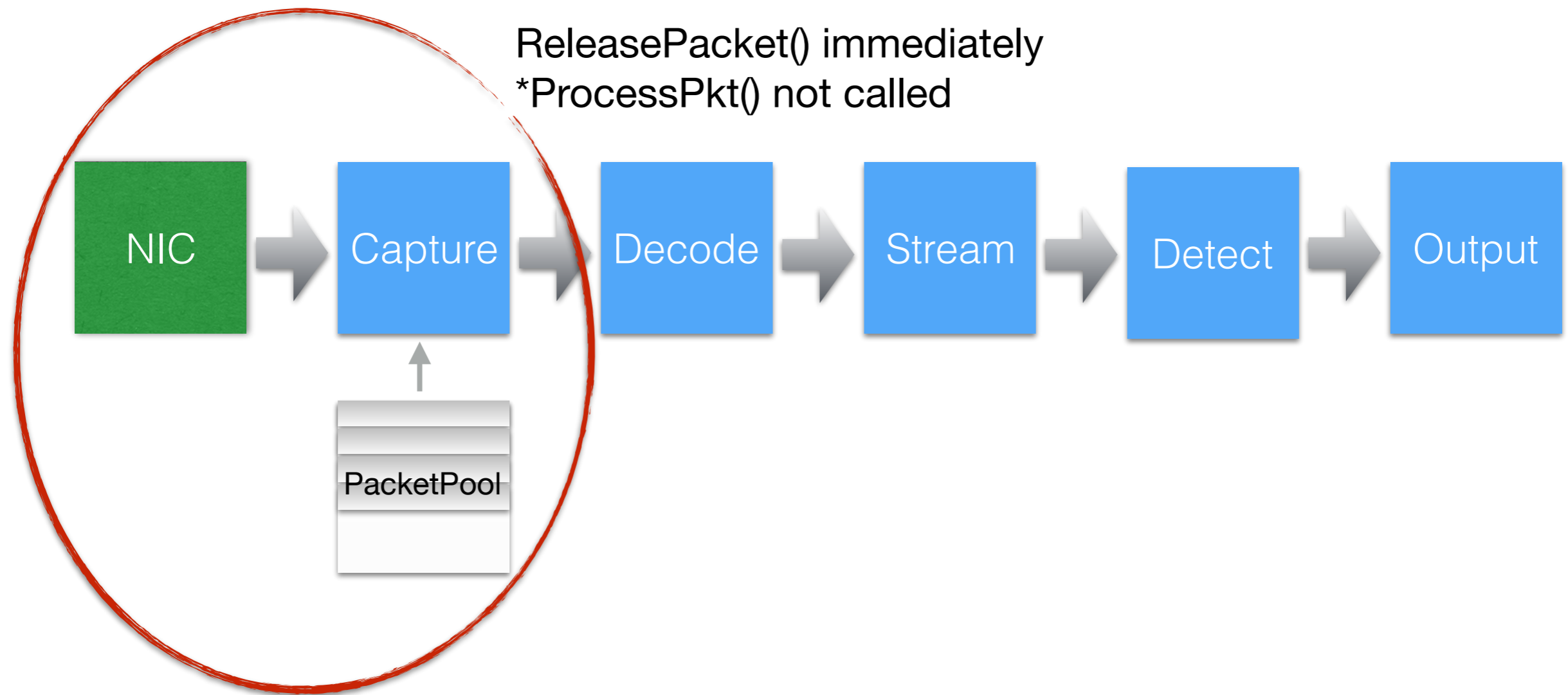
Enabling support for non-Intel kernel-bypass

- Ability to run on top of FPGA-based kernel-bypass cards:



Performance Improvements [1/2]

Packet acquisition performance tests (no processing)



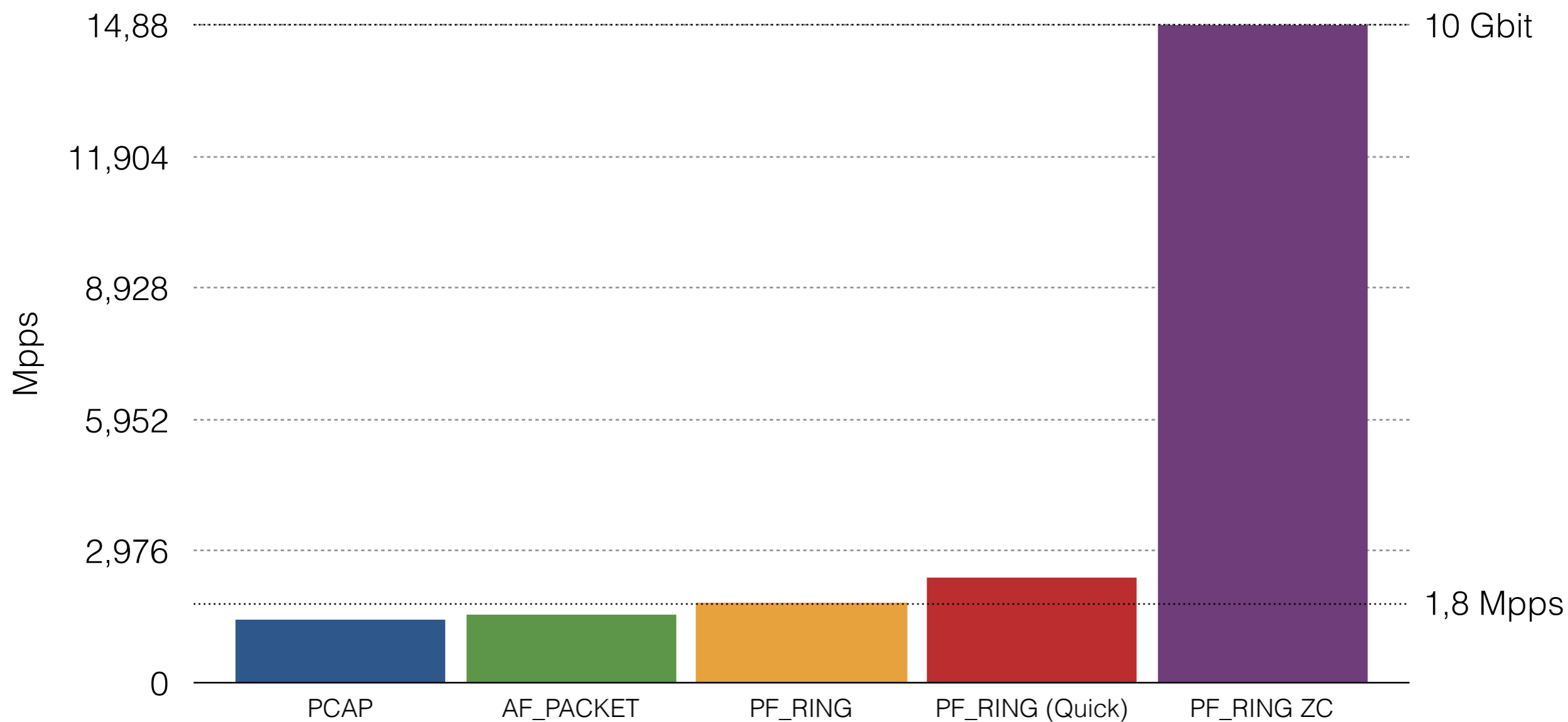
Performance Improvements [2/2]

Capture speed (no processing) with PF_RING ZC (Intel 82599) using a single thread on E3-1230 v3 at 10 Gbit line-rate:

- Before patches: 9.52 Mpps
- After patches: 14.88 Mpps (line-rate)
- **>55%** performance boost (after latest improvements)
- More CPU cycles for real processing !

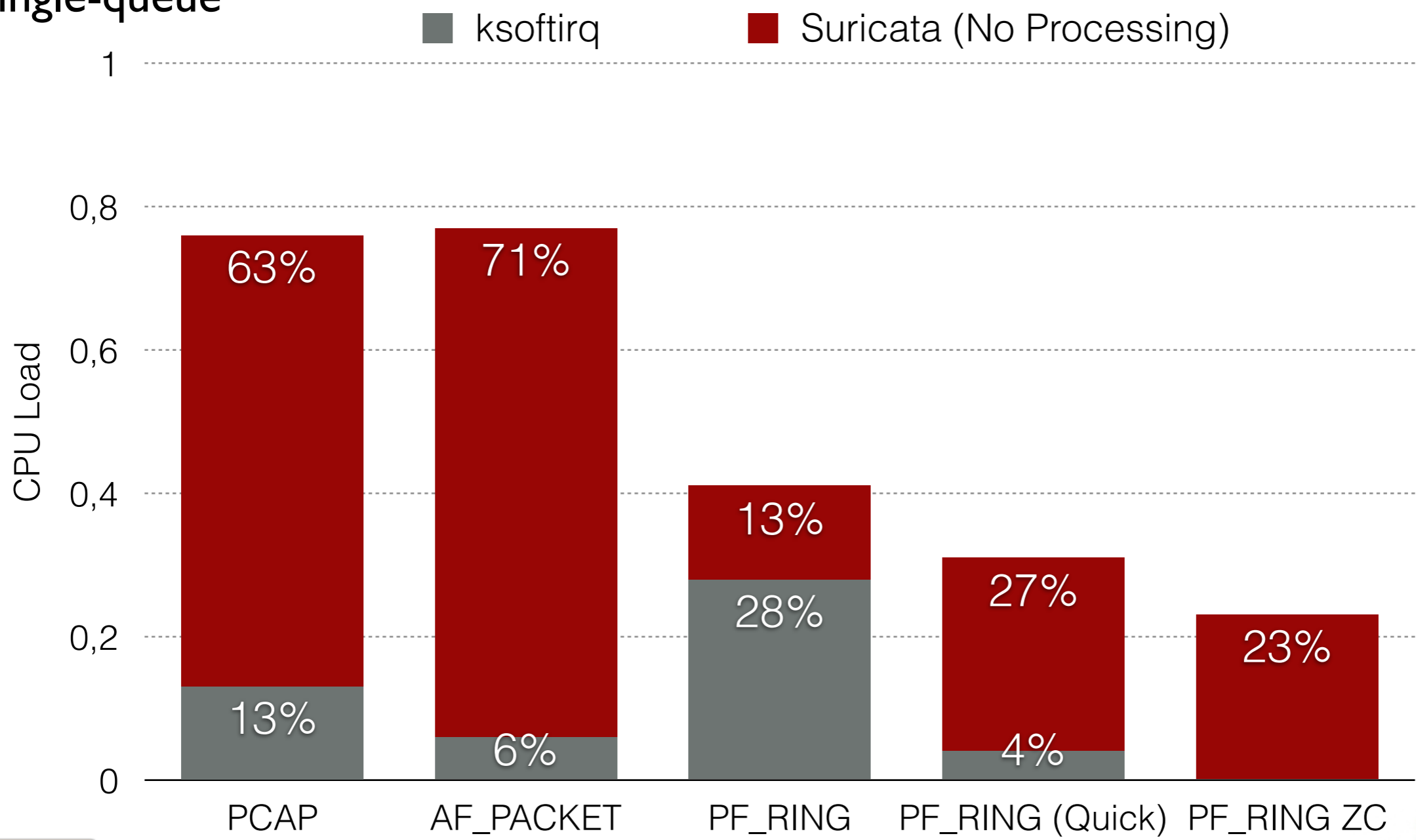
PF_RING/Suricata Performance [1/4]

Packet acquisition performance - single thread - Intel Xeon E3-1230 v3 - Intel 82599 single-queue



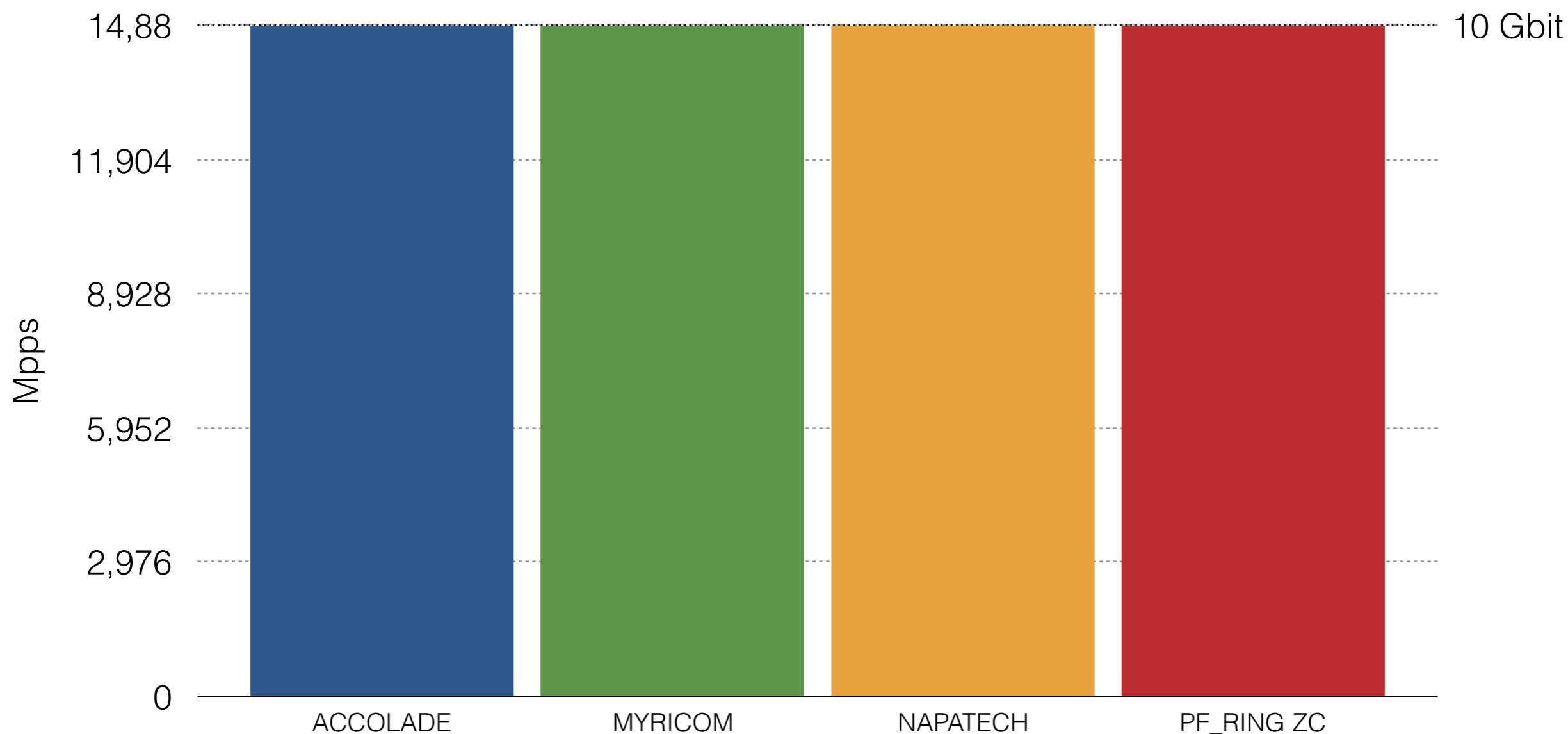
PF_RING/Suricata Performance [2/4]

CPU Load at 1 Mpps - single thread - Intel Xeon E3-1230 v3 - Intel 82599 single-queue



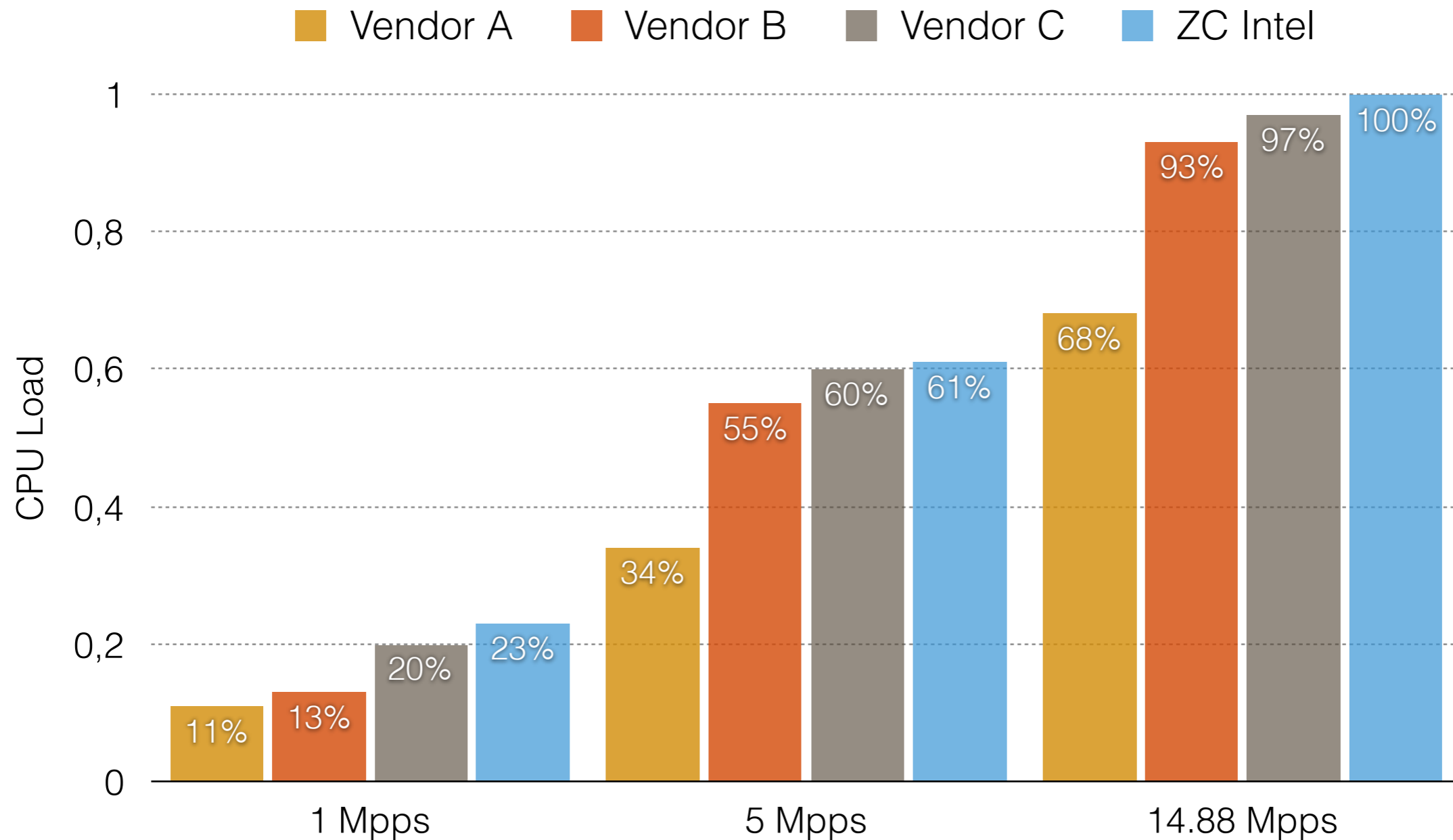
PF_RING/Suricata Performance [3/4]

Packet acquisition performance - Single thread/core
Intel Xeon E3-1230 v3 - kernel-bypass technologies



PF_RING/Suricata Performance [4/4]

CPU Load at 1 Mpps - Single thread/core - Intel Xeon E3-1230 v3



PF_RING in Real Life Scenarios

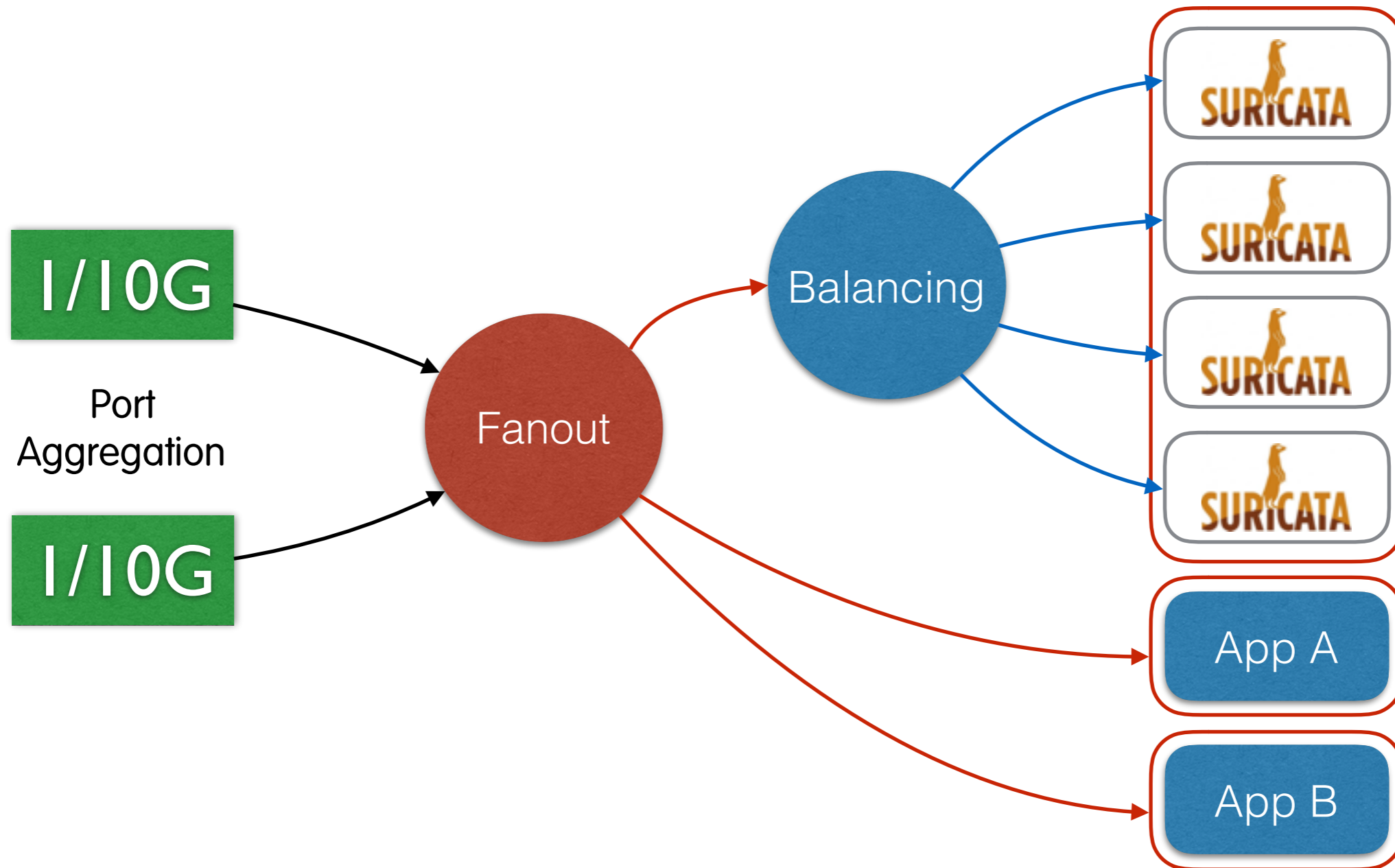
Is Packet Capture Acceleration Still a Good Argument? [1/2]

- For years the industry and community focused mainly on packet capture/filtering.
- Applications were relative simple and self-contained: network security, traffic monitoring, high-frequency trading, packet-to-disk....
- With the advent of big-data systems (and not only that) and reduction of data center space, people like to collapse multi-apps on a single box.
- As previously stated, packet processing at 10 Gbit is now possible with commodity hardware.

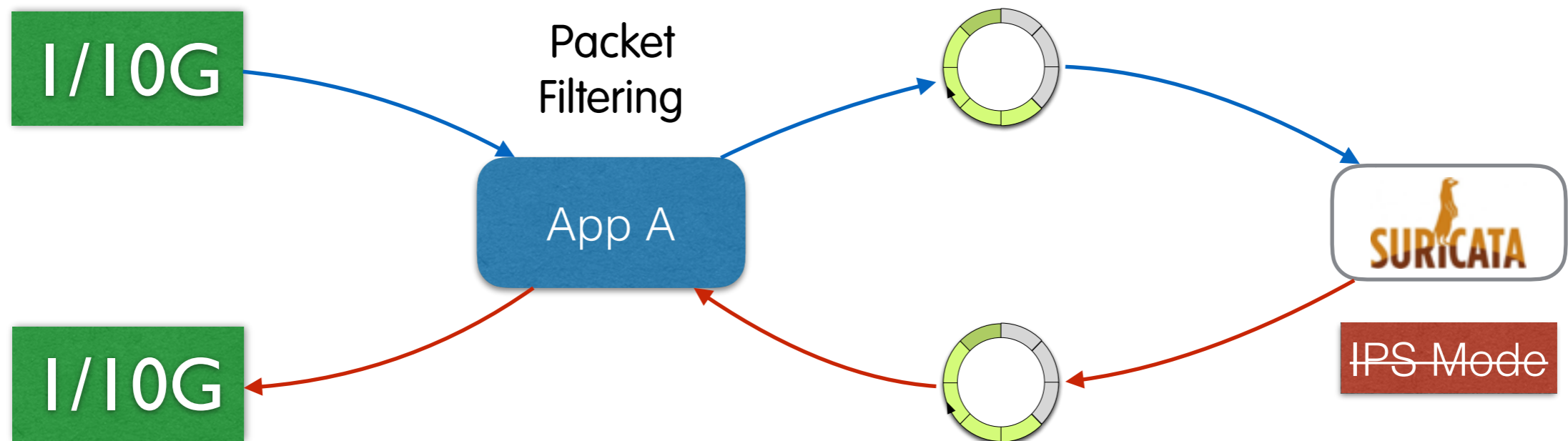
Is Packet Capture Acceleration Still a Good Argument? [2/2]

- 100 Gbit (and partially 40 Gbit) are raising the bar once more, and FPGA-based NICs are currently the winners in this scenario.
- In summary: “packet capture acceleration” is no longer enough as we often need to combine it with multi-app traffic distribution, balancing, and pipelining in order to collapse on one box at high speed, functionalities that were previously implemented onto multiple boxes.

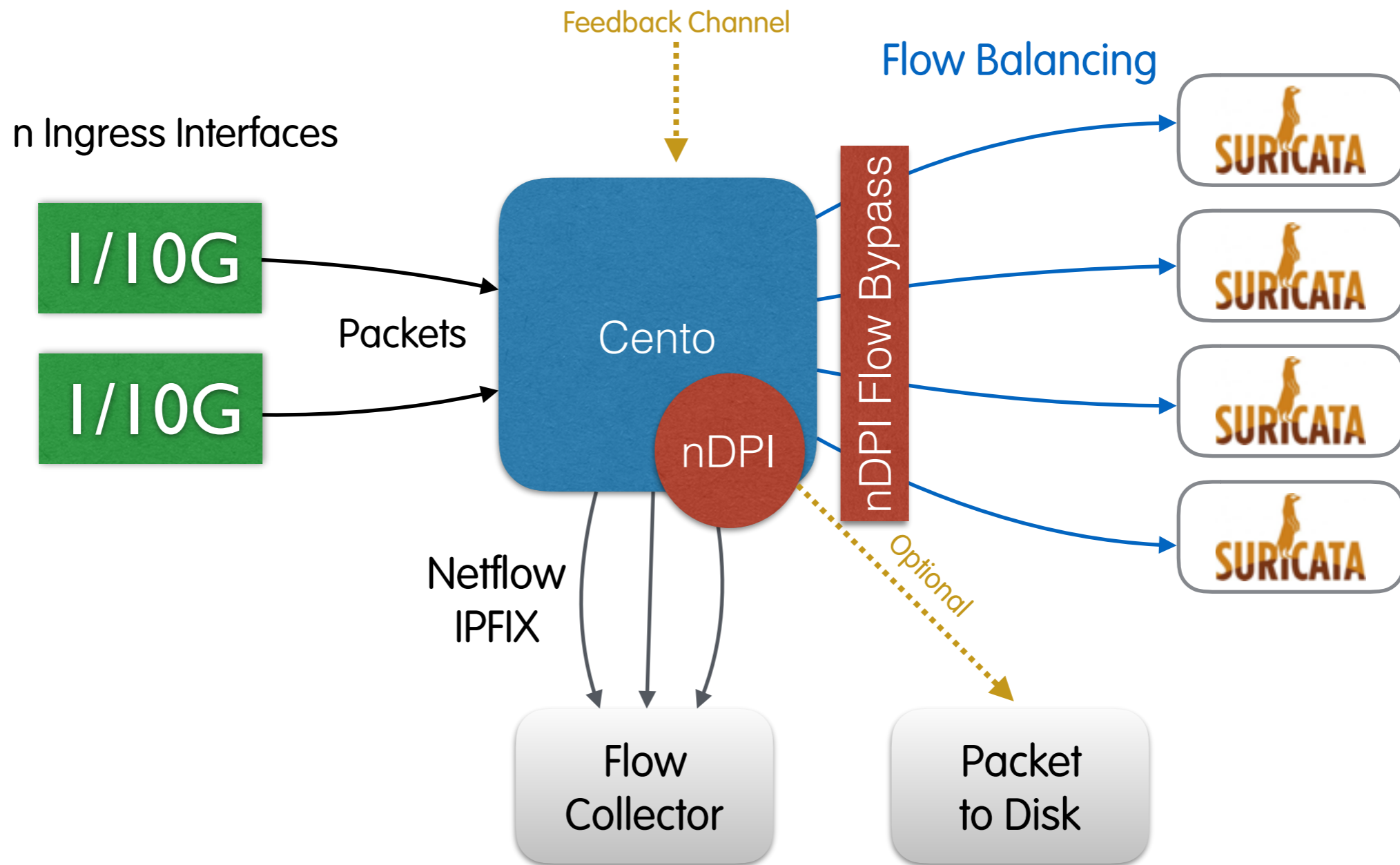
PF_RING: Aggregation + FanOut + Balancing



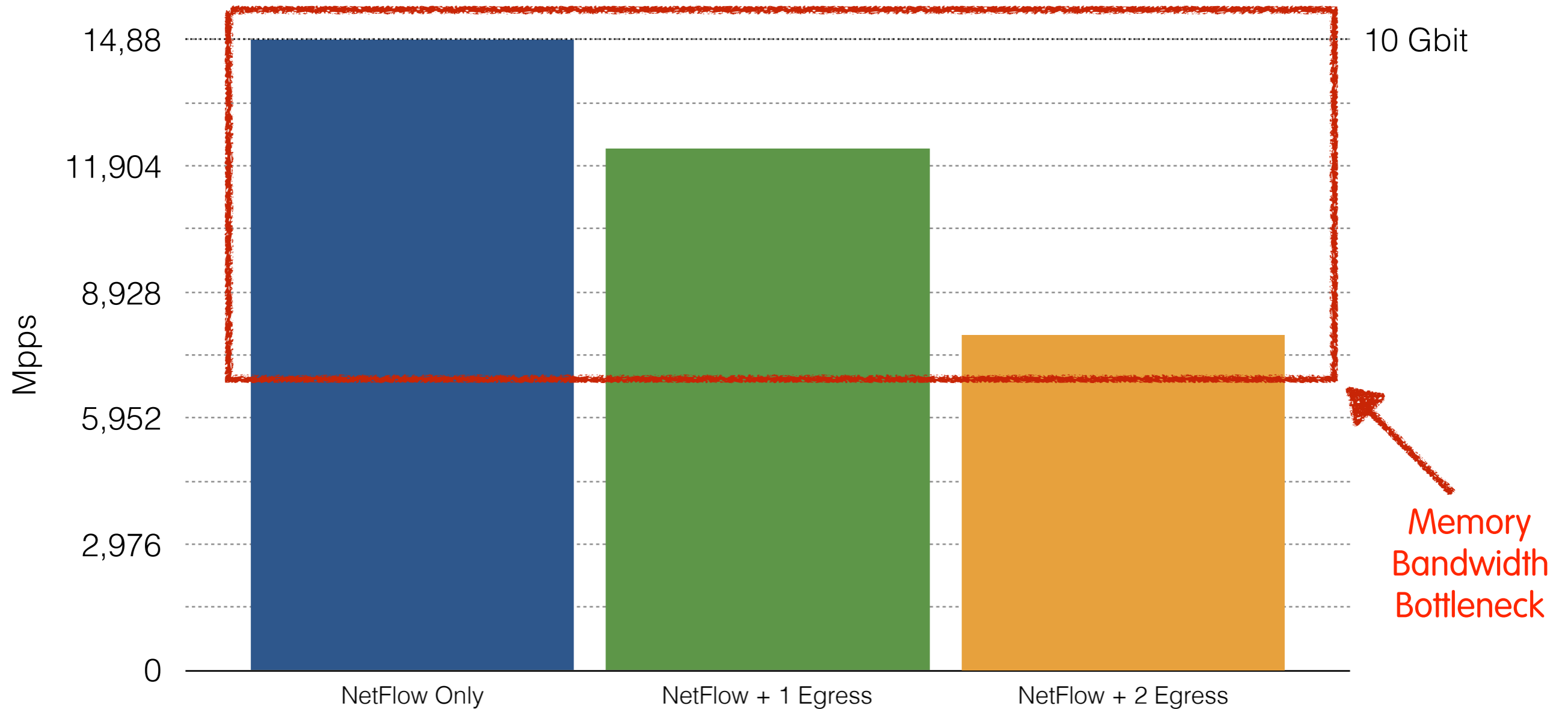
PF_RING: Pipelining + Traffic Cleanup



PF_RING: Flows + DPI + Security [1/2]



PF_RING: Flows + DPI + Security [2/2]



```
cento -i zc:eth6 -9 127.0.0.1:1234 -t 5 -o 2 (500k flows, expiring ever 5 seconds, export in NFv9)  
taskset -c 2,3 suricata --pfring-int=zc:l0@0 --pfring-int=zc:l0@1 -c /etc/suricata/suricata.yaml --runmode=workers
```

Final Remarks

The Big Picture

- PF_RING implements a packet processing framework featuring:
 - Single API regardless of the network adapter being used.
 - Support for commodity and FPGA-based adapters.
 - Native support of many opensource applications as well native Suricata integration (IDS and IPS modes).
 - Developers can focus on a single API and let their users choose the best NIC for their project.
 - Stable and maintained product (since 12 years).
 - Download it at https://github.com/ntop/PF_RING/