14 Years Of PF_RING

How packet capture acceleration evolved
from pf_ring.ko to PF_RING FT

Alfredo Cardigliano
cardigliano@ntop.org

Introduction

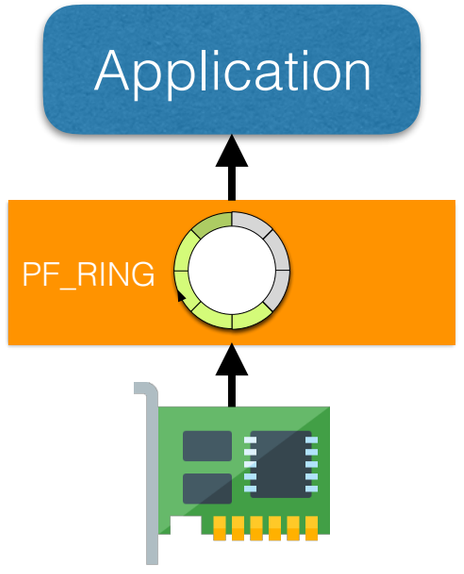
- Network Monitoring tools need high-speed, promiscuous, raw packet capture.

- Specialized adapters are often not affordable, or not flexible enough, or they do not provide an “open” API.

- Commodity network adapters and device drivers are designed for providing host connectivity and are not optimized for high-speed raw packet capture.

PF_RING

- PF_RING has been introduced in 2004 for improving the performance of network monitoring applications, accelerating **packet capture** which was the main bottleneck on commodity hardware at that time.



- Packet capture does not mean just providing a buffer with the packet data, it also means providing a rich set of features for **manipulating**, **filtering**, and **processing** packets at high rates.

- PF_RING offers on commodity hardware (a standard PC) the ability to receive and transmit at wire-rate up to 100 Gbit.

ntop

The Bro Use Case

- Bro IDS is a flexible network analysis framework providing:
  - analyzers for many protocols
  - a scripting language to define monitoring policies
    - application-layer state
    - many other nice features
- CPU bound application due to processor-intensive features
- Let’s see how PF_RING has been used to accelerate Bro throughout its evolution..

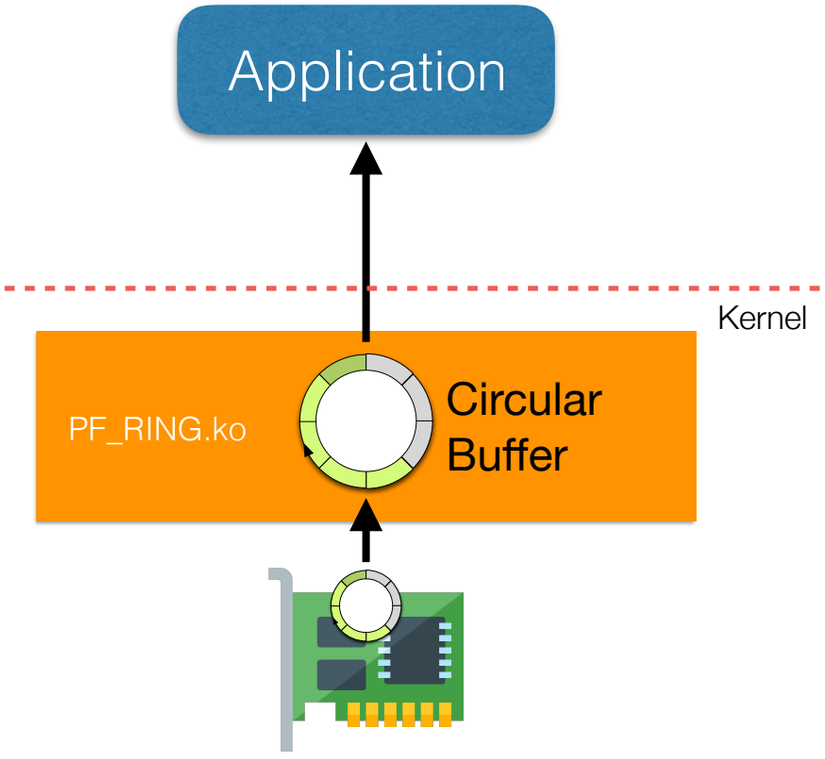
Packet Capture Evolution

ntop

The Kernel Module

- The **pf_ring** module copies packets from the card to a circular buffer.

- The application reads packets directly from the circular buffer.



Capture performance: up to 1-2 Mpps

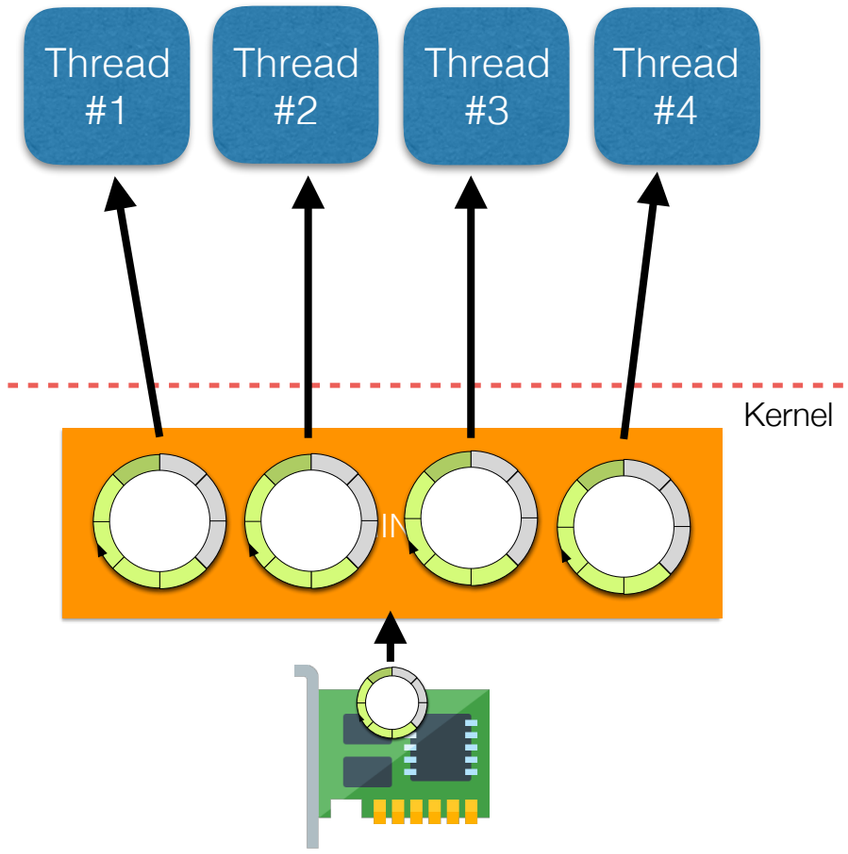
Bro performance: 280 Kpps

pf_ring.ko | Cluster | TNAPI/RSS | DNA | ZC

ntop

Load Balancing

- A **PF_RING Cluster** distributes traffic to multiple threads or application instances.
- It keeps flow coherency (all packets for the same flow to the same thread).
- Each application instance needs to handle a portion of the whole stream.



Capture performance: up to 1-2 Mpps

Bro performance: 1 Mpps on 4 cores

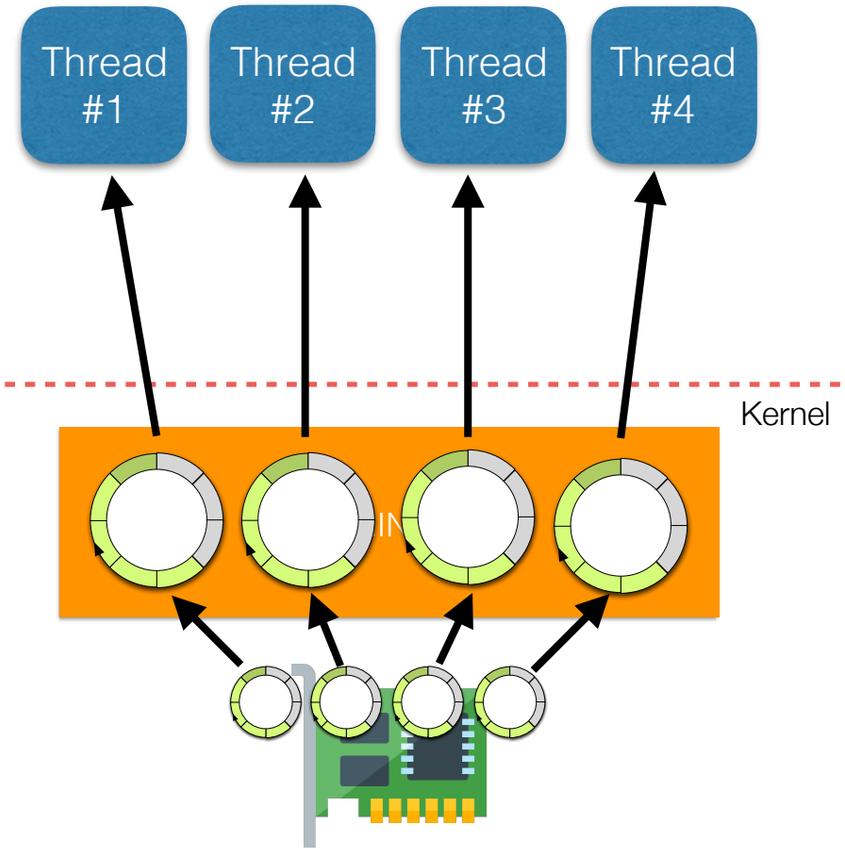
pf_ring.ko | **Cluster** | TNAPI/RSS | DNA | ZC

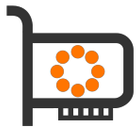
ntop

Hardware Load-Balancing

- **RSS** is a hardware technology that distributes the load across multiple queues keeping flow coherency.

- **TNAPI** (deprecated) was a multi-threaded driver, able to fully exploit RSS and multi-core architectures to deliver data through independent data streams.





Capture Performance: up to 8-10 Mpps on 4 cores

Bro performance: 1 Mpps on 4 cores

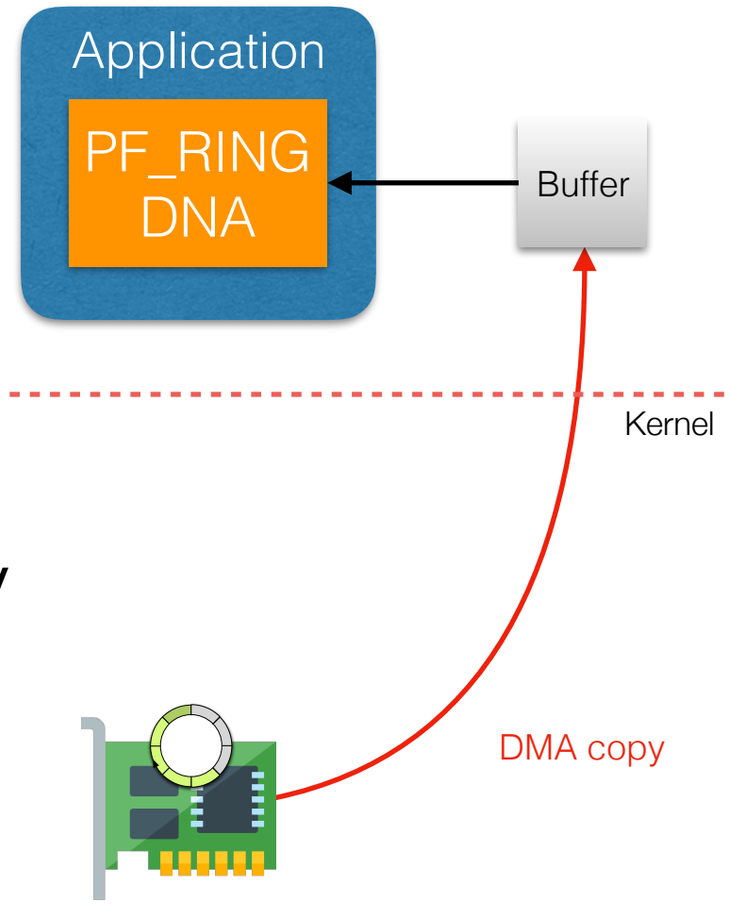
pf_ring.ko → Cluster → **TNAPI/RSS** → DNA → ZC

ntop

Zero-Copy Drivers

- **DNA** (today known as **ZC Driver**) is a kernel-bypass technology for Intel cards.

- Packets get copied by the card directly into the application memory.



Capture Performance: up to 15–20 Mpps/ core

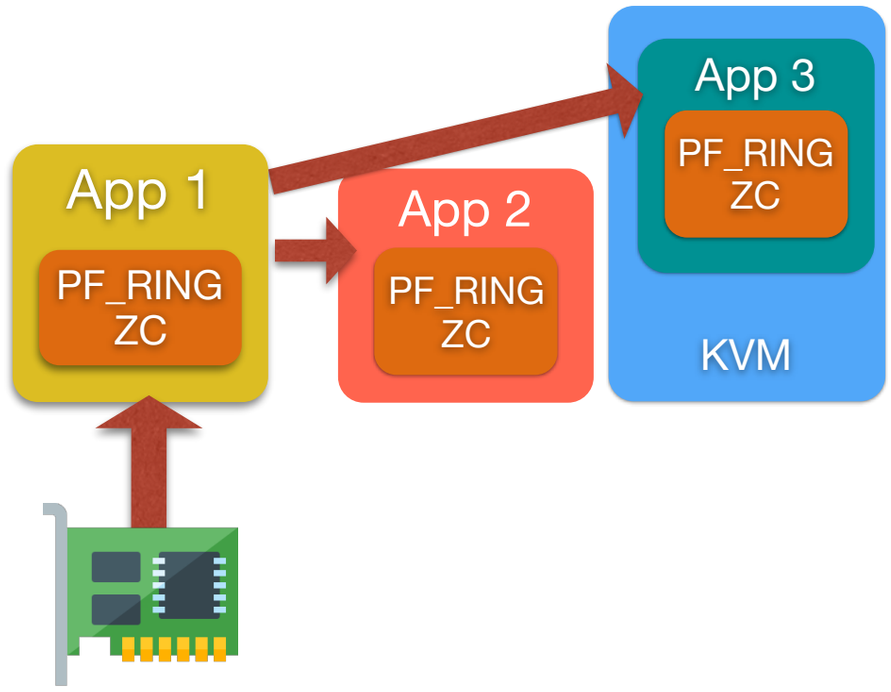
Bro performance: 1 Mpps on 4 cores

pf_ring.ko → Cluster → TNAPI/RSS → **DNA** → ZC

ntop

Zero-Copy API

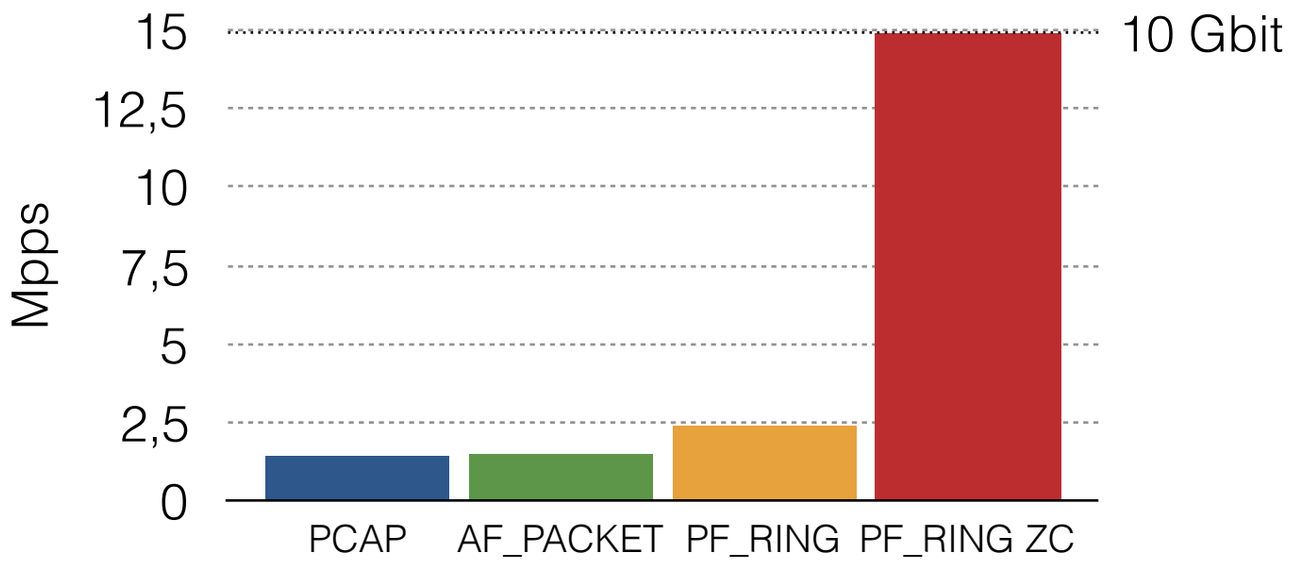
- **PF_RING ZC** provides a flexible API to create full zero-copy processing patterns (load-balancing, pipelining, etc).
- Inter-VM support with KVM.
- Multi-vendor FPGA support.



pf_ring.ko → Cluster → TNAPI/RSS → DNA → **ZC**

Capture Performance vs Application Performance

- Capture speed with Intel cards using 1 core @ 10 Gbit:



- Scale up to 100 Gbps with FPGA adapters using multiple cores.

- Ok, capture speed is impressive, but how to further improve application performance (without touching the code)?

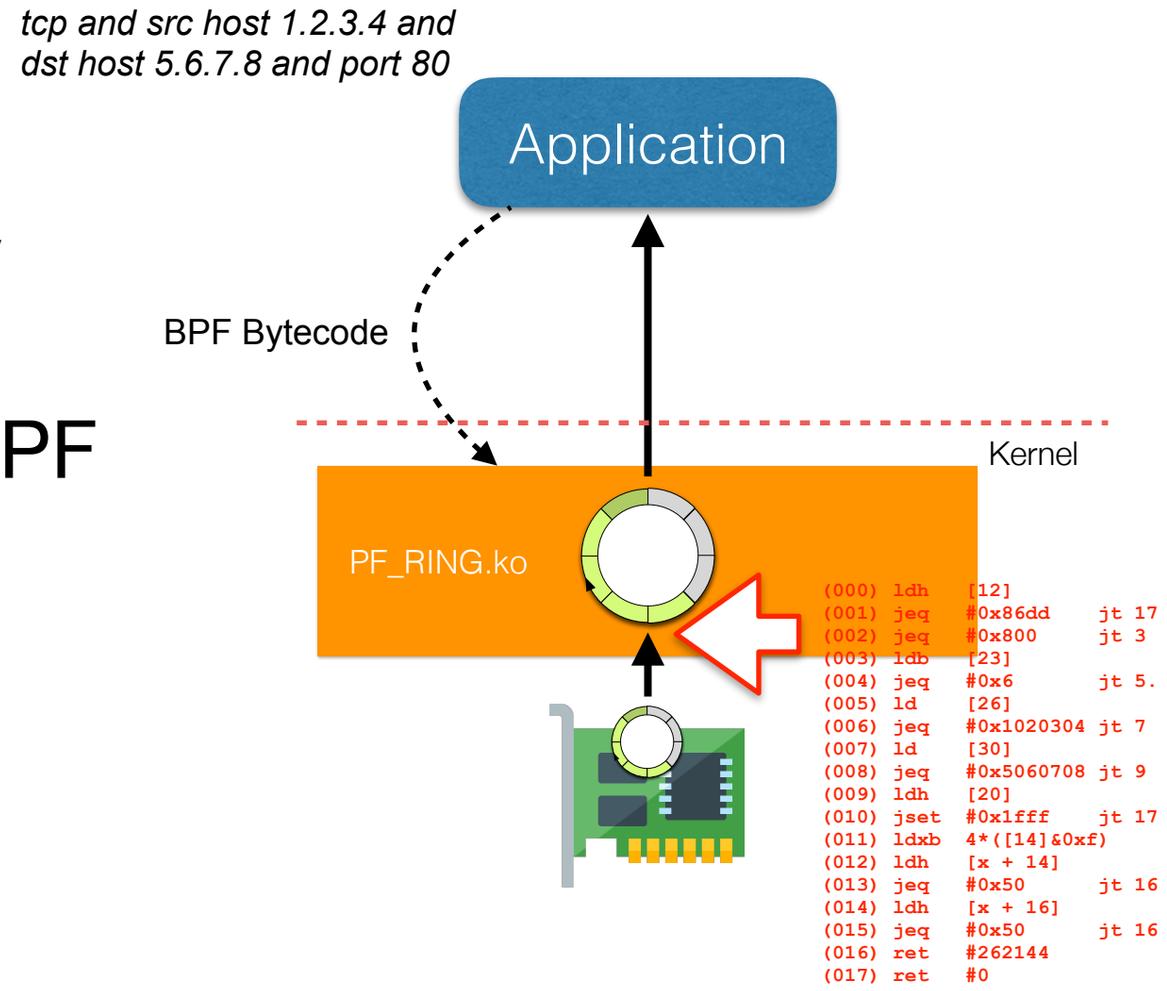
ntop

Packet Filtering Evolution

ntop

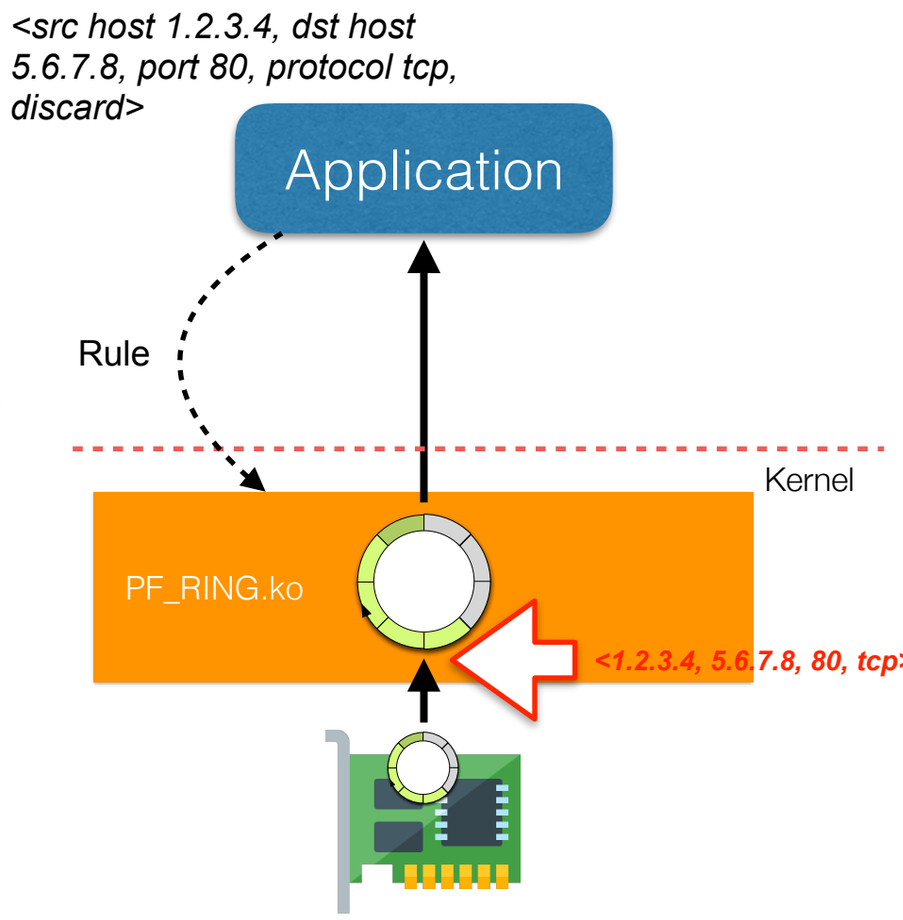
Software Filtering - BPF

- **BPF** filters (those supported by applications like *tcpdump* and *Wireshark),* are compiled into BPF bytecode and executed by the BPF Virtual Machine, for each received packet.



Software Filtering - Rules

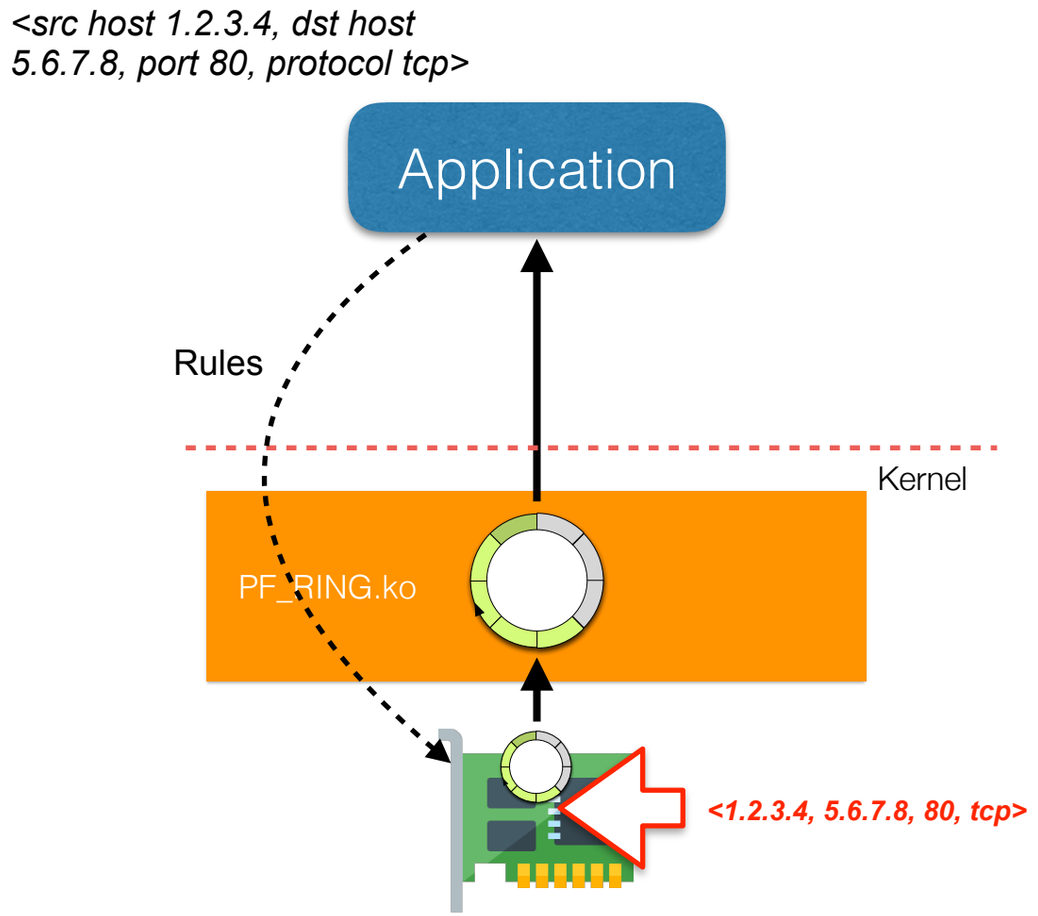
- **Software filtering rules** consist of:
  - Packet elements to match (ip, port, protocol, etc).
  - Action to be performed when a packet matches the filter (pass, discard, forward, etc).



Hardware Filtering Rules

- **Hardware filtering rules** are evaluated by the network card (no CPU overhead).

- Available on Intel (82599/X520), Silicom Redirector, others..

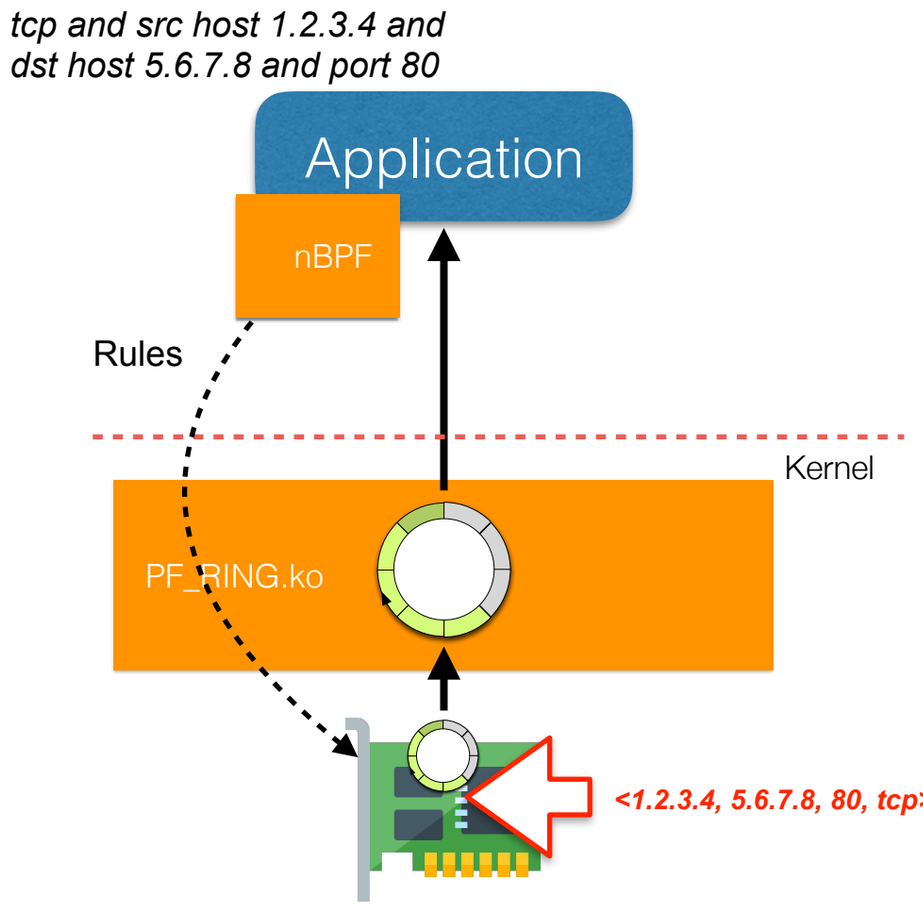


BPF to Filtering Rules

- **nBPF** is a filtering engine supporting the well-known BPF syntax.

- It can generate hardware rules, and filter in software what is not filtered by the card.

- Able to generate hardware rules for FPGA adapters to filter traffic at 100 Gbit.

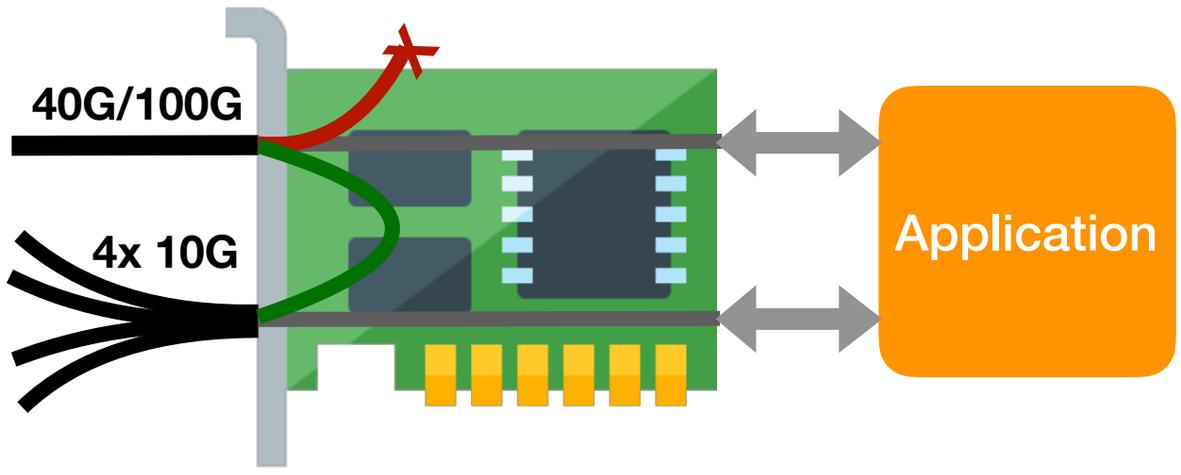


Kernel Filtering | Hardware Filtering | **nBPF** | nBroker | FT/nDPI

ntop

Hardware Traffic Steering

- **nBroker** is a library for controlling traffic filtering and steering on Intel 100 Gbit adapters (FM10000).



- CLI tool to easy the internal switch configuration:

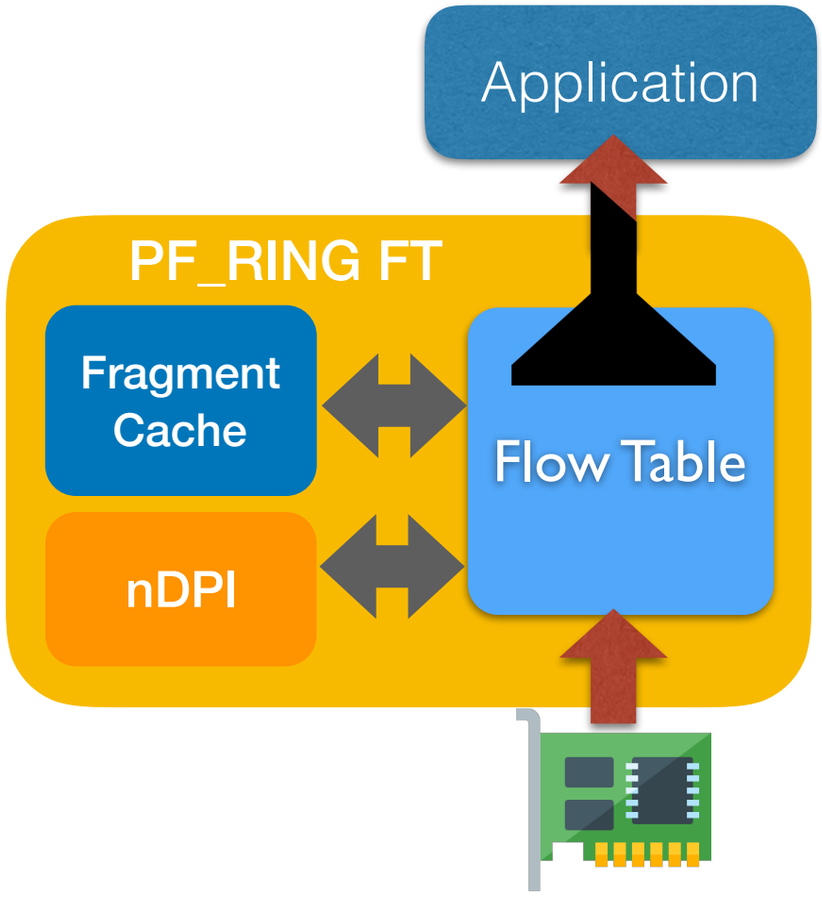
```
> port eth1 match shost 10.0.0.1 dport 80 steer-to eth2
```

ntop

L7 Filtering

- **PF_RING FT (Flow Table)** is a highly optimized library able to classify L7 traffic.

- It leverages on **nDPI** to detect application protocols (250+ protocols including Facebook, Skype, Youtube, BitTorrent, …)



Performance: 10+ Mpps/core

Bro performance (filtering out multimedia traffic*):
1.6 Mpps / 10 Gbit Internet traffic on 4 cores (+60%)

* Multimedia traffic (NetFlix, Spotify, etc) is not really interesting for an IDS..

Kernel Filtering → Hardware Filtering → nBPF → nBroker → FT/nDPI

ntop

Conclusions

- Capture speed and (filtering) features has been improved in PF_RING over the years to assist applications processing high traffic rate.

- Moving to 40 or 100 Gbit accelerating packet capture is not enough, network speed grows faster than CPU speed!

- The best way for improving the performance of CPU bound applications is to scale with cores and nodes, and filter as much as possible.

- Do not confuse capture performance with application performance.