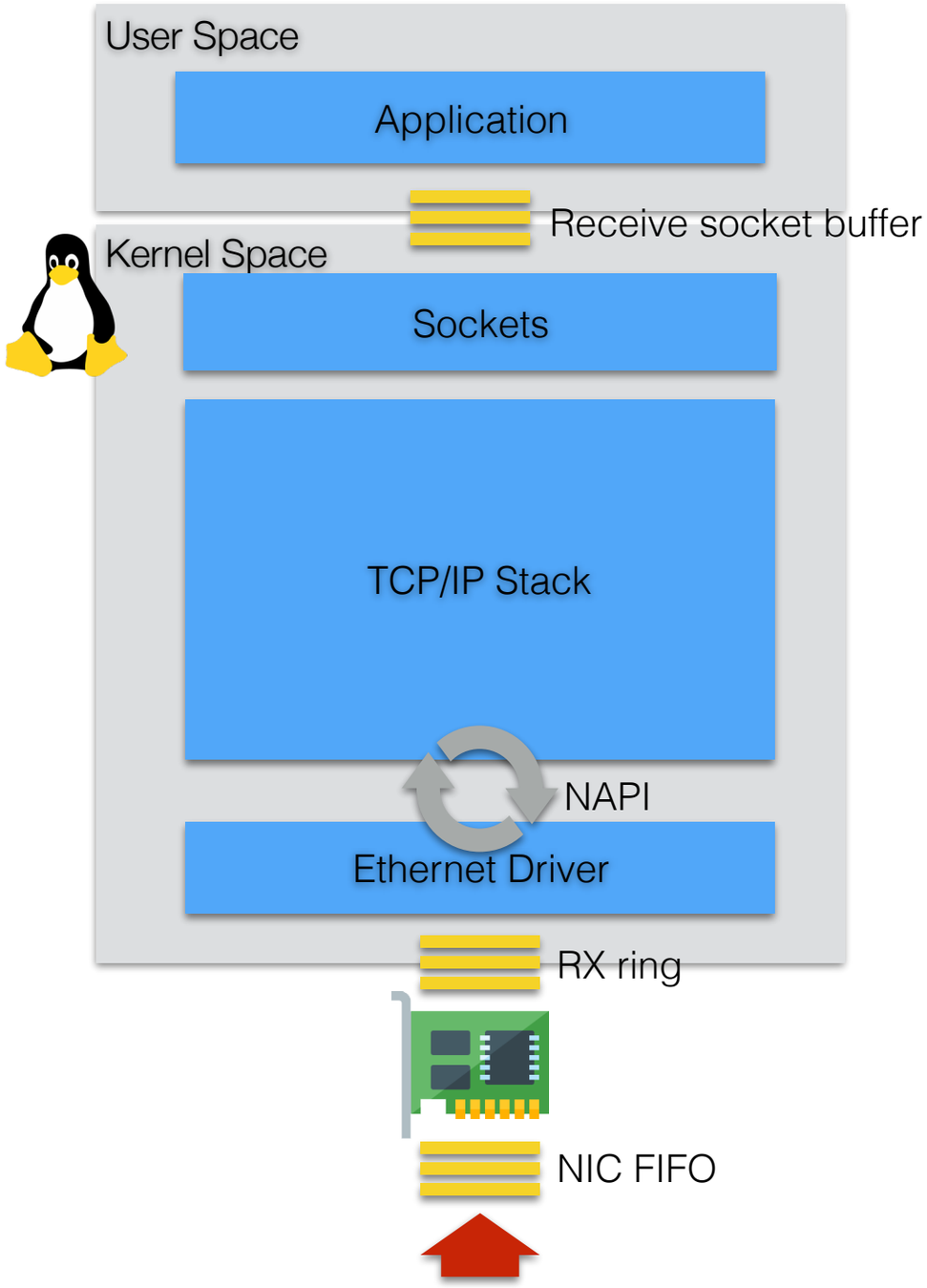
# Joining Forces:
# PF_RING and XDP

ntop

# Introduction

- Network Monitoring tools need **high-speed, promiscuous**, raw packet capture.
- Commodity network adapters and device drivers are designed for providing host connectivity and are not optimized for high-speed raw packet capture.
- Specialized adapters are often not affordable, or not flexible enough.

# Traditional Linux Networking

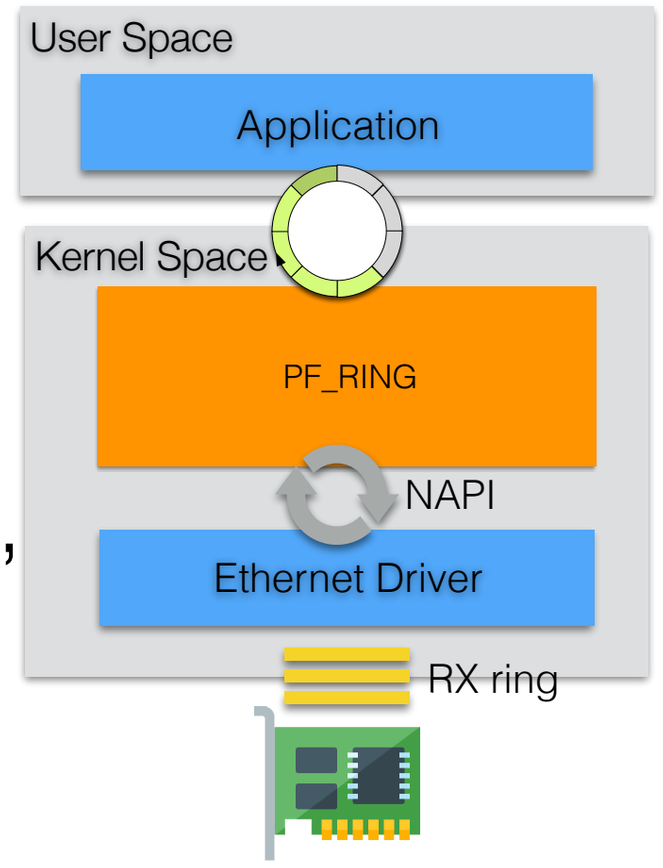


# High-Speed Capture is Hard

| Speed | Packets/sec | Nsec per packet |
|---|---|---|
| 1 | 1.48 Mpps | 675 ns |
| 10 | 14.8 Mpps | 67 ns |
| 100 | 148 Mpps | 6.7 ns |

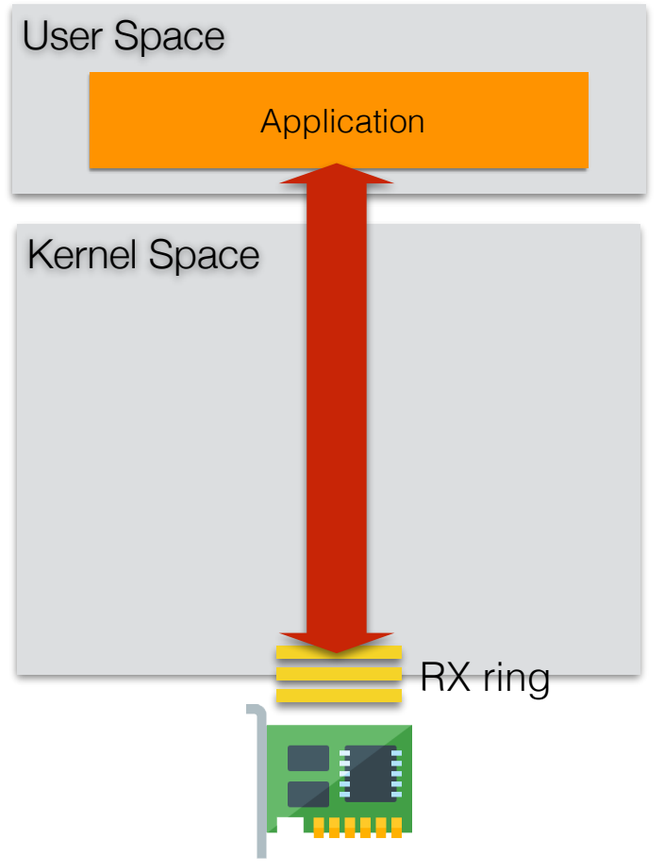
# PF_RING [1/2]

- Introduced in 2004 for improving the performance of network monitoring applications, accelerating **packet capture**.
- 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 [2/2]

- PF_RING today delivers wire-rate packet capture up to 100 Gbit on standard servers thanks to **kernel-bypass** ZC (Zero-Copy) drivers and support for many FPGA adapters.
- It is used by all ntop application, as well as third-party software (e.g. Bro, Snort, Suricata).

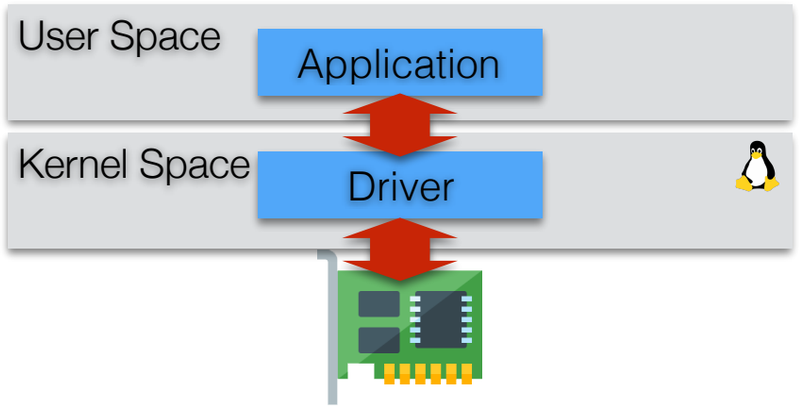


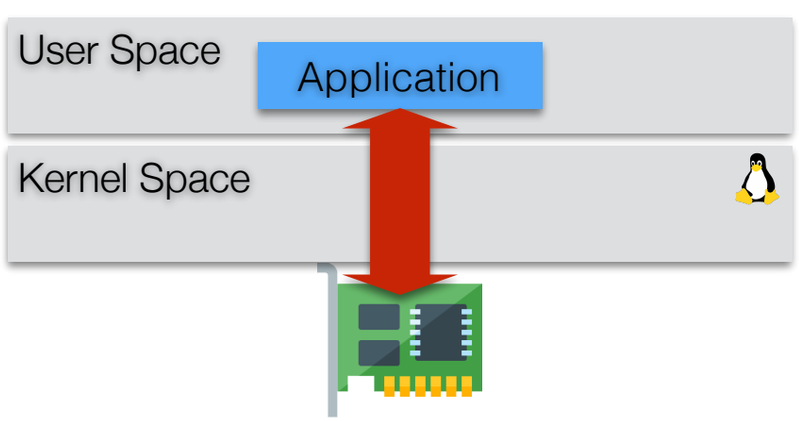
# Capture Technologies

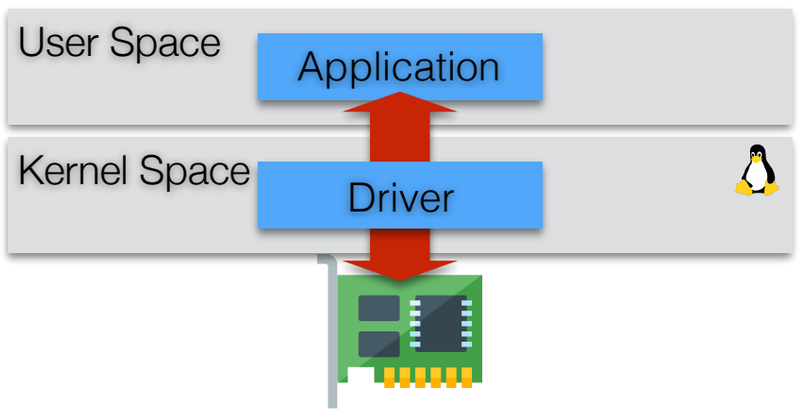
- Kernel-based
  - PF_RING
  - Libpcap/AF_PACKET

- Kernel-bypass
  - PF_RING ZC
  - DPDK
  - Snabb (Lua)

- Hybrid
  - XDP/AF_XDP
  - netmap



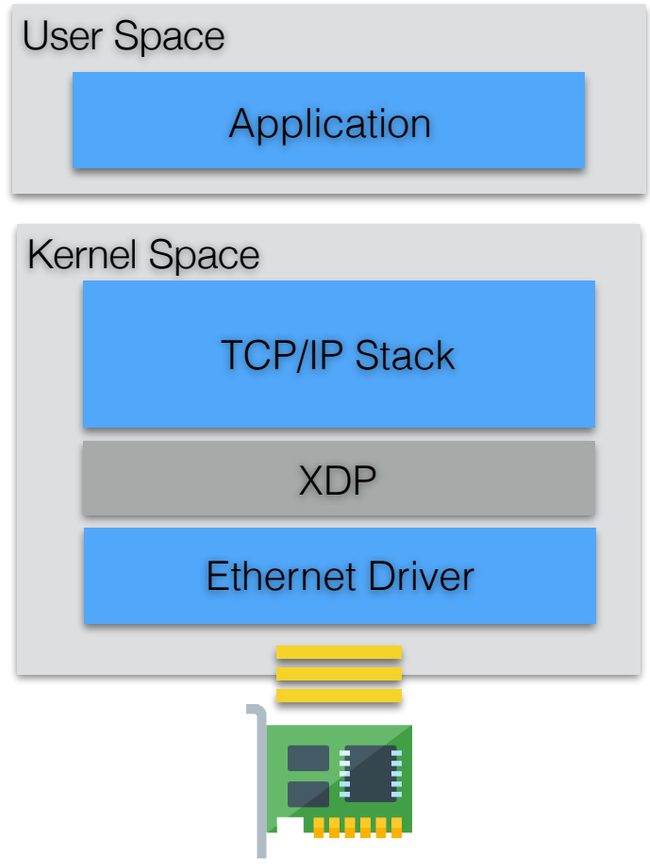




ntop

# XDP

- XDP (eXpress Data Path) is a new layer in the Linux kernel before the network stack.
- Not kernel bypass: data-plane inside the kernel.
- Drivers need to implement hooks to control (filter or redirect) packets before they reach the Linux stack.
- Programmable by means of eBPF.
- Under active development, all drivers will support it soon!

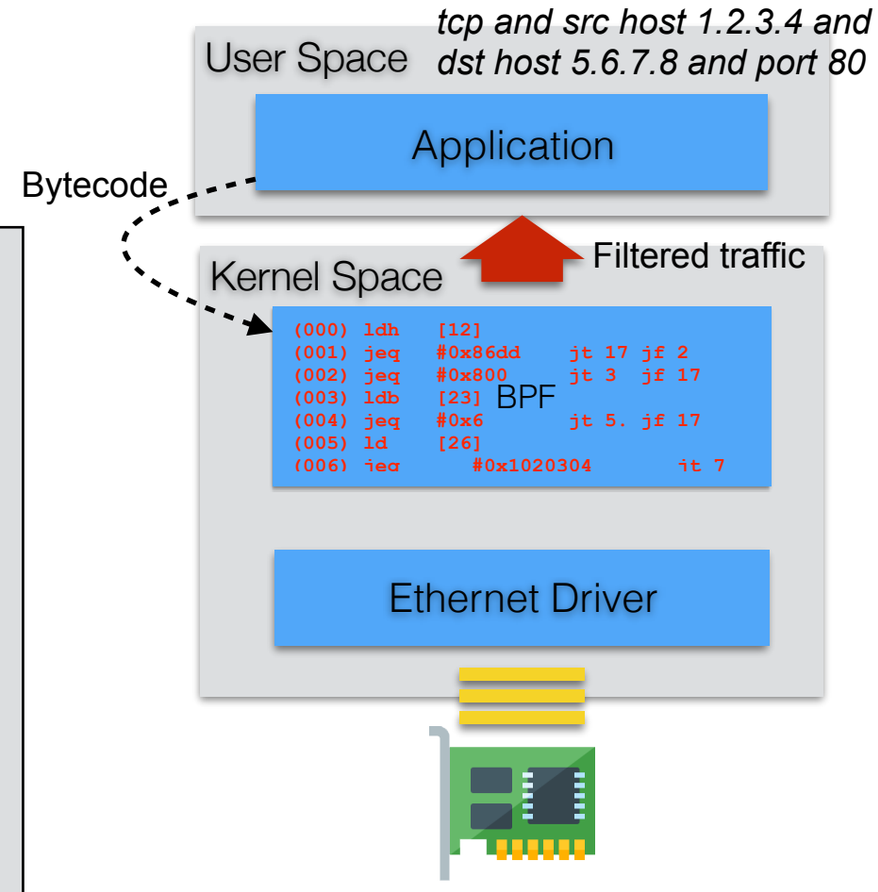



ntop

# BPF (aka cBPF)

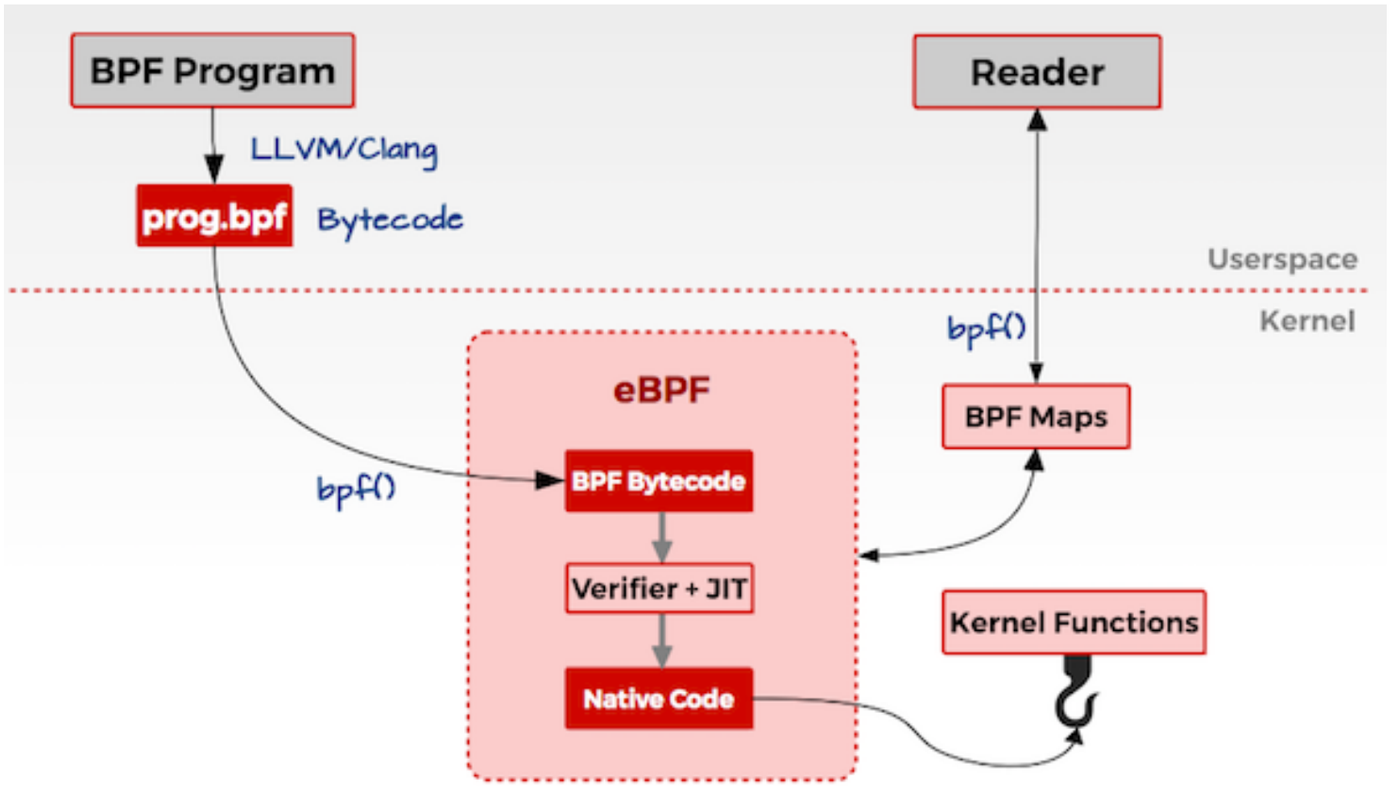
- Classic BPF (Berkeley Packet Filter) (1992)

```
tcpdump -ni em1 -d tcp and src host 1.2.3.4 and port 80
(000) ldh      [12]
(001) jeq      #0x86dd          jt 15  jf 2
(002) jeq      #0x800           jt 3   jf 15
(003) ldb      [23]
(004) jeq      #0x6             jt 5   jf 15
(005) ld       [26]
(006) jeq      #0x1020304       jt 7   jf 15
(007) ldh      [20]
(008) jset     #0x1fff          jt 15  jf 9
(009) ldxb     4*([14]&0xf)
(010) ldh      [x + 14]
(011) jeq      #0x50            jt 14  jf 12
(012) ldh      [x + 16]
(013) jeq      #0x50            jt 14  jf 15
(014) ret      #262144
(015) ret      #0
```



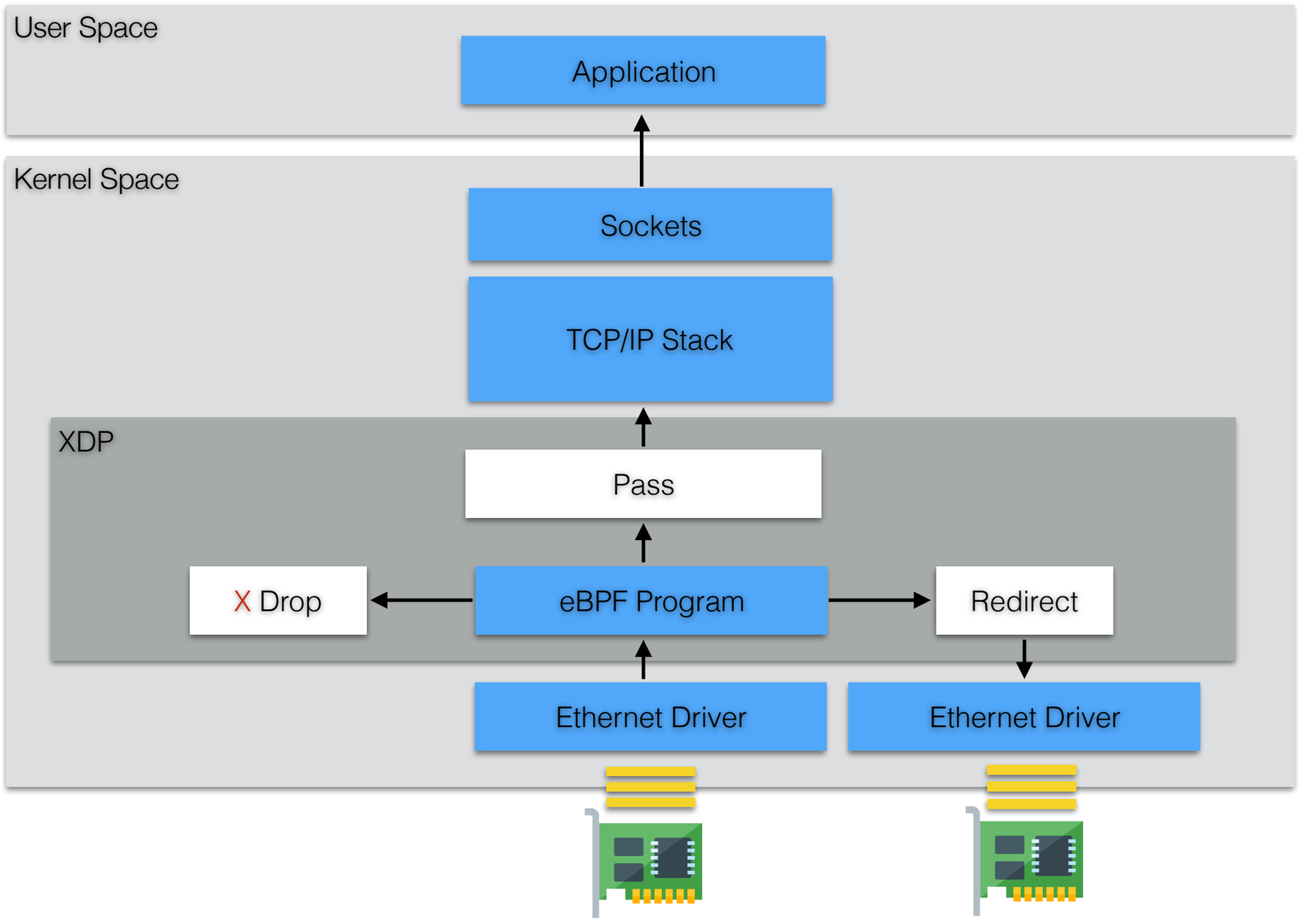
# eBPF

- Enhanced BPF (2014)
- User-defined bytecode executed by an in-kernel sandboxed virtual machine.
- In-kernel hooks to attach eBPF programs and run them when the corresponding code path is traversed.




ntop

# XDP Actions

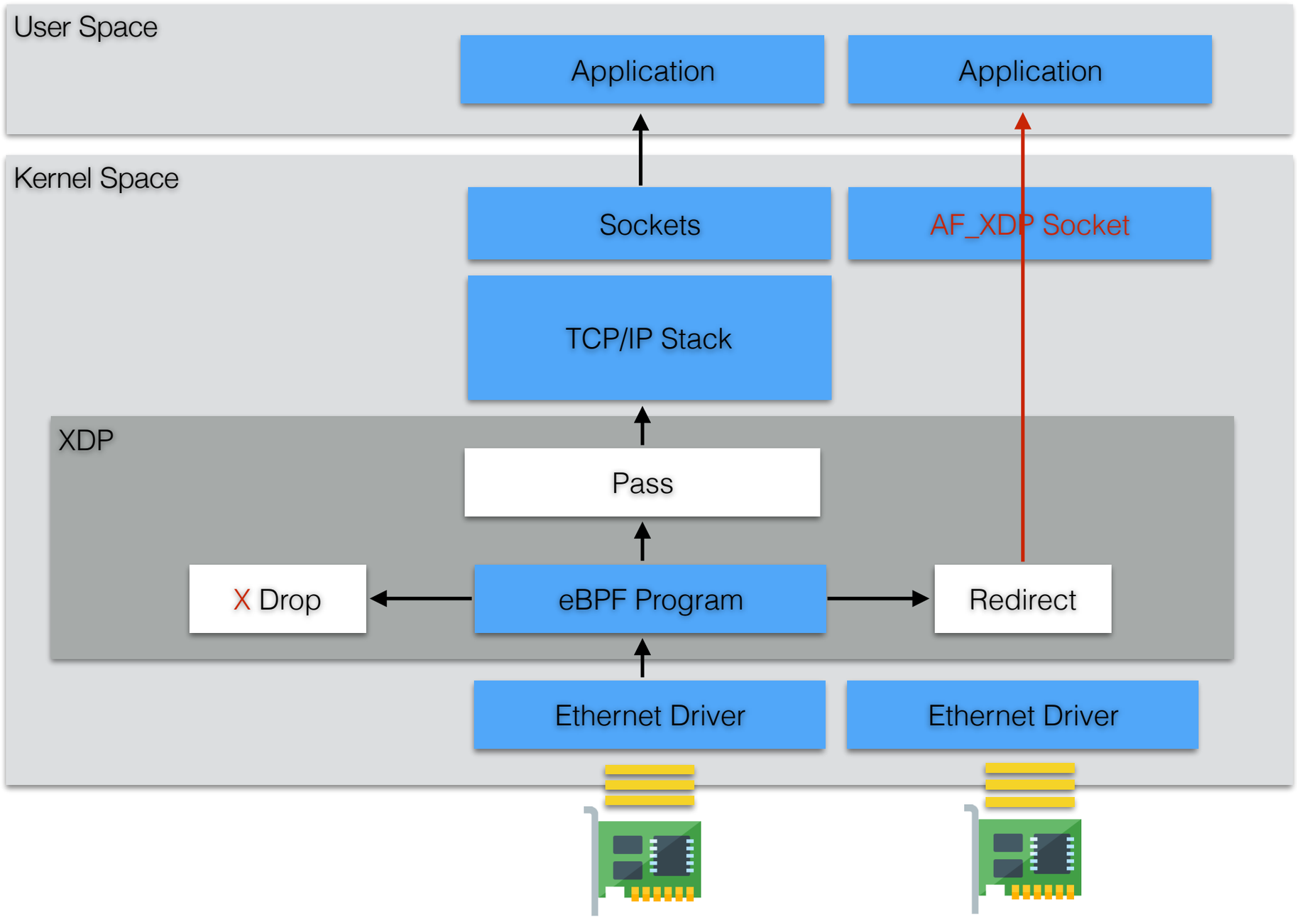



ntop

# AF_XDP

- AF_XDP is the socket used to deliver packets to userspace (applications).
- eBPF filters are used to REDIRECT packets into AF_XDP sockets.
- Integrated with the Linux network stack.
  - Access to kernel functionalities.
  - Forward selected traffic to userspace, while using the interface as a standard interface.

# Redirect to AF_XDP



# AF_XDP Pros/Cons

+ Adding driver support is fairly easy, all drivers will support it soon.

+ Almost the same performance as full kernel-bypass technologies.

+ Inside the kernel: integrated with the network stack.

- Inside the kernel: it does not support FPGA adapters implementing kernel-bypass.

- It just provides a pipe for moving packets from the driver to userspace, no enhanced packet processing features.

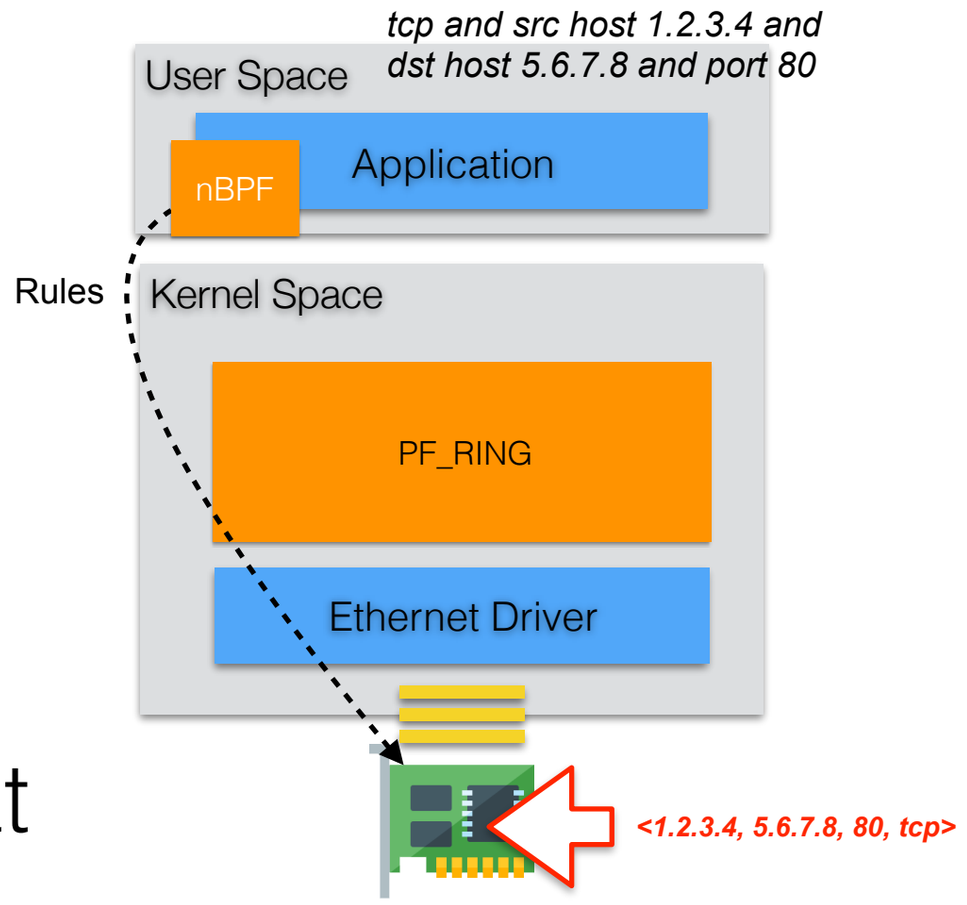
# AF_XDP and PF_RING [1/2]

- PF_RING is a rich framework providing, in addition to packet capture, support for:
  - Packet decoding
  - Filtering
  - Fanout/Load-balancing
  - L7 classification and filtering/shunting
  - …
- What about integrating AF_XDP with PF_RING?

ntop

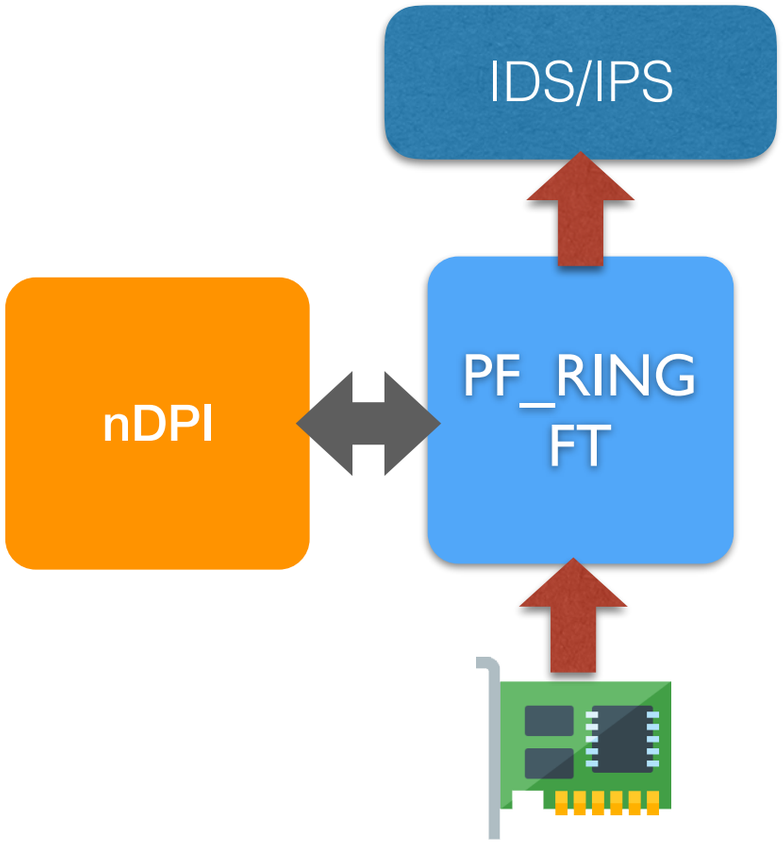
# Examples - BPF to Hardware Filtering

- Hardware filtering rules are not user-friendly
- **nBPF** is a filtering engine supporting the well-known BPF syntax
- It is able to generate hardware rules (able to filter traffic at high rates!), and filter in software what is not filtered by the card.



# Examples - L7 Support [1/3]

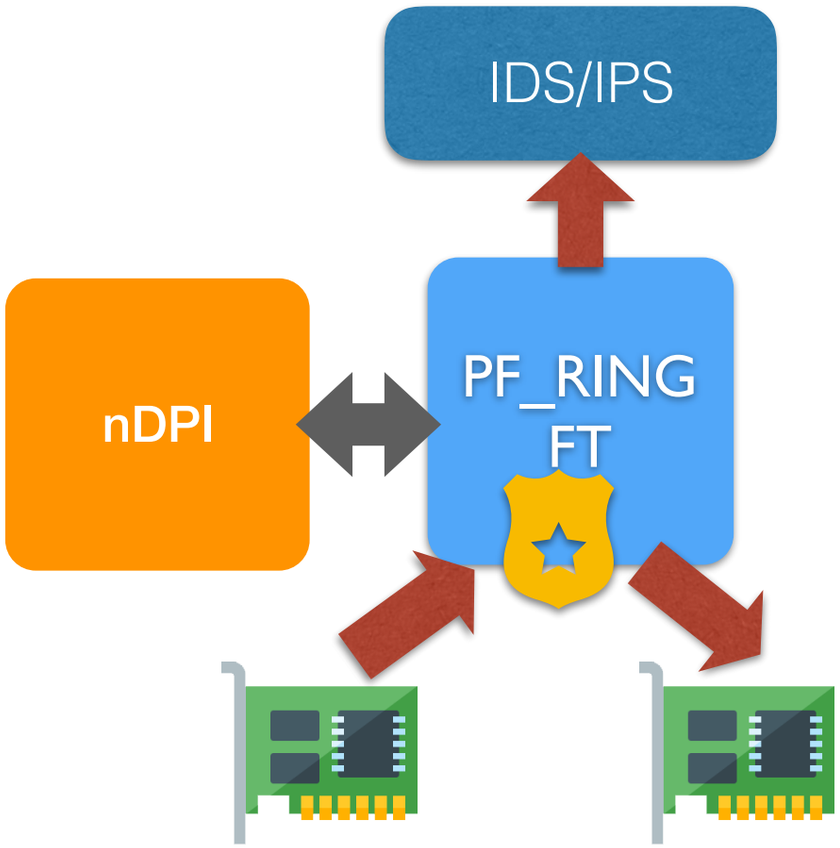
- **PF_RING FT (Flow Table)** is a stateful highly optimized library able to classify and filter L7 traffic.
- It leverages on **nDPI** to detect application protocols (250+ protocols including Facebook, Skype, Youtube, BitTorrent, …).




ntop

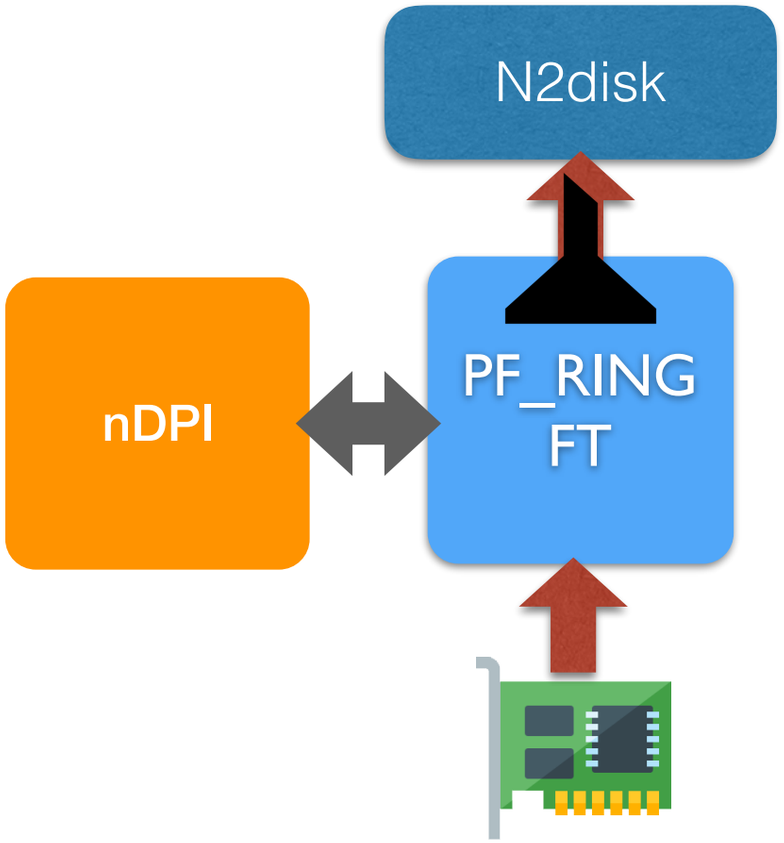
# Examples - L7 Support [2/3]

- Discarding elephant flows is becoming a common practice for reducing the amount of traffic an **IDS** needs to inspect, typically multimedia traffic (e.g. Netflix) dramatically improving the performance.
- No need to change a single line of code for applications already using PF_RING (e.g. Suricata, Bro) or libpcap-over-pf_ring.



# Examples - L7 Support [3/3]

- In **traffic recording** applications like **n2disk**, discarding unwanted traffic increases the data retention time.
- Filtering traffic based on static IP/Port fields is in many cases not a viable solution.
- Filtering out or shunting uninteresting traffic based on application protocols is probably a better option!
- E.g. for SSL we probably want to see the handshake only, while discarding all encrypted traffic taking a lot of disk space.



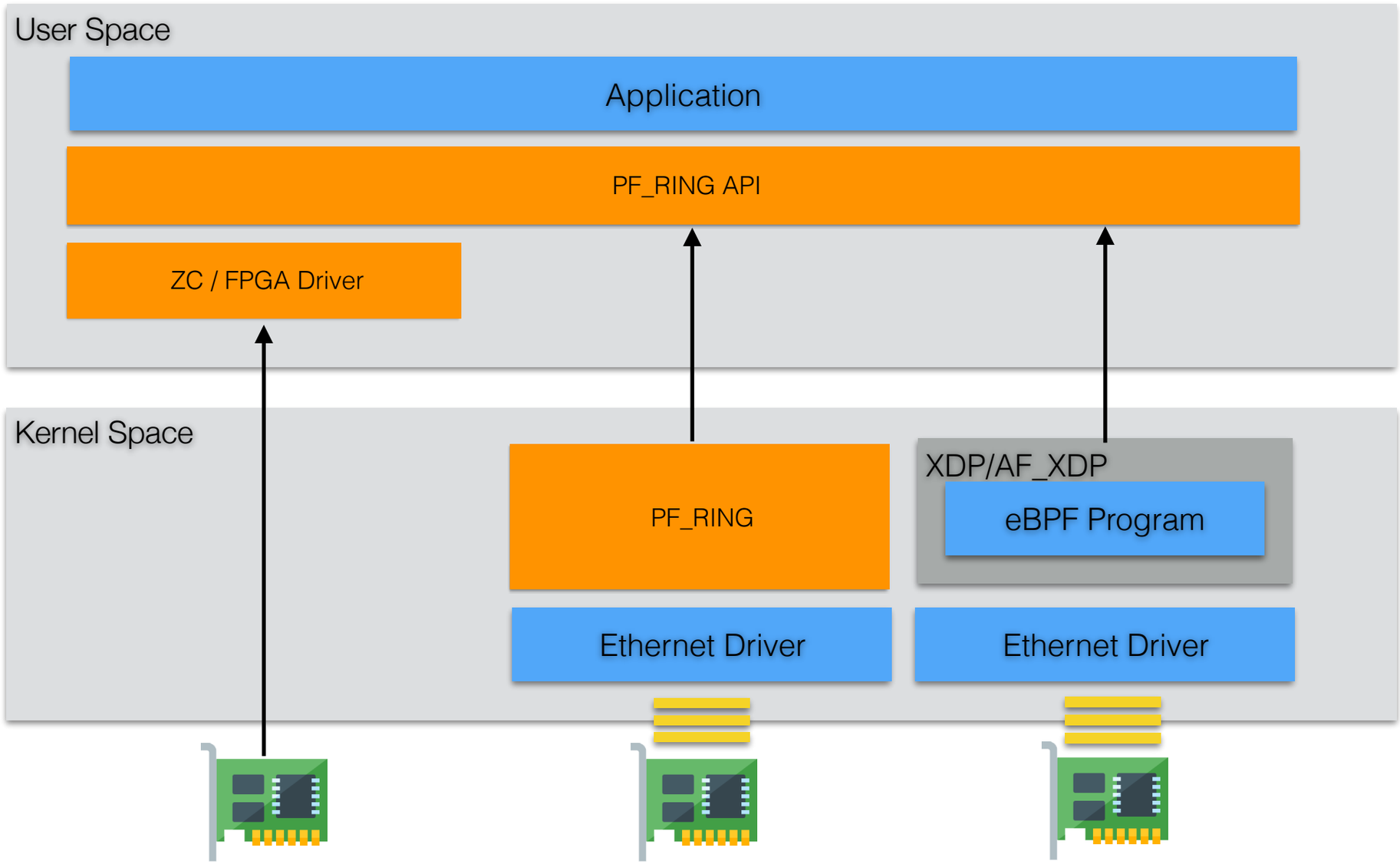
# AF_XDP and PF_RING [2/2]

- Full kernel-bypass for FPGA adapters and Intel cards with PF_RING ZC.

- Capture acceleration for other interfaces provided by AF_XDP.

Others…

# XDP Behind the PF_RING API




ntop

# Conclusions

- We are working on XDP integration in PF_RING to improve the performance of all adapters not supported by the PF_RING Zero-Copy drivers.
- This will combine the XDP advantages with the features provided by PF_RING for packet parsing, filtering, processing, fanout, load-balancing, L7 classification..
- It's still a bit too early as it is not yet supported by most of the kernels in production, but it's coming!

# Thank you!

ntop