Improving Passive Packet Capture: Beyond Device Polling

Luca Deri <deri@ntop.org>

ntop.org

Packet Capture: State of the Art

- Many available passive network monitoring tools are based on libpcap (http://www.tcpdump.org) or winpcap (http://winpcap.polito.it)
- Despite libpcap offers the very same programming interface across different OSs, the library performance are very different depending on the platform being used.
- Some library components (e.g. BPF packet filtering) have been implemented into the kernel for better performance.

Packet Capture: Open Issues

- Monitoring low speed (100 Mbit) networks is already possible using commodity hardware and tools based on libpcap.
- Sometimes even at 100 Mbit there is some (severe) packet loss: we have to shift from thinking in term of speed to number of packets/second that can be captured analyzed.
- Problem statement: monitor high speed (1 Gbit and above) networks with common PCs (64 bit/66 Mhz PCI bus) without the need to purchase custom capture cards or measurement boxes.

Libpcap Performance on a Vanilla OS

Traffic Capture Application	Linux 2.4.x	FreeBSD 4.8	Windows 2K
Standard Libpcap	0.2 %	34 %	68 %
mmap Libpcap	1 %		
Kernel module	4 %		

Percentage of captured packets [~80K packet/sec, ~45 Mbit]

Testbed:

- Sender: Dual 1.8 GHz Athlon, 3Com 3c59x Ethernet card
- Collector: VIA C3 533 MHz, Intel 100Mbit Ethernet card
- Network Switch: Cisco Catalyst 3548 XL
- Traffic Generator: tcpreplay (http://tcpreplay.sourceforge.net/)

Libpcap Performance Using Polling

Traffic Capture Application	Linux 2.6 with NAPI	FreeBSD 4.8 with Polling
Standard Libpcap	5.6 %	99.9 %
Mmap Libpcap	Not (yet) working	
Kernel module	99.5 %	

Percentage of captured packets (same test setup)

- Device Polling significantly improved the performance on a 100 Mbit Ethernet card
- Linux still performs much worse than FreeBSD at userspace
- Linux kernel performance is basically the same of FreeBSD at userspace

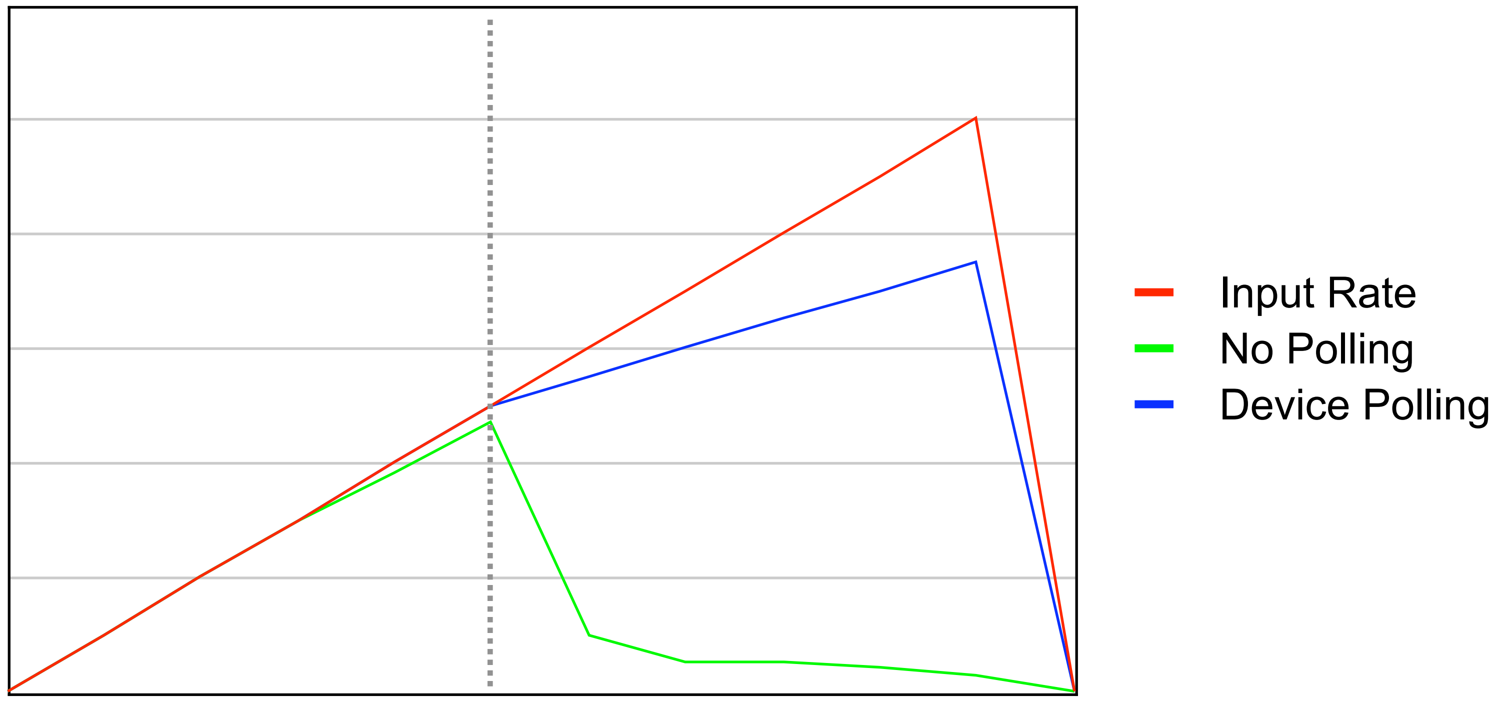
Libpcap Performance at Gbit Speeds

Traffic Capture Application	FreeBSD 4.8 with Polling
Standard Libpcap	148’000 (~3.7%)

Dropped packets [4’000’000 packet sample] (165-195k pkt/sec, ~480Mbit)

- While FreeBSD is usable at 100 Mbit, at Gbit the packet loss is greater (Linux still performs much worse).
- Device polling, nor card interrupt mitigation, is not enough for capturing almost everything.
- It is necessary to rethink the traffic capture architecture to achieve better speeds.

Polling vs. non Polling

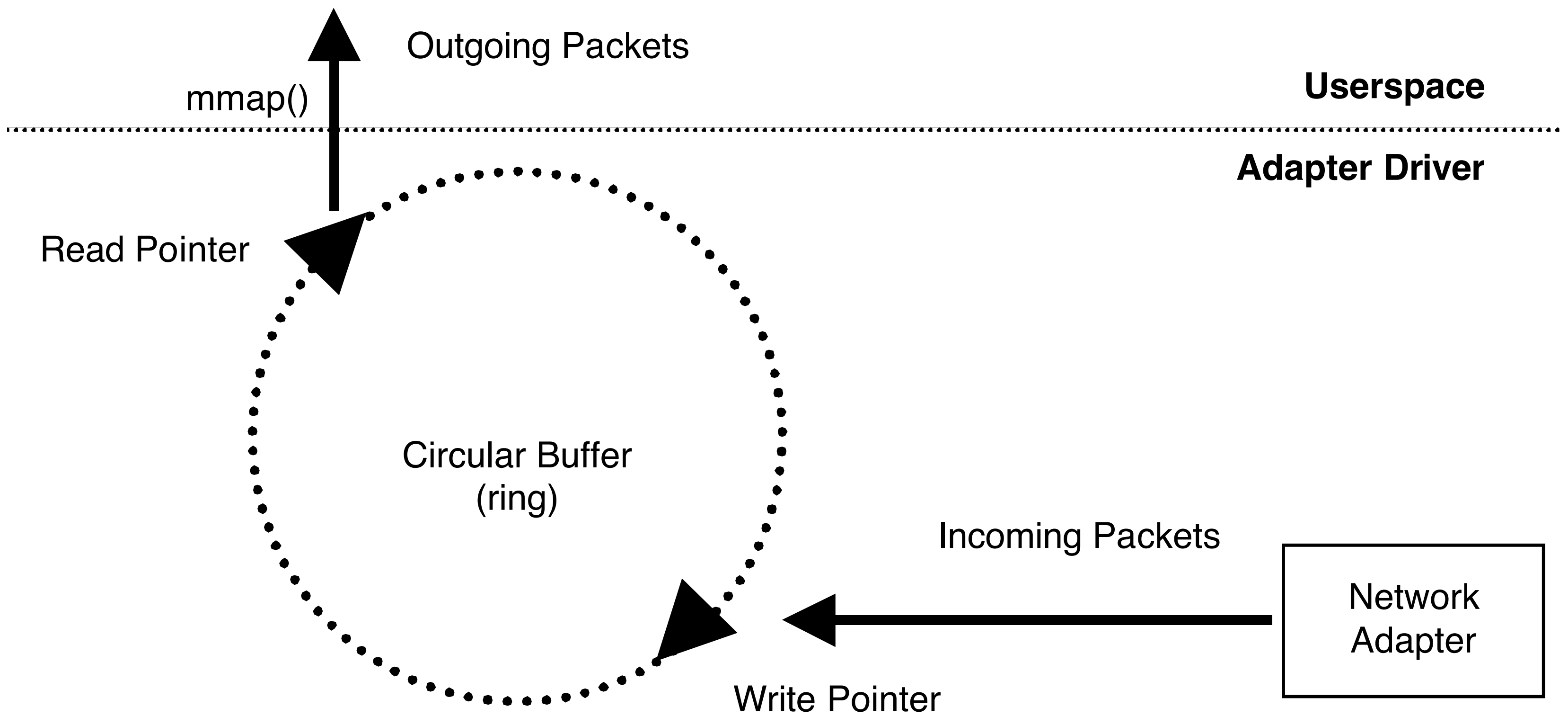


Sidenote. Polling is usually disabled by default in OSs as:

- Slow polling can have negative effects on packet timestamp precision.
- Fast polling can take over all/most of the available CPU cycles.

ntop.org

Proposed Solution: Driver Packet Ring



Driver Packet Ring: Features [1/2]

-
- The card driver (not the kernel) fills up the buffer.
- Features like packet sampling and filtering can be efficiently implemented into the driver (otherwise the packet arrives to the kernel/userland then is filtered/discarded).
- The bucket copy operation is very fast (memcpy() with no memory handling needed) so even with kernel that does not support device polling, under strong traffic conditions the system is usable.

Driver Packet Ring: Features [2/2]

- Straight path from the driver to userspace with no kernel overhead (beside the PCI bus). Note that packets are transfer over the PCI bus via DMA and that interrupts arrive only when the packet is already available to the driver.
- Packets are not queued into kernel network data structures but directly accessible to user-space applications.
- The mmap() primitive allows userspace applications to access the circular buffer with no overhead due to system calls such as in the case of socket calls.

Driver Packet Ring: Main Advantages

- Enable people to capture packets at (almost) wire speed without having to purchase a dedicated card or measuring box.
- It exploits both polling and interrupt mitigation (if available).
- Common belief that coding into the kernel is more efficient than in userspace is not completely correct. In fact a kernel module is slower than a userland application+ring as packets have to traverse kernel structures until they can talk with the kernel module.

Driver Packet Ring: Validation

Traffic Capture Application	Linux 2.6 with NAPI	FreeBSD 4.8 with Polling
Standard Libpcap		96.3% (148'000 drops)
Circular Buffer Libpcap	> 99.99% (160 drops)	

Captured packets [4'000'000 packet sample] (165-195k pkt/sec, ~480Mbit)

- Linux drops most of the packets at startup (polling issue?), little loss over the time.
- The CPU load on the Linux PC is < 10% (i.e. there's room for improvement), whereas is very high on FreeBSD (little CPU time left to measurement applications).

Driver Packet Ring: Current Status

- Port of the code to Linux 2.6.
- Currently available only for Intel cards (10/100/1Gb/10Gb).
- Easy to port to other card drivers as the code is device independent.
- Enhanced libpcap to seamlessly support the ring.
- Successfully tested several applications (e.g. tcpdump) on top of the ring in order to validate the code.
- Both ntop and nProbe (homegrown NetFlow v5/v9 probe for IPv4/6) run on top of the ring.

Work in Progress / Future Work

- Testing at 10 Gbit using a commercial card (e.g. Intel 10 Gbit) and comparison of its performance with custom cards (DAG, Combo6).
- Wishlist: port it to *BSD in order to evaluate the overall performance gain on a non-Linux platform.

Availability

- Paper and Documentation: http://luca.ntop.org/Ring.pdf
- Code (GNU GPL2 license): http://sourceforge.net/projects/ntop/