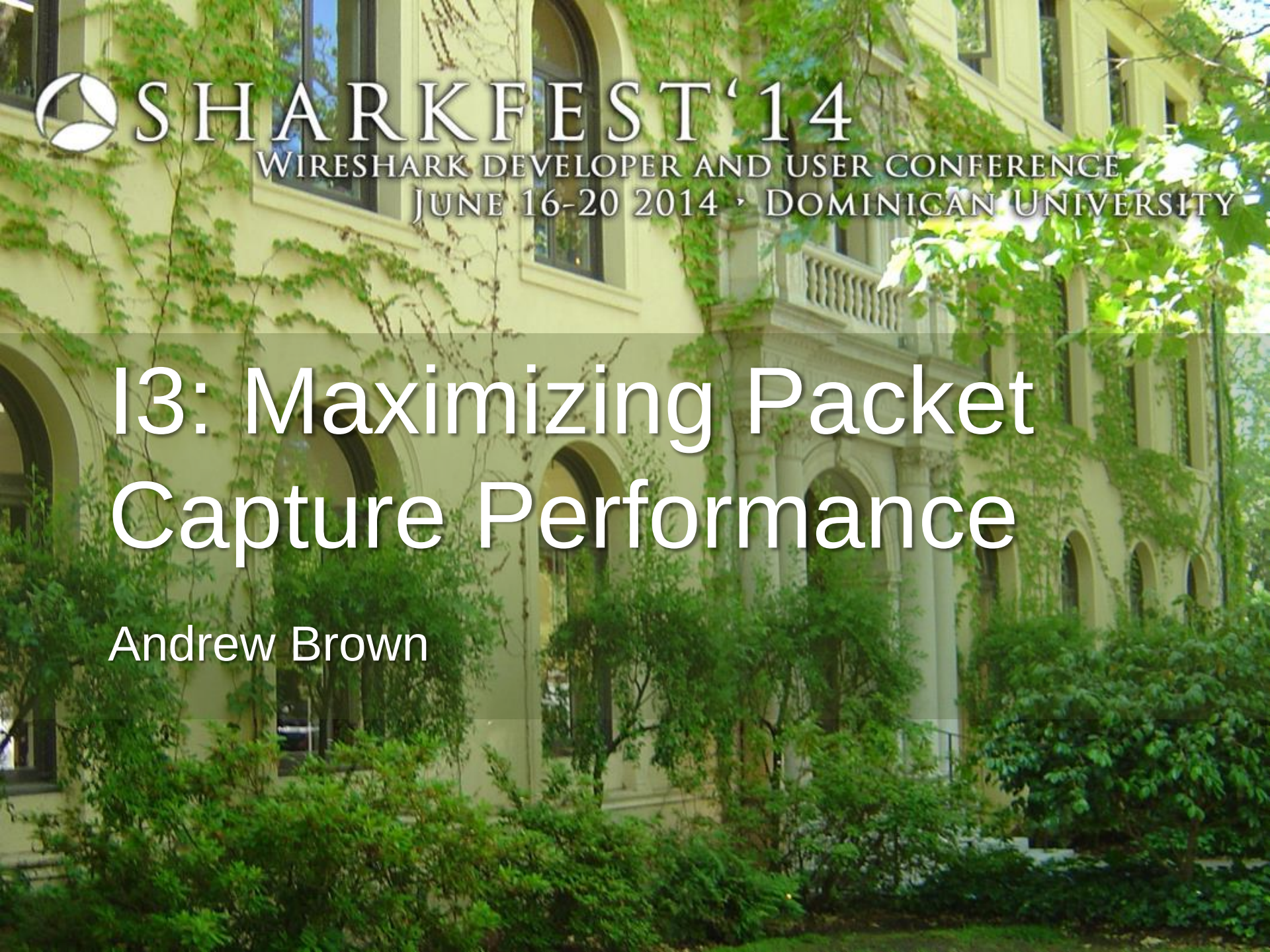


I3: Maximizing Packet Capture Performance

Andrew Brown

Agenda

- Why do captures drop packets, how can you tell?
- Software considerations
- Hardware considerations
- Potential hardware improvements
- Test configurations/parameters
- Performance results

What is a drop?

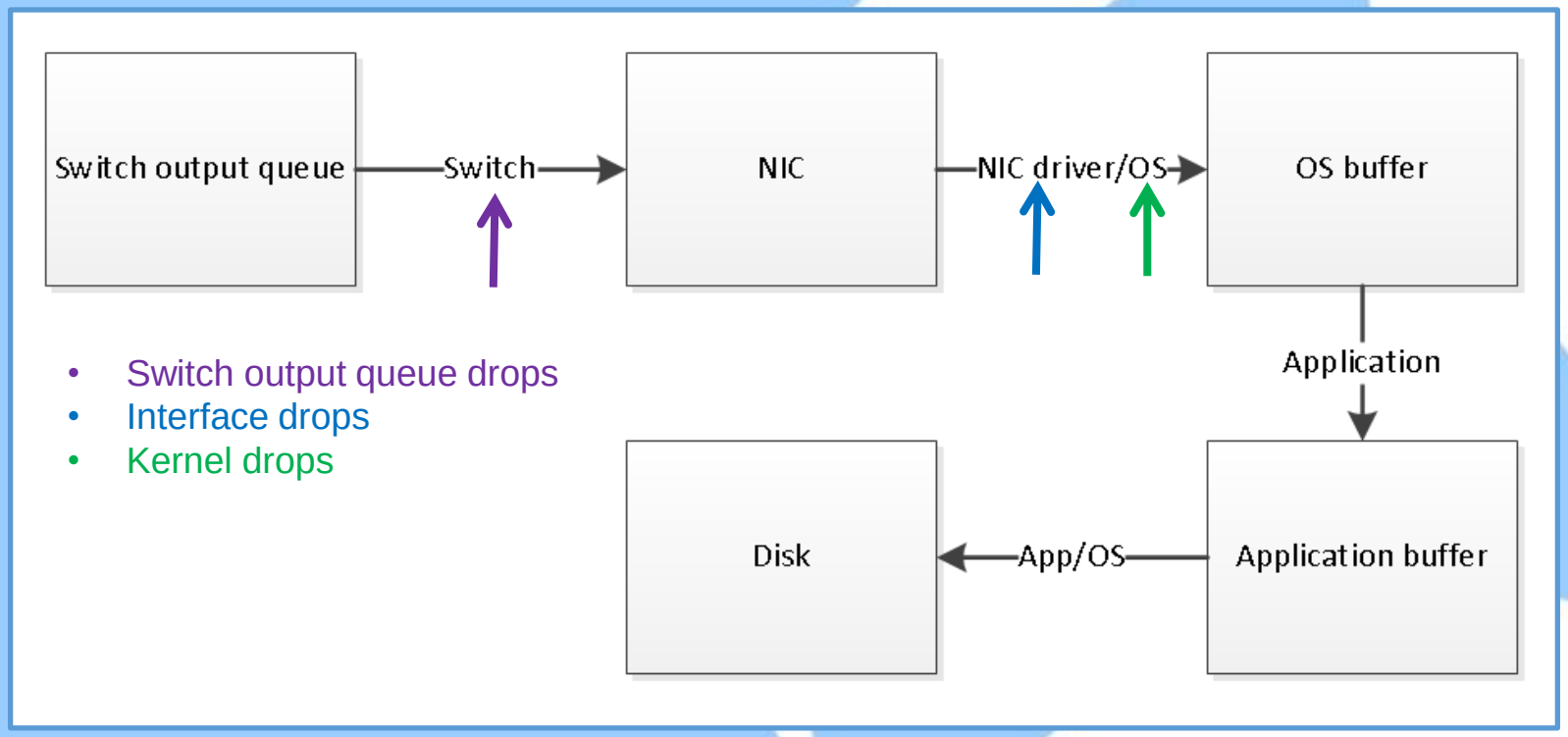
- Failure to capture a packet that is part of the traffic in which you're interested
- Dropped packets tend to be the most important
- Capture filter will not necessarily help

Why do drops occur?

- Applications don’t know that their data is being captured
- Result: Only one chance to capture a packet
- What can go wrong?
  Let’s look at the life of a packet

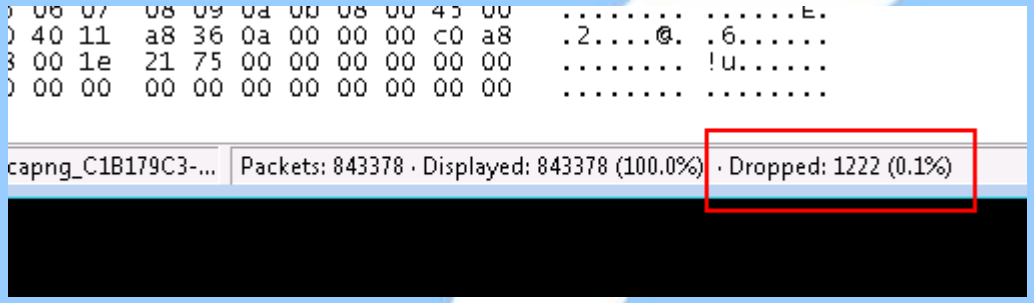
Internal packet flow

- Path of a packet from NIC to application (Linux)



Identifying drops

- Software reports drops



- L4 indicators (TCP ACKed lost segment)
- L7 indicators (app-level sequence numbers revealed by dissector)

When is (and isn't) it necessary to take steps to maximize capture performance?

- Not typically necessary when capturing traffic of <= 1G end device
- More commonly necessary when capturing uplink traffic from a TAP or SPAN port
- Some sort of action is almost always necessary at 10G
- Methods described aren't always necessary
- Methods focus on free solutions

Software considerations - Windows

- Quit unnecessary programs
- Avoid Wireshark for capturing
  - Saves to TEMP
  - Additional processing for packet statistics
    - Uses CPU
    - Uses memory over time, can lead to out of memory errors

Software considerations – Windows *(continued)*

- Alternative? Dumpcap
  - Command-line utility
  - Called by Wireshark/Tshark for capture
  - Provides greater control
  - Dumpcapui for CLIphobic
  - "At the limits" example
    - Dumpcap captured 100% of packets sent
    - Wireshark captured 68% of packets sent

Software considerations – Windows *(continued)*

- Windows dumpcap buffer tuning
  - Large buffers are generally good, but…
  - Increased bandwidth has a tipping point
    - Write to disk slows significantly
    - Larger buffers make it worse
    - Made buffer selection for testing difficult
    - Best option seemed to be 50MB

Software considerations – Windows *(continued)*

- Dumpcap "slow count" example
  - Sending 844,600 packets @ .4Gb
  - Packets take 1.48 seconds to send
  - 20MB buffer takes ~2.5 seconds to write
  - 512MB buffer takes ~46 seconds to write
  - Neither setting captured all packets
  - Not cosmetic (break out and file is truncated)
  - Issue disappears at lower bandwidth

Software considerations – Windows *(continued)*

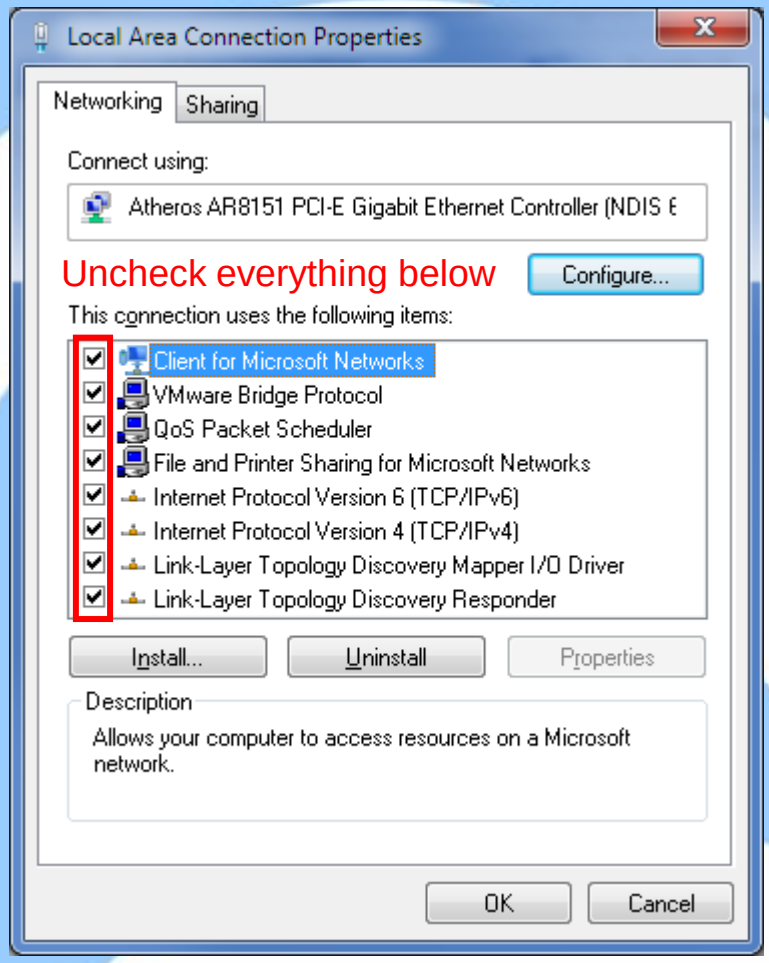
- Video of normal count

Software considerations – Windows *(continued)*

- Video of “slow count”

Software considerations – Windows *(continued)*

- Disable protocols on interface (TAP/SPAN)
  - Pure TAP/SPAN capture
  - Only for TAP/SPAN
  - Prevents OS from attempting to interpret packets
  - Tested performance with destination MAC set to broadcast address
  - Result: Captured 100% with protocols disabled, only 40% when enabled
  - Eliminate performance impact immediately after link up



Software considerations – Linux

- Quit unnecessary programs
- Use tcpdump with 512MB buffer
- Ensure libpcap >= 1.0.0 (tcpdump -h)
- Watch value of -s flag
- No option to disable protocols like Windows
- Static (or no) IP for dedicated capture interface
- Use XFS with RAID and coordinate stripe sizes

Software considerations – Linux
(continued)

- Access to development resources? Look at PF_RING
  - Module/NIC driver combination
  - Improves capture performance
  - Included tcpdump wasn't better than stock
  - We use the API and it works
  - Different performance tiers some are free

Software considerations – Linux
(continued)

- PF_RING
  - Kernel module/NIC driver combination
  - Improves capture performance via various methods
  - Included tcpdump wasn't better than stock
  - We use the API and it works
  - Different performance tiers some are free

Hardware considerations - Storage

- 1Gb line rate traffic generates 123-133MB in one second
- WD Black 7.2K RPM: 171MB/s
- WD Raptor 10K RPM: 200MB/s
- If 10Gb is 10X 1Gb… (do the math)
- SSD: ~500MB/s
- RAM disk is another option

Hardware considerations - CPU

- Three considerations
  - Number of cores
  - Clock speed
  - Performance per clock
- Clock speed * PPC = Per-core performance
- Multicore is good …
- … but per-core performance is better than many cores

Hardware considerations - NIC

- Intel (regular NIC)
  - Drivers more actively maintained
  - Best PF_RING support
  - 10G NIC doesn't help with 1G capture (1G and 10G NICs had the same max bandwidth at or below 1G)
- Avoid USB NICs
  - USB 2.0 is too slow (480Mb/s)
  - USB 3.0 didn't perform well

Benchmark methodology

- Tested limits of capture configurations at 1G and 10G
  - For each configuration, increase bandwidth until it fails
  - Failure is defined as not capturing all packets
  - Highest performing solutions formed basis for recommendations

Obvious question: Traffic profile?

- If not testing for a specific use case, what is the appropriate traffic with which to test?
  - What mix of TCP/UDP?
  - What duration, frequency, severity of bursts?
  - What mix of small/large packets?

(My) Answer: Many copies of a single packet with tests at various packet sizes

- Takes Receive Side Scaling out of the picture
- Removes buffering from the equation
- Tends to be pessimistic

Test configuration

- Unicast UDP packet used for (almost) all tests
- Packet sizes of 64, 128, 256, 512, 1024, 1500 bytes
  - Additional CPU overhead for every packet
  - One second at 1Gb is ~82K 1500 byte packets
  - One second at 1Gb is ~1.49M 64 byte packets
- Number of packets tailored to generate a ~1.5GB capture file
- Careful to eliminate disk as a bottleneck

Improving performance
The ideal

- Ideal capture laptop
  - Fast CPU
  - Fast storage (SSD RAID)
  - Dedicated Intel NIC
  - 10G capability
- Perfect except for one issue

  …it doesn't exist

Improving performance Thunderbolt

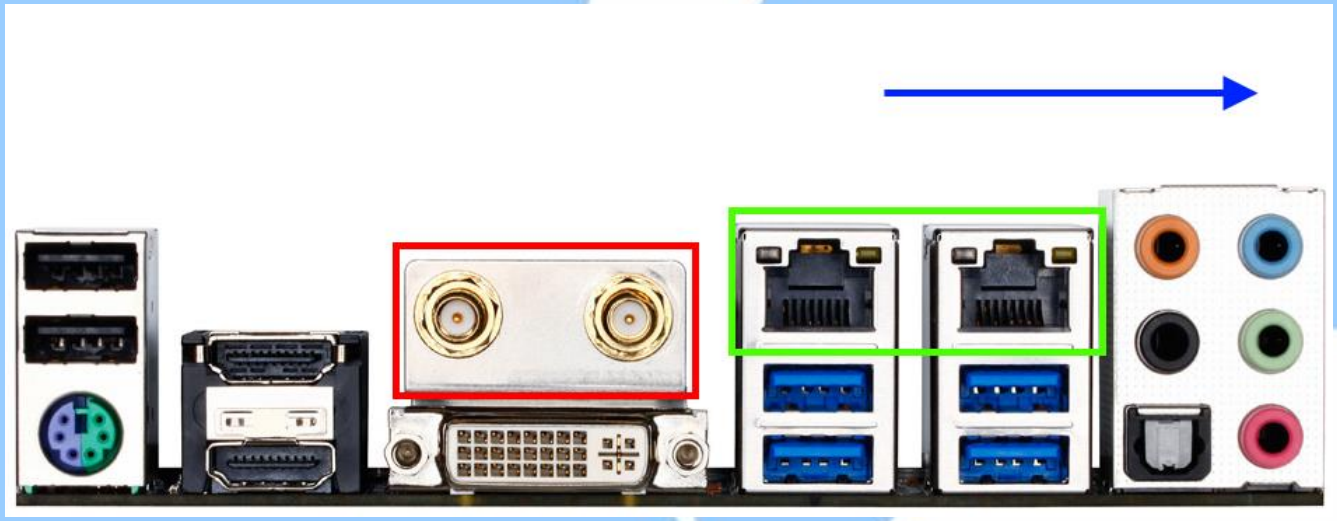
- PCIe via a cable (developed by Intel)
- Allows use of desktop cards on a laptop
- Expensive
- Not very widespread (mostly Apple computers)
- Other laptop limitations are still a problem

Improving performance
Laptop alternative

- What level of performance is possible from (relatively) portable commodity hardware?
- Packet toaster
  - Used for all capture testing
  - Intel i5 4570 desktop CPU (3.6GHz quad-core)
  - Up to 16GB RAM for RAM disk
  - Up to 4 SSD in RAID 0
  - Cost ~$800 with 8GB RAM, 2 SSDs
- Concept: Run without monitor, manage via laptop

Packet Toaster port layout

- Intel 1G NIC
- Additional 1G NIC for management (SSH/RDP)
- 802.11n for capture (Linux) or management
- PCIe slot for 10G



Solarflare

- Low-latency NIC with stack bypass
- Why include it?
  - Price competitive with other commodity 10G NICs
  - Works as a regular NIC under Linux, Windows, Mac etc.
  - Works at 1G also
  - SolarCapture app for high-performance Linux capture
- Hardware/software capture solution
- Tested with Packet Toaster and MacBook Pro (via Thunderbolt)

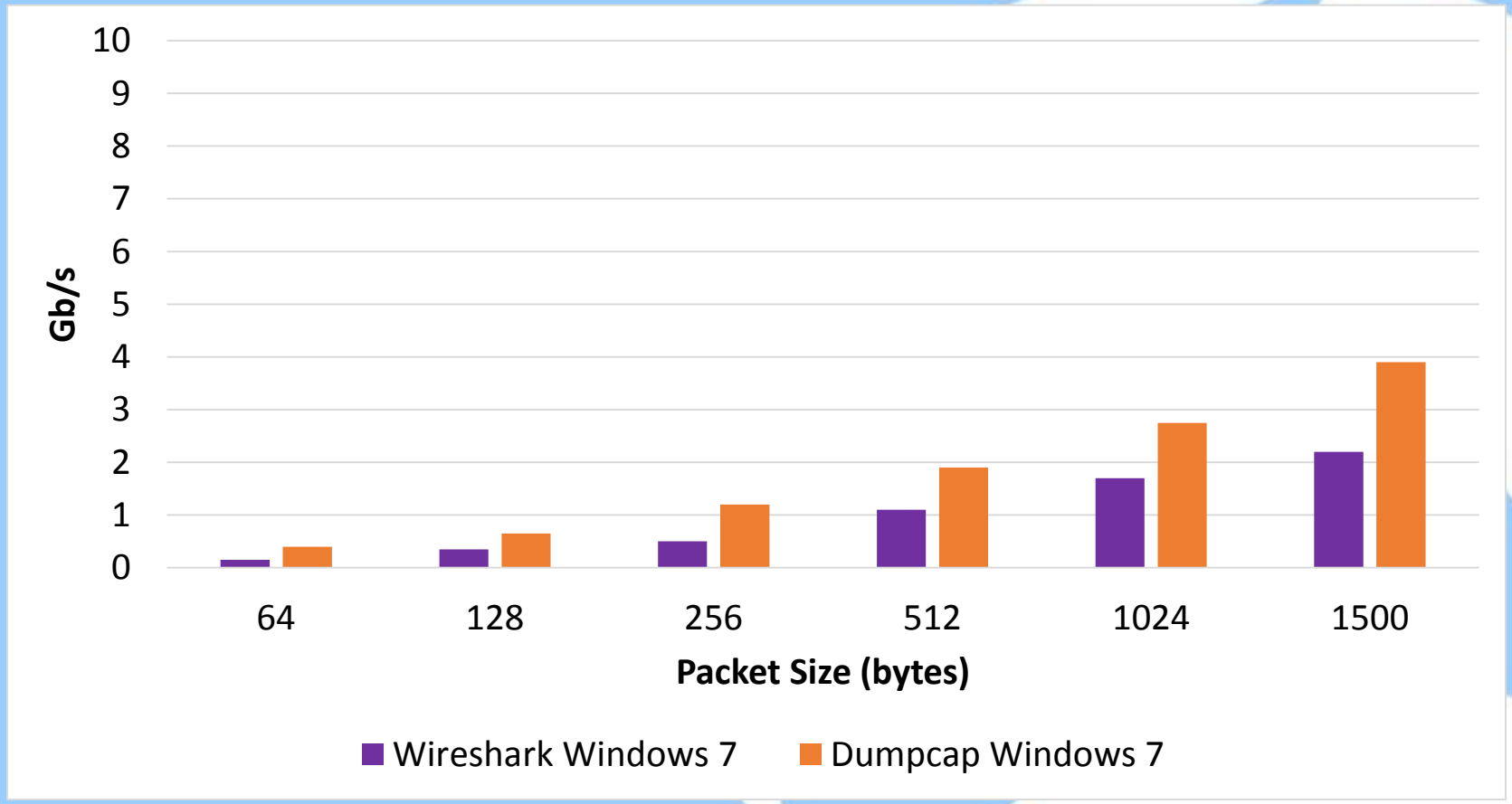
The difference a week makes

- At the time of testing, SolarCapture was a free download
- Less than a week ago, Solarflare changed licensing tiers; free SolarCapture is no longer available
- Pricing is reasonable (in my opinion) but…
  - …reasonable is relative
  - …this breaks my original concept of free software
- Debated removing results but couldn't (impacted other results and no time to re-test)

Performance Results Configurations

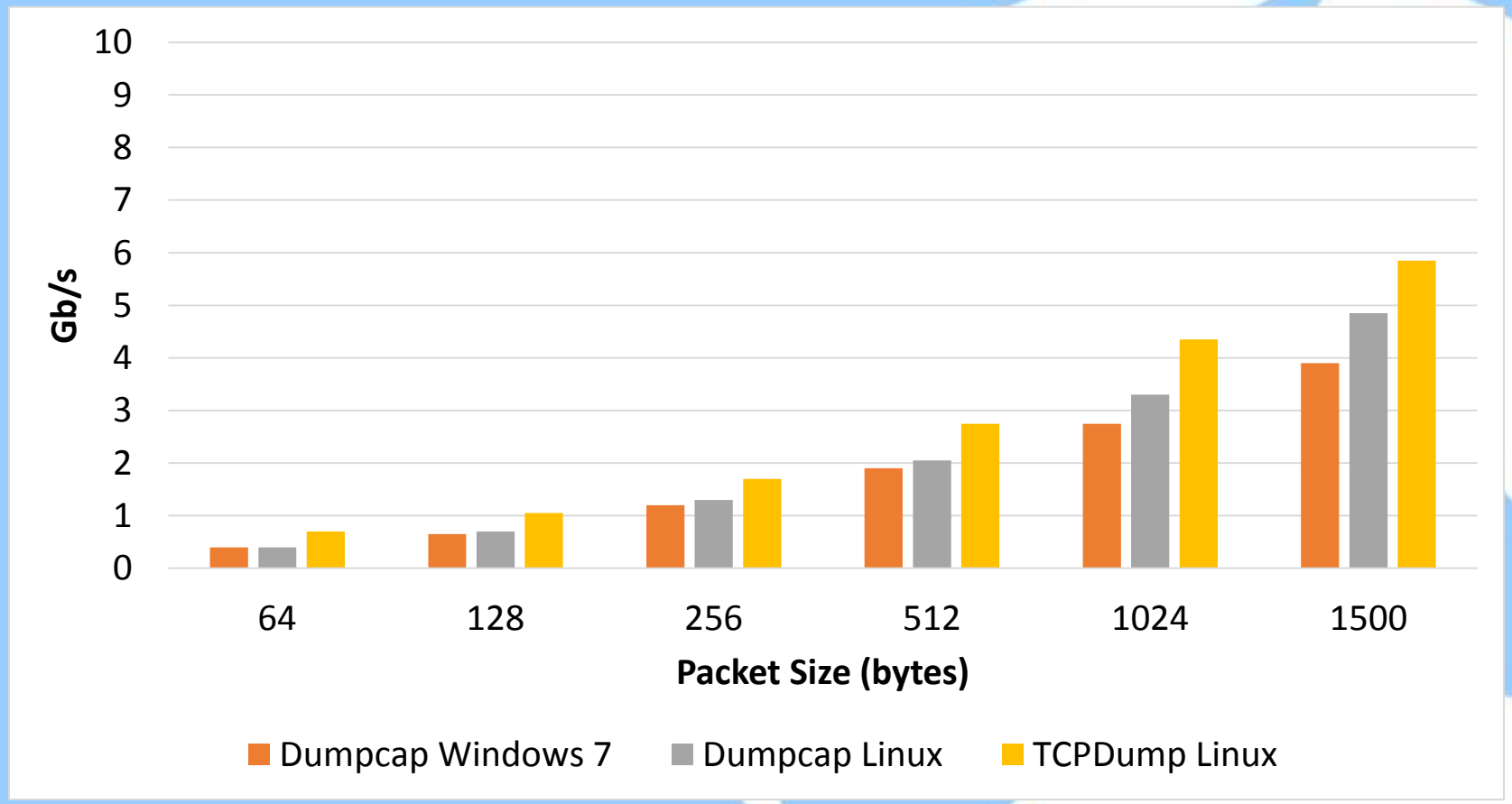
- Wireshark under Windows 7 (SSD)
- Dumpcap under Windows 7 (SSD)
- Dumpcap under Linux (SSD)
- TCPDump under Linux (SSD)
- SolarCapture under Linux on MacBook Pro via Thunderbolt (RAM)
- SolarCapture under Linux (SSD)
- SolarCapture under Linux (RAM)

Performance Results
Wireshark vs. Dumpcap (Win 7)



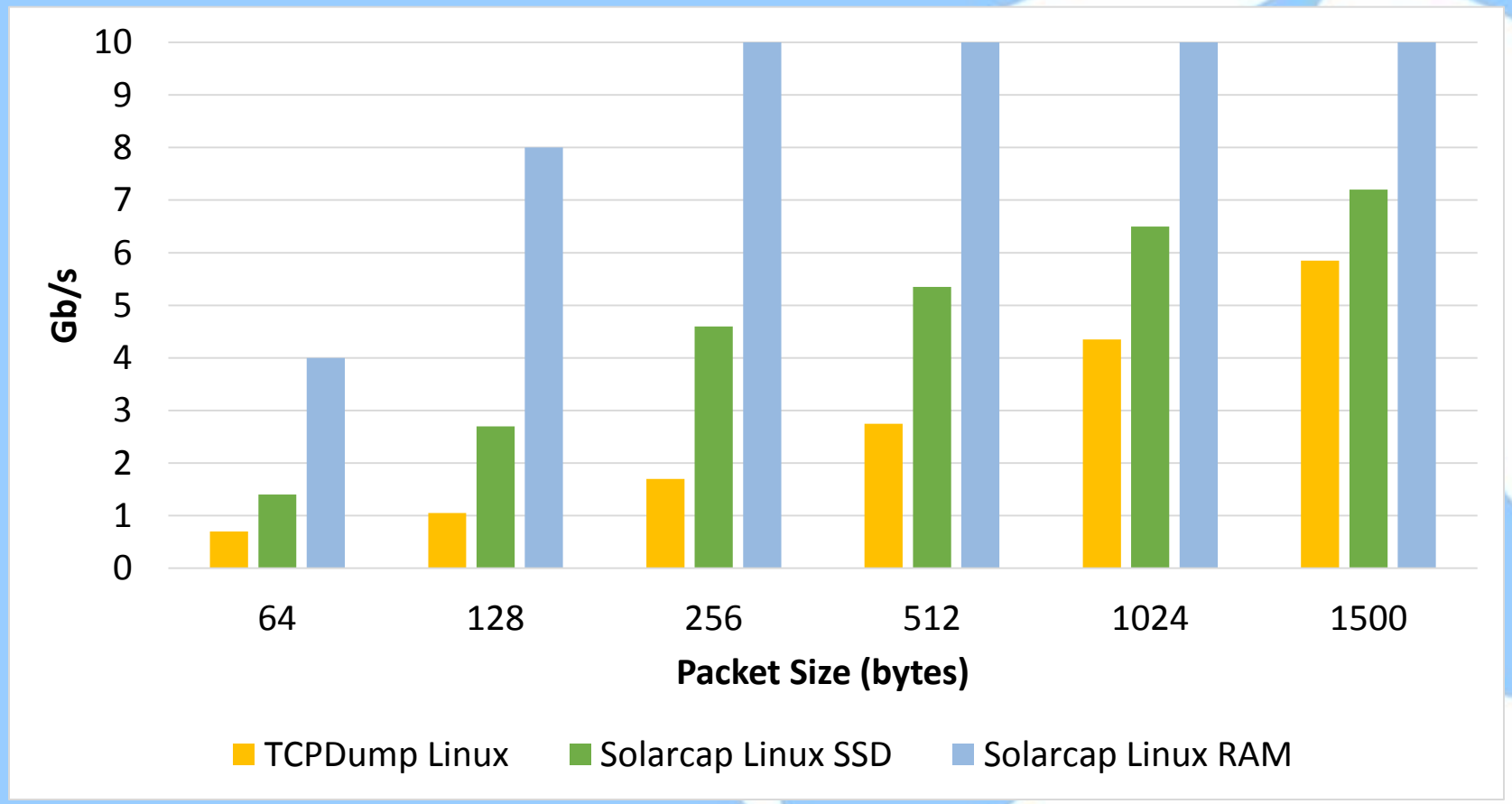
Performance Results

Dumpcap (Win7) - Dumpcap (Linux) – TCPDump (Linux)



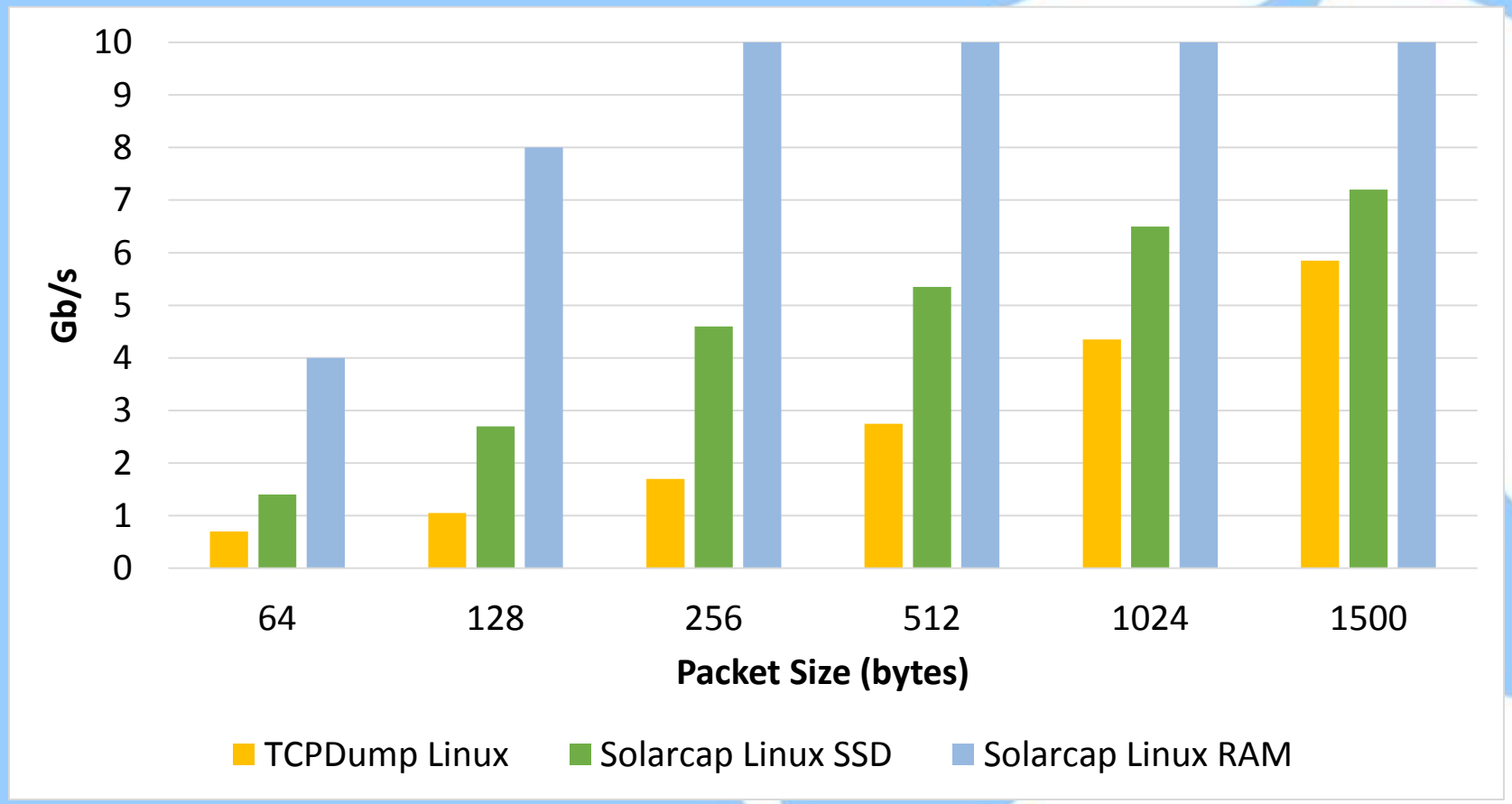
Performance Results

Dumpcap (Win7) - Dumpcap (Linux) – TCPDump (Linux)



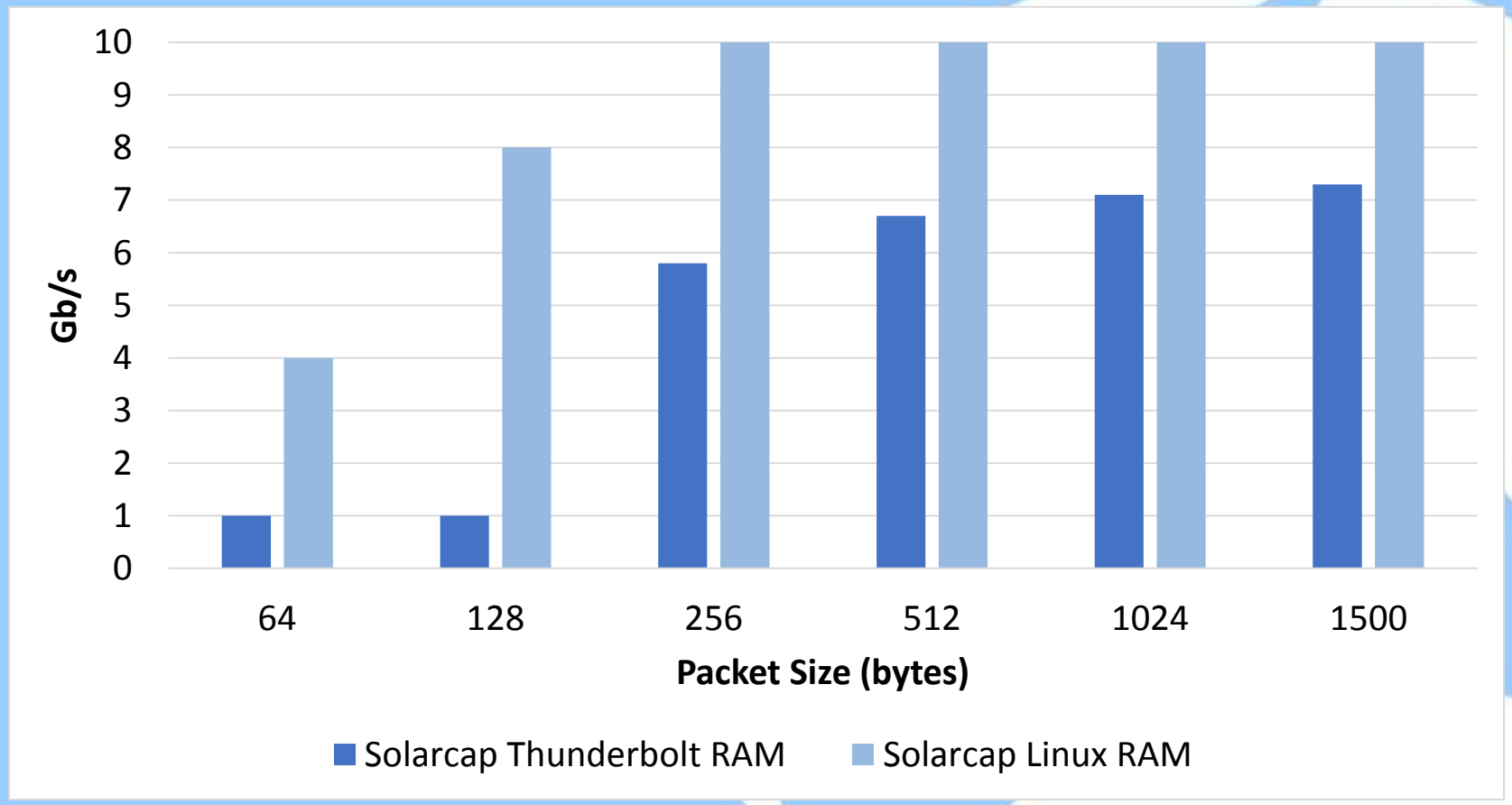
Performance Results

TCPDump (Linux) – SolarCapture (SSD) – SolarCapture (RAM)

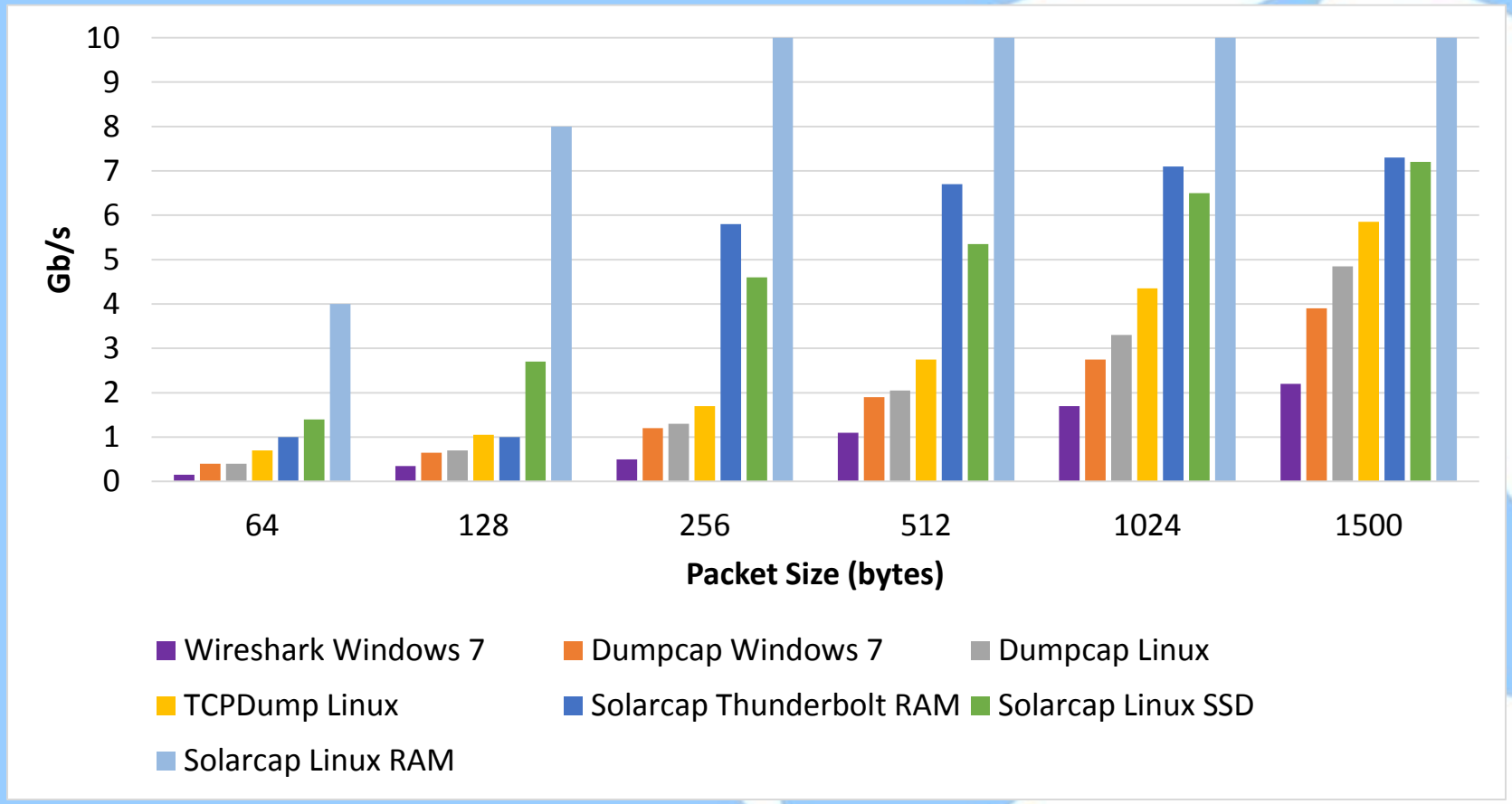


Performance Results

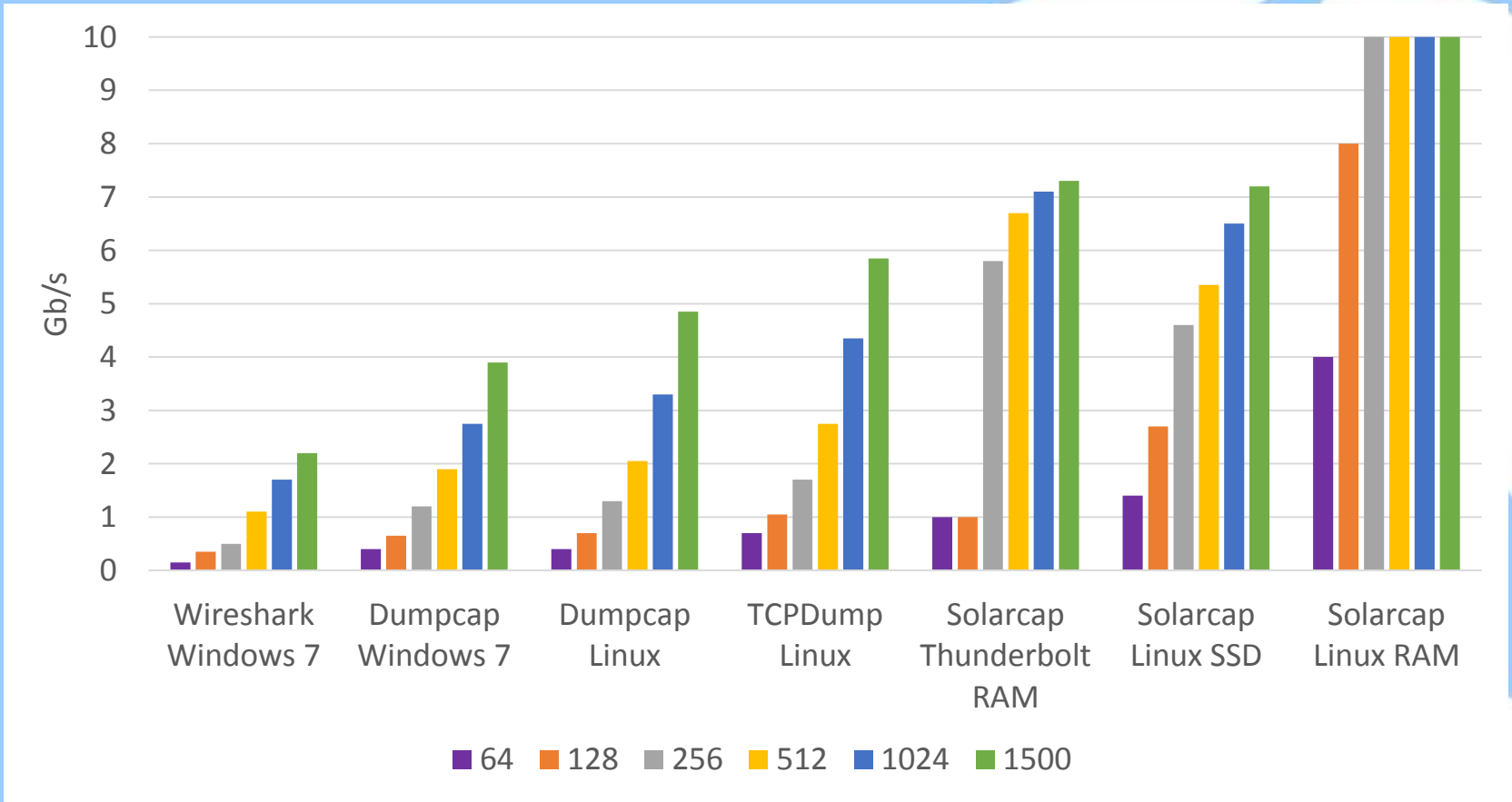
Dumpcap (Win7) - Dumpcap (Linux) – TCPDump (Linux)



Performance Results By Packet Size



Performance Results By Configuration



Acknowledgements

- BATS Global Markets
- Guy Harris
  - Core developer: libpcap, tcpdump and Wireshark

Appendix - Links

- Links
  - http://www.intel.com (Intel NICs)
  - http://www.ntop.org (PF_RING)
  - http://www.solarflare.com (SolarCapture)
  - http://www.tcpdump.org (TCPdump/Libpcap)
  - http://www.wireshark.org (Wireshark/Dumpcap)
  - http://www.macsales.com (Thunderbolt enclosure)

Appendix – Packet Toaster Specs

- CPU: Intel i5 4570 (3.6GHz quad-core)
- Motherboard: Gigabye Z87N-WIFI
- RAM: 8GB DDR3
- Storage
  - Samsung 840 Evo (Operating System)
  - 2 x Sandisk Extreme in RAID 0 (Capture destination)

Questions