

10Gbit/s Bi-Directional Routing on standard hardware running Linux

Bifrost Workshop 2010
d.27/1-2010
(Talk given at LinuxCom 2009)

by

Jesper Dangaard Brouer <hawk@comx.dk>

Master of Computer Science

Linux Kernel Developer

ComX Networks A/S

The logo for ComX Networks A/S, featuring the word "comx" in a lowercase, sans-serif font. The letters "com" are in a dark grey color, and the letter "x" is in a bright orange color.

ComX Networks A/S

Who am I

- Name: Jesper Dangaard Brouer
 - Edu: Computer Science for Uni. Copenhagen
 - Focus on Network, Dist. sys and OS
 - Linux user since 1996, professional since 1998
 - Sysadm, Developer, Embedded
 - OpenSource projects
 - Author of
 - ADSL-optimizer
 - CPAN IPTables::libiptc
 - Patches accepted into
 - Kernel, iproute2, iptables and wireshark

Presentation overview

- When you leave this presentation, you will know:
 - The Linux Network stack scales with the number of CPUs
 - About PCI-express overhead and bandwidth
 - Know what hardware to choose
 - If its possible to do 10Gbit/s bidirectional routing on Linux?

How many think is possible to do:
10Gbit/s bidirectional routing on Linux?

ComX Networks A/S

- I work for ComX Networks A/S
 - Danish Fiber Broadband Provider (TV, IPTV, VoIP, Internet)
- This talk is about
 - our experiments with 10GbE routing on Linux.
- Our **motivation**:
 - Primary budget/money (in these finance crisis times)
 - **Linux solution: factor 10 cheaper!**
 - (60K USD -> 6K USD)
 - Need to upgrade capacity in backbone edges
 - Personal: Annoyed with bug on Foundry and tech support

Performance Target

- Usage "normal" Internet router
 - 2 port 10 Gbit/s router
 - Bidirectional traffic
 - Implying:
 - 40Gbit/s through the interfaces (5000 MB/s)
 - Internet packet size distribution
 - No jumbo frame "cheats"
 - Not only max MTU (1500 bytes)
 - Stability and robustness
 - Must survive DoS attack with small packet

Compete: 10Gbit/s routing level

- Enabling hardware factors
 - PCI-express: Is a key enabling factor!
 - Giving us enormous “backplane” capacity
 - PCIe x16 gen.2 marketing numbers 160 Gbit/s
 - one-way 80Gbit/s → encoding 64Gbit/s → overhead ~54 Gbit/s
 - Scaling: NICs with multiple RX/TX queues in hardware
 - Makes us scale beyond one CPU
 - Large effort in software network stack (thanks DaveM!)
 - Memory bandwidth
 - Target 40Gbit/s → 5000 MBytes/s

PCI-express: Overhead

- Encoding overhead: 20% (8b/10b encoding)
 - PCIe gen 1. 2.5Gbit/s per lane → 2 Gbit/s
- Generation 1 vs. gen.2:
 - Double bandwidth, 2 Gbit/s → 4 Gbit/s
- Protocol overhead: Packet based
 - Overhead per packet: 20 Bytes (32-bit), 24 bytes (64-bit)
 - MaxPayload 128 bytes => 16 % additional overhead
 - PCIe x8 = 32 Gbit/s
 - MaxPayload 128 bytes => 26.88 Gbit/s
- Details see: <http://download.intel.com/design/intarch/papers/321071.pdf>
Title: “Hardware Level IO Benchmarking of PCI express*”

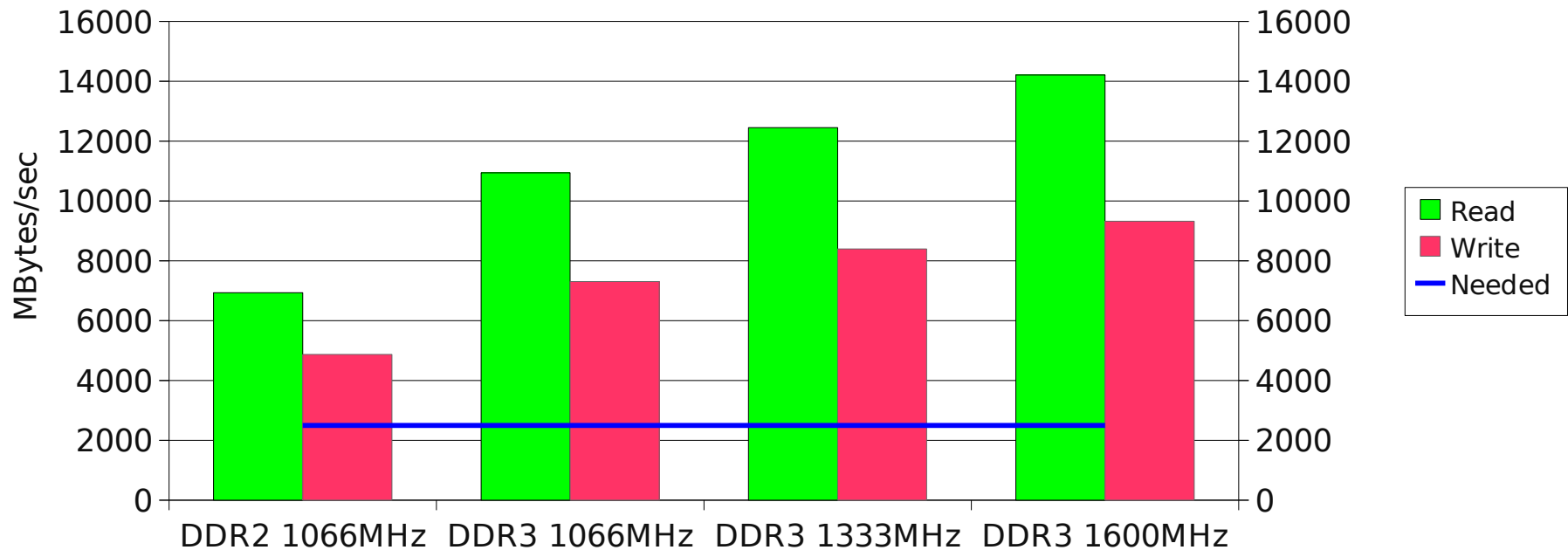
Hardware: Device Under Test (DUT)

- Starting with cheap gaming style hardware
 - No budget, skeptics in the company
 - CPUs: Core i7 (920) vs. Phenom II X4 (940)
 - RAM: DDR3 vs DDR2
- Several PCI-express slots Generation 2.0
 - Motherboards
 - Core i7 – Chipset X58: Asus P6T6 WS Revolution
 - AMD – Chipset 790GX: Gigabyte MA790GP-DS4H

What can we expect from this hardware...

DUT: Memory Bandwidth

Memory bandwidth (Imbench: bw_mem)



- Raw memory bandwidth enough?
 - Memory types: DDR2: Phenom II / DDR3: Core i7
- Target 5000 MBytes/s (40Gbit/s) (2500 MB/s write and 2500MB/s reads)
 - HW should be capable of doing several 10GbE

10Gbit/s Network Interface Cards

- Network Interface Cards (NICs) under test:
 - Sun Neptune (niu): 10GbE Dual port NIC PCIe x8 gen.1 (XFP)
 - PCIe x8 = 16Gbit/s (overhead 16% = 13.44 Gbit/s)
 - SMC Networks (sfc): 10GbE NIC, Solarflare chip (XFP)
 - hardware queue issues, not default enabled
 - only one TX queue, thus cannot parallelize
 - Intel (ixgbe): newest Intel 82599 chip NIC
 - Fastest 10GbE NIC I have ever seen!!!
 - Engineering samples from:
 - Intel: Dual port SFP+ based NIC (PCIe x8 Gen.2)
 - Hotlava Systems Inc.: 6 port SFP+ based NIC (PCIe x16 Gen.2)

Preliminary: Bulk throughput

- Establish: enough PCIe and Memory bandwidth?
 - Target: 2 port 10GbE bidir routing
 - collective 40Gbit/s
 - with packet size 1500 bytes (MTU)

Answer: **Yes, but only with special hardware:**

CPU Core i7

Intel 82599 based NIC

DDR3 Memory minimum at 1333 MHz

QuickPath Interconnect (QPI) tuned to 6.4GT/s (default 4.8GT/s)

Observations: AMD: Phenom II

- AMD Phenom(tm) II X4 940 (AM2+)
 - Can do 10Gbit/s one-way routing
 - Cannot do bidirectional 10Gbit/s
- Memory bandwidth should be enough
 - Write 20Gbit/s and Read 20Gbit/s (2500MB/s)
- 1800 Mhz HyperTransport seems too slow
 - HT 1800Mhz ~ bandwidth 57.6 Gbit/s
 - "Under-clocking" HT: performance followed
 - Theory: Latency issue
 - PCIe to memory latency too high, outstanding packets

Scaling with the number of CPUs

- To achieve these results
 - distribute the load across CPUs
 - A single CPU cannot handle 10GbE
- Enabling factor: “multiqueue”
 - NICs with multiple hardware RX/TX queues
 - **Seperate IRQ per queue** (both RX and TX)
 - Lots of IRQs used
 - look in /proc/interrupts eg. ethX-rx-2

Linux Multiqueue Networking

- RX path: NIC computes hash
 - Also known as RSS (Receive-Side Scaling)
 - Bind flows to queue, avoid out-of-order packets
- Large effort in software network stack
 - TX qdisc API "hack", backward compatible
 - http://vger.kernel.org/~davem/davem_nyc09.pdf
 - Beware: Bandwidth shapers break CPU scaling!

Linux Network stack scales with the number of CPUs

Practical: Assign HW queue to CPUs

- Each HW (RX or TX) queue has individual IRQ
 - Look in `/proc/interrupts`
 - A naming scheme: `ethXX-rx-0`
- Assign via "smp_affinity" mask
 - In `/proc/irq/nn/smp_affinity`
 - Trick: `/proc/irq/*/eth31-rx-0/../../smp_affinity`
 - Trick: `"grep . /proc/irq/*/eth31-*x-*/../../smp_affinity"`
- Use tool: `'mpstat -A -P ALL'`
 - see if the interrupts are equally shared across CPUs

Binding RX to TX: Stay on same CPU

- RX to TX queue mapping: tied to the same CPU
 - Avoid/minimize cache misses, consider NUMA
- 3 use-cases for staying on the same CPU:
 - Forwarding (main focus) (RX to TX other NIC)
 - How: Record queue number at RX and use it at TX
 - Kernel 2.6.30 for proper RX to TX mapping
 - Server (RX to TX)
 - Caching of socket info (Credit to: Eric Dumazet)
 - Client (TX to RX)
 - Hard, Flow "director" in 10GbE Intel 82599 NIC

Start on results

- Lets start to look at the results...
 - Have revealed
 - Large frames: 10GbE bidirectional was possible!
 - What about smaller frames? (you should ask...)
 - First need to look at:
 - Equipment
 - Tuning
 - NICs and wiring

Test setup: Equipment

- Router: Core i7 920 (default 2.66 GHz)
 - RAM: DDR3 (PC3-12800) 1600MHz X.M.P settings
 - Patriot: DDR3 Viper Series Tri-Channel
 - Low Latency CAS 8-8-8-24
 - QPI at 6.4 GT/s (due to X.M.P)
- Generator#1: AMD Phenom 9950 quad
- Generator#2: AMD Phenom II X4 940

Test setup: Tuning

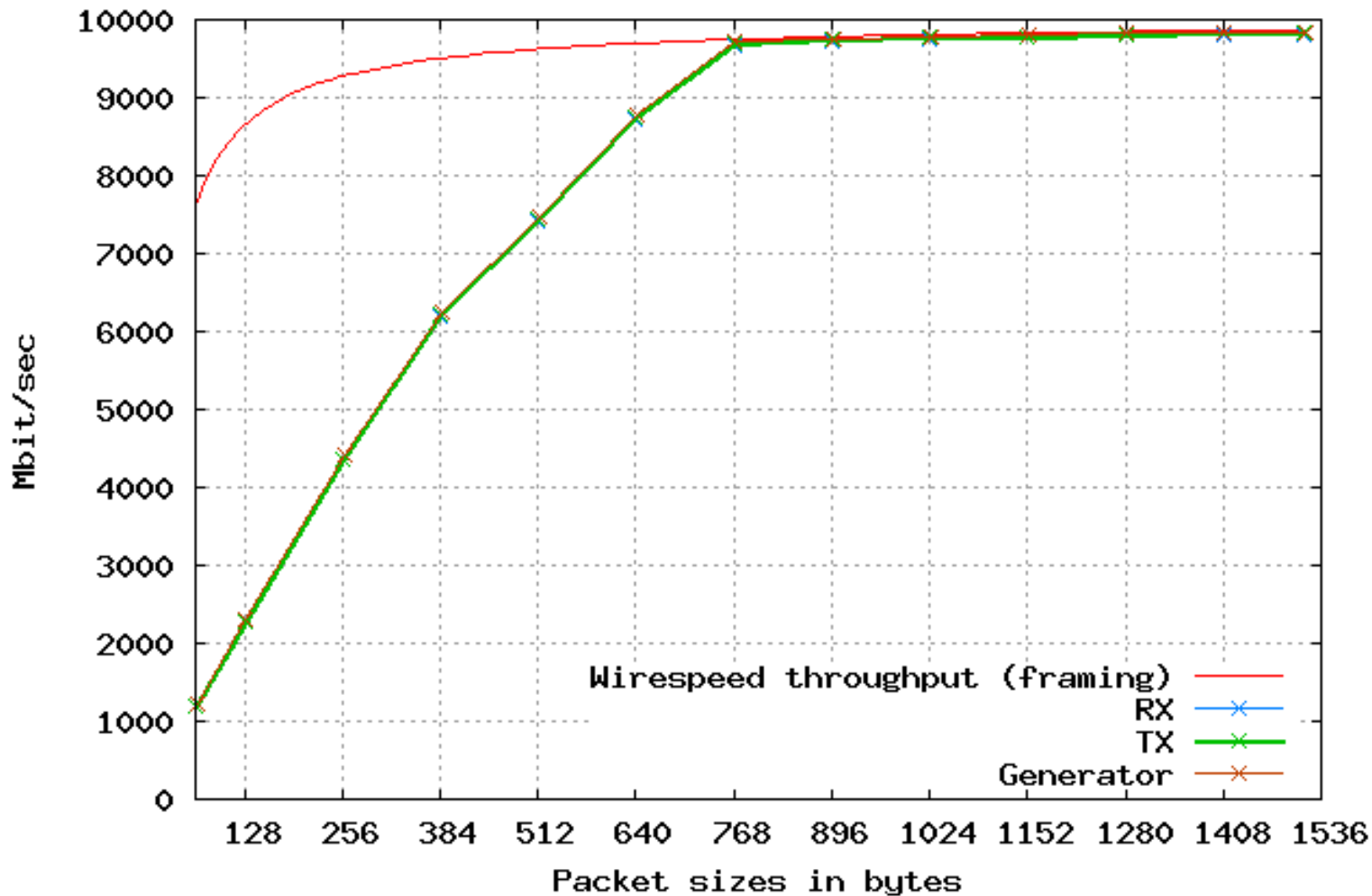
- Binding RX to TX queues
- Intel NIC tuning
 - Adjust interrupt mitigation parameters `rx-usecs` to 512
 - Avoiding interrupt storms (at small packet sizes)
 - `'ethtool -C eth31 rx-usecs 512'`
 - Ethernet flow control (pause frames)
 - Turn-off during tests:
 - To see effects of overloading the system
 - `'ethtool -A eth31 rx off tx off'`
 - Recommend turning on for production usage

NIC test setup #1

- Router
 - Equipped with: Intel 82599 Dual port NICs
 - Kernel: 2.6.31-rc1 (net-next-2.6 tree [8e321c4](#))
- Generators
 - Equipped with NICs connected to router
 - Sun Neptune (niu)
 - SMC (10GPCIe-XFP) solarflare (sfc)
 - Using: pktgen
 - UDP packets
 - Randomize dst port number – utilize multiple RX queue

Setup #1: 10GbE uni-directional

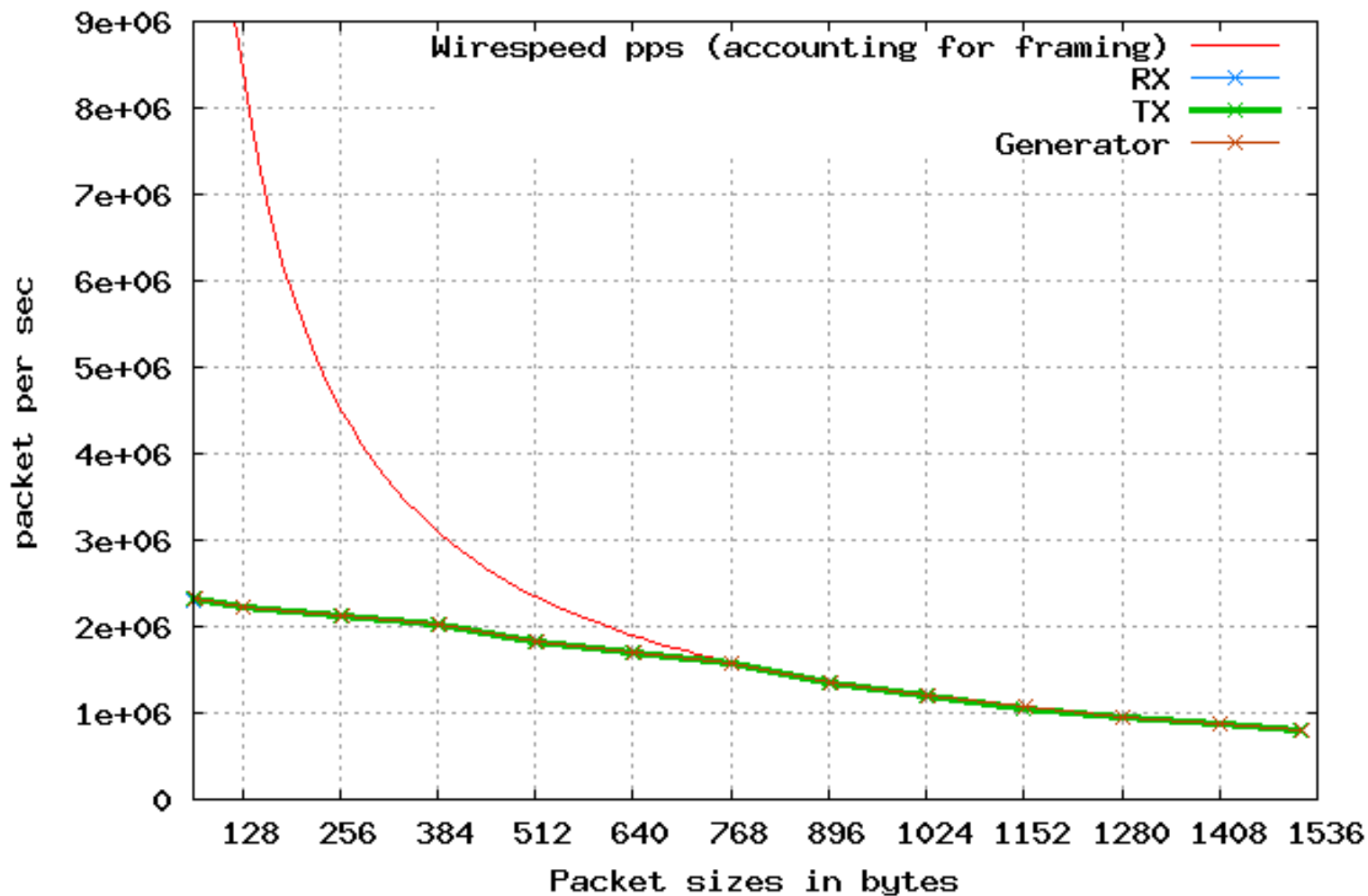
10Gbit/s routing (uni-directional)
Throughput vs Packets size
(generator too slow)



- Be skeptic
 - Packet generator
 - Too slow!
 - Sun NICs

Setup #1: Packet Per Sec

10Gbit/s routing (uni-directional)
Packets per sec vs. Packet size
(generator too slow)



Be skeptic

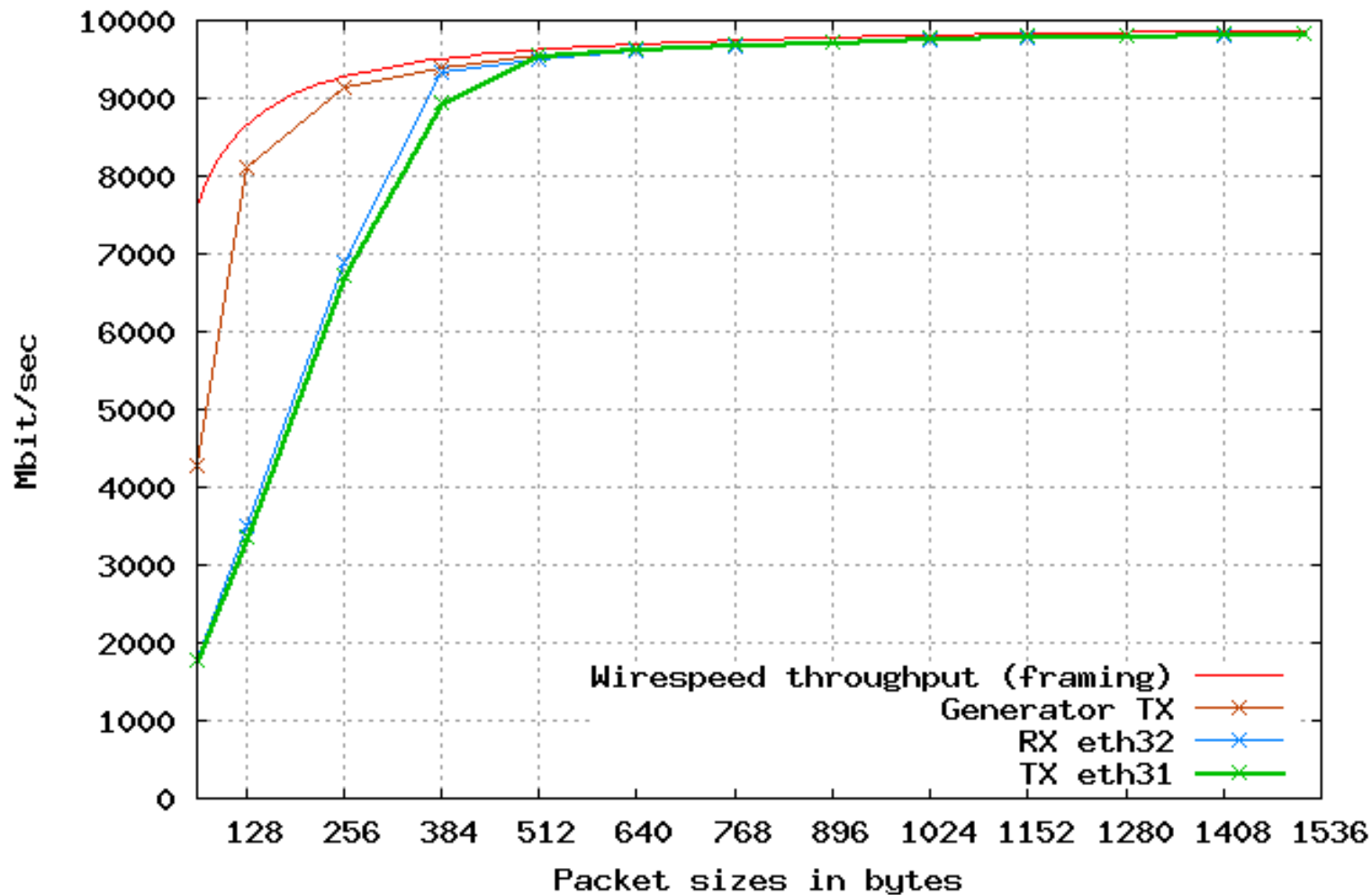
- Packet generator
 - Too slow!
 - Sun NICs
 - PPS limit

NIC test setup #2

- Router and Generators
 - All equipped with:
 - Intel 82599 based NICs
- Pktgen limits
 - Sun NICs max at 2.5 Mpps
 - Intel 82599 NIC (at packet size 64 byte)
 - 8.4 Mpps with AMD CPU
 - 11 Mpps with Core i7 CPU as generator

Setup #2: 10GbE uni-dir throughput

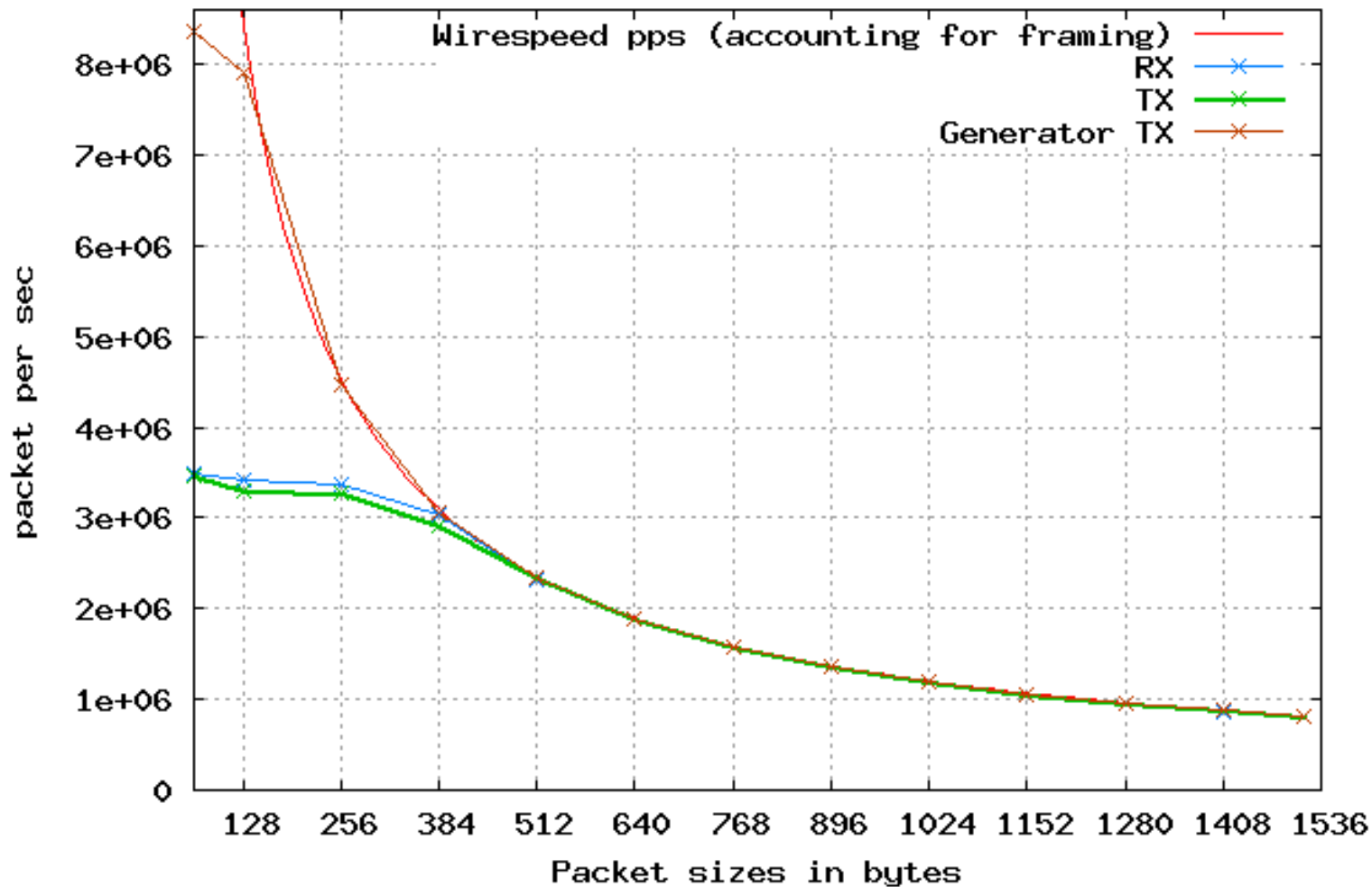
10Gbit/s routing (uni-directional)
Throughput vs Packets size
(try overload, no flow control)



- Wirespeed 10GbE
 - uni-dir routing
 - pktsize 420

Setup #2: 10GbE uni-dir PPS

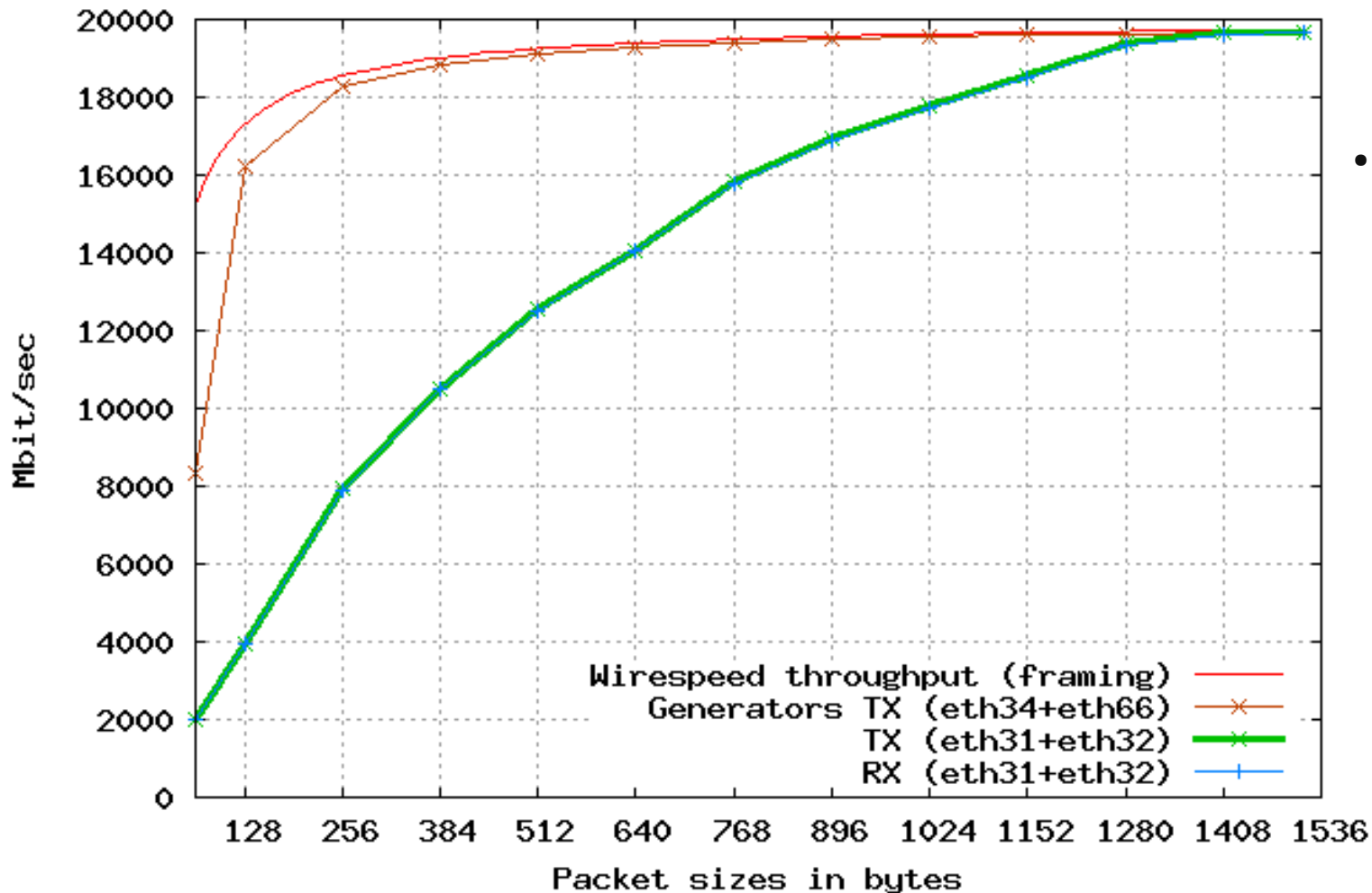
10Gbit/s routing (uni-directional)
Packets per sec vs. Packet size
(no flow control)



- Limits
 - Packet Per Sec
 - 3.5 Mpps
 - at 64bytes

10GbE Bi-directional routing

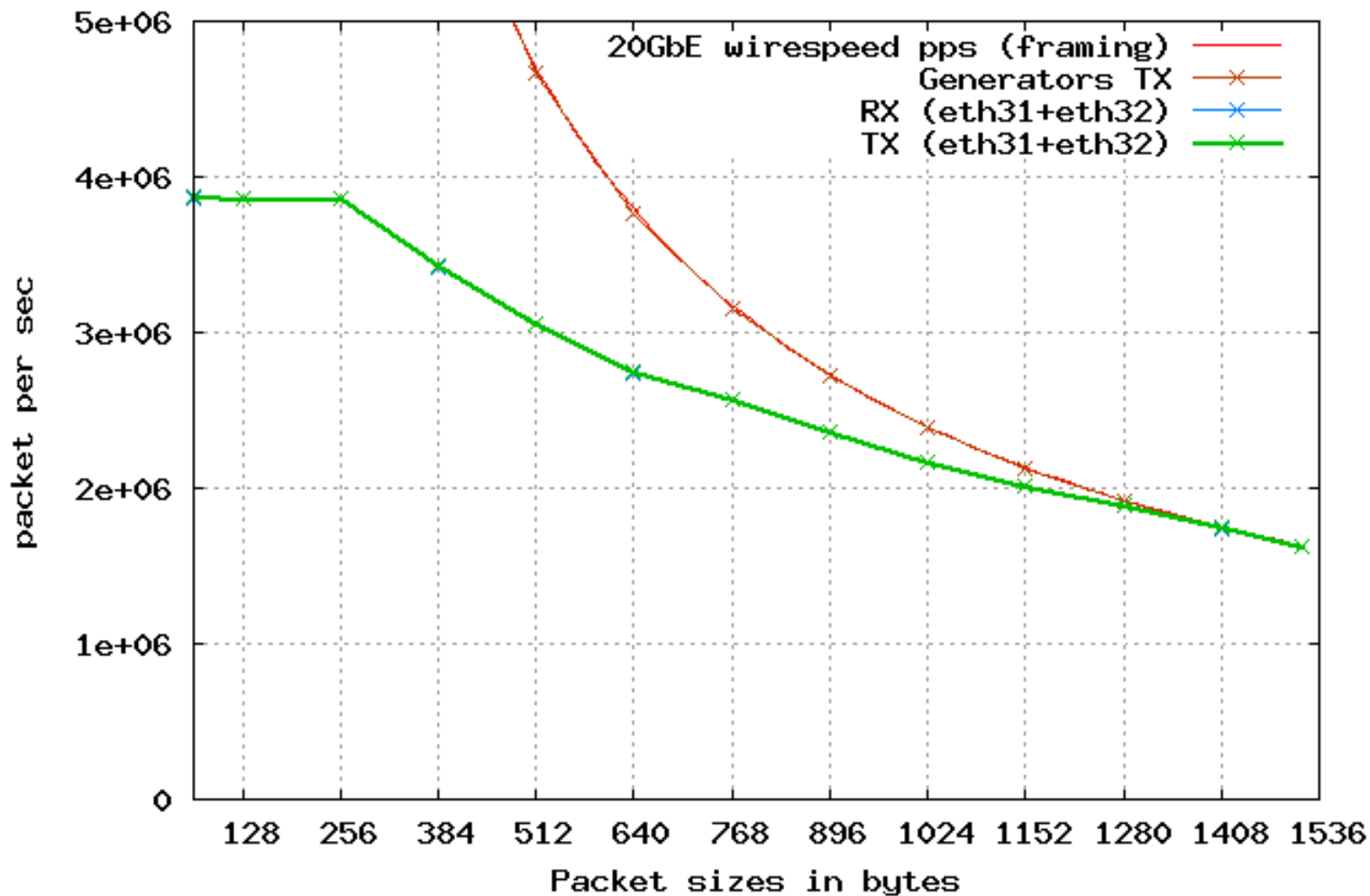
2 port 10Gbit/s routing (bi-directional)
Throughput vs Packets size
(trying to overload, no flow control)



- Wirespeed at
 - 1514 and 1408
 - 1280 almost
- Fast generators
 - can survive load!

10GbE Bi-dir: Packets Per Sec

10Gbit/s routing (bi-directional)
Packets per sec vs. Packet size
(overload, no flow control)



- Limited
 - by PPS rate
 - Max at
 - 3.8 Mpps

Simulate: Internet traffic pattern

- Target: Internet router
 - Simulate Internet traffic with pktgen
 - Based on Robert Olssons Article:
 - "Open-source routing at 10Gb/s"
 - Packet size distribution

pktsize	distribution	approximate
64	45.00%	50.00%
576	25.00%	25.00%
1514	30.00%	25.00%

avg. size **627 bytes** **554 bytes**
 - Large number of flows
 - 8192 flows, duration 30 pkts, destinations /8 prefix (16M)
 - watch out for:
 - slow-path route lookups/sec
 - size of route cache

Uni-dir: Internet traffic pattern

- Simulate Internet Traffic
 - 10GbE uni-directional

	Gbit/s	Mpps
Generator	9.5	2.17
RX-router	9.4	2.14
TX-router	9.4	2.14

New route lookups/s: 68k/sec

- Route Cache is scaling
 - 900k entries in route cache
 - very little performance impact
 - almost same as const size packet forwarding

Bi-Dir: Internet traffic pattern

- 10GbE bi-directional

	Gbit/s	Mpps	avg. pkt
Generator	19.0 Gbit/s	4.3	554
RX	11.6 Gbit/s	3.4	423
TX	11.6 Gbit/s	3.4	423

New route lookups/s: 140k/sec

Comparing with constant size packet tests

Size 423	11.4 Gbit/s	3.3	423
Size 554	15.6 Gbit/s	3.5	554

- Strange: Average packet size in test
 - Generators: 554 bytes vs. Receive size: 423 bytes
- Route Cache seems to scale (1.2M entries)
 - when comparing to const size packet tests

Summary: Target goal reached?

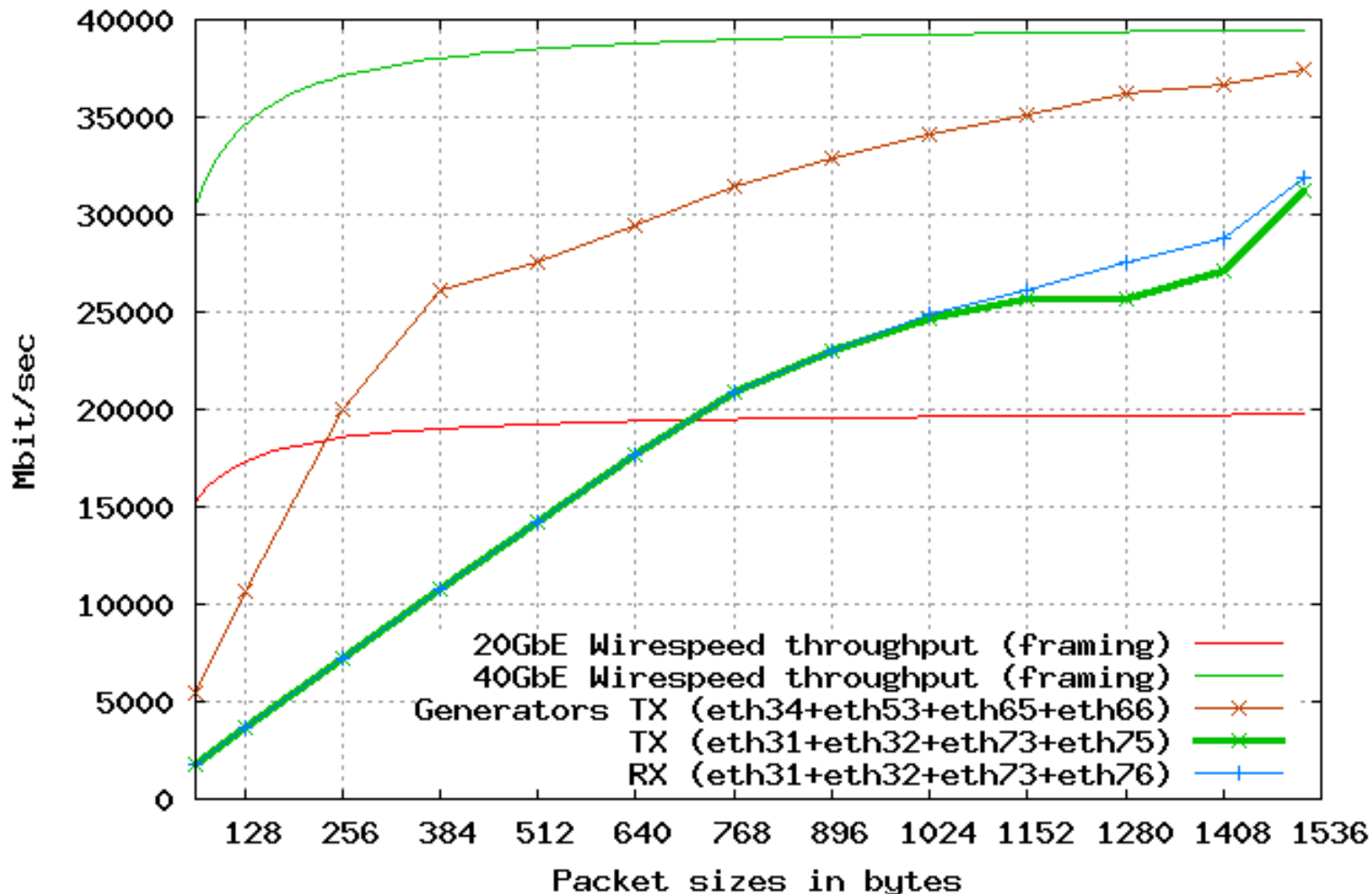
- 2 port 10GbE “Internet” router
 - Uni-Dir: Very impressive, Wirespeed with small pkts
 - Bi-dir: Wirespeed for 3 largest packet sizes
 - Good curve, no choking
 - Bandwidth: Well within our expected traffic load
 - Internet type traffic
 - Uni-dir: Impressive
 - Bi-dir: Follow pkt size graph, route cache scales
 - Traffic Overloading / semi-DoS
 - Nice graphs, doesn't choke with small pkts

Yes! - It is possible to do:

10Gbit/s bidirectional routing on Linux

Bandwidth for 4 x 10GbE bidir ?

4 x 10Gbit/s routing (bi-directional)
Throughput vs Packets size
(limited data, slow generators)

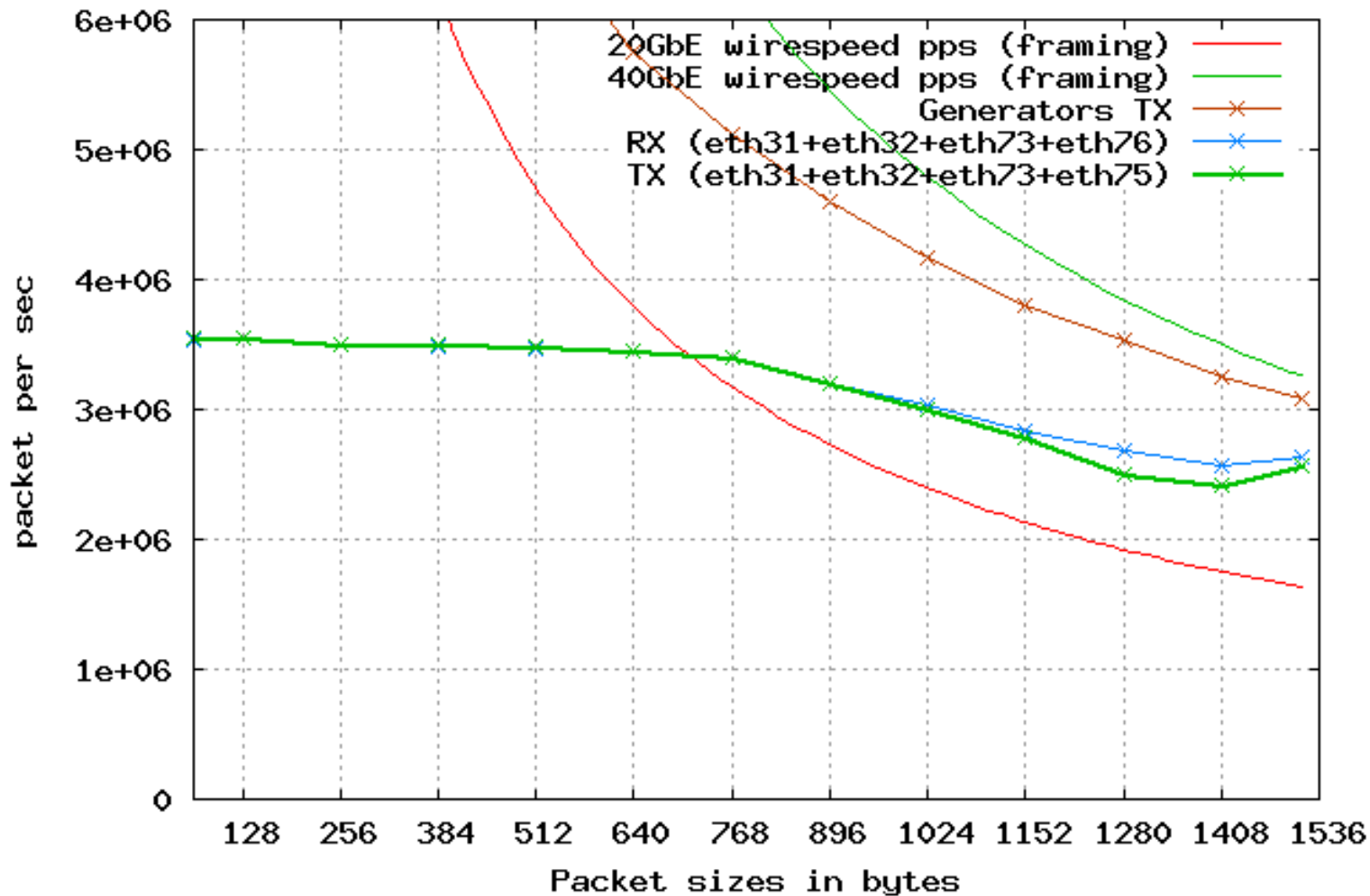


Beyond 20GbE

- Pktsize: 1514
 - Tx: 31.2 Gb/s
 - Rx: 31.9 Gb/s
- Enough mem bandwidth
- > 2x 10GbE bidir
- Real limit: PPS

4 x 10GbE bidir: Packets Per Sec

4 x 10Gbit/s routing (bi-directional)
Packets per sec vs. Packet size
(overload, no flow control)



Real limit

- PPS limits

Summary: Lessons learned

- Linux Network stack scales
 - “multiqueue” framework works!
- Hardware
 - Choosing the right hardware essential
- PCI-express latency is important
 - Choose the right chipset
 - Watch out for interconnecting of PCIe switches
- Packets Per Second
 - *is the real limit*, not bandwidth

Future

- Buy server grade hardware
 - CPU Core i7 → Xeon 55xx
 - Need min Xeon 5550 (QPI 6.4GT/s, RAM 1333Mhz)
 - Two physical CPUs, NUMA challenges
- Smarter usage of HW queues
 - Assure QoS by assigning Real-time traffic to HW queue
 - ComX example: IPTV multicast streaming
- Features affecting performance?
 - Title: “How Fast Is Linux Networking”
 - Stephen Hemminger, Vyatta
 - Japan Linux Symposium, Tokyo (22/10-2009)

The End

Thanks!

Engineering samples:

Intel Corporation

Hotlava Systems Inc.

SMC Networks Inc.



Famous Top Linux Kernel Comitter
David S. Miller

10G optics too expensive!

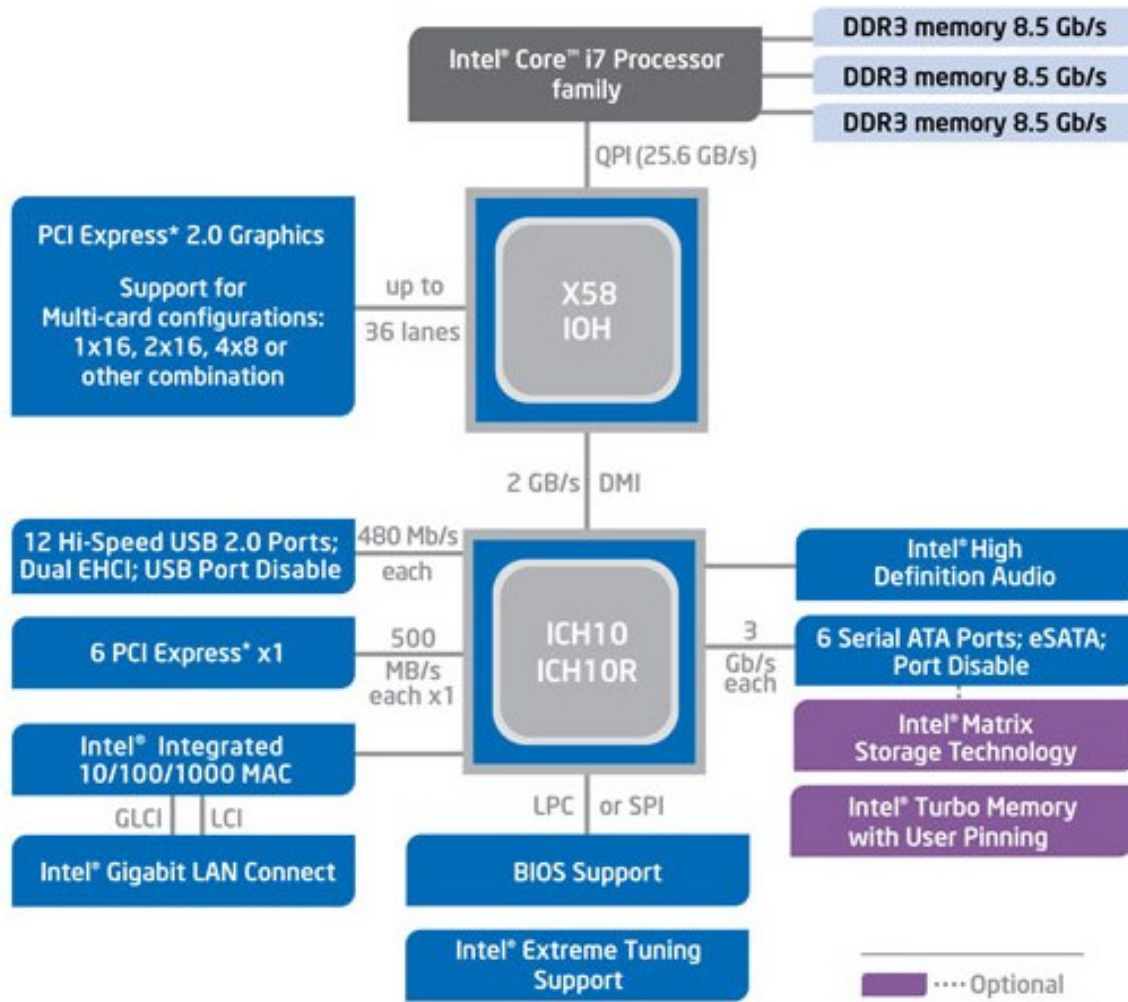
- 10GbE SFP+ and XFP optics very expensive
 - *There is a cheaper alternative!*
 - Direct Attached Cables
 - SFP+ LR optics price: 450 USD (need two)
 - SFP+ Copper cable price: 40 USD
 - Tested cable from:
 - Methode dataMate
 - <http://www.methodedatamate.com>



Pitfalls: Bad motherboard PCIe design

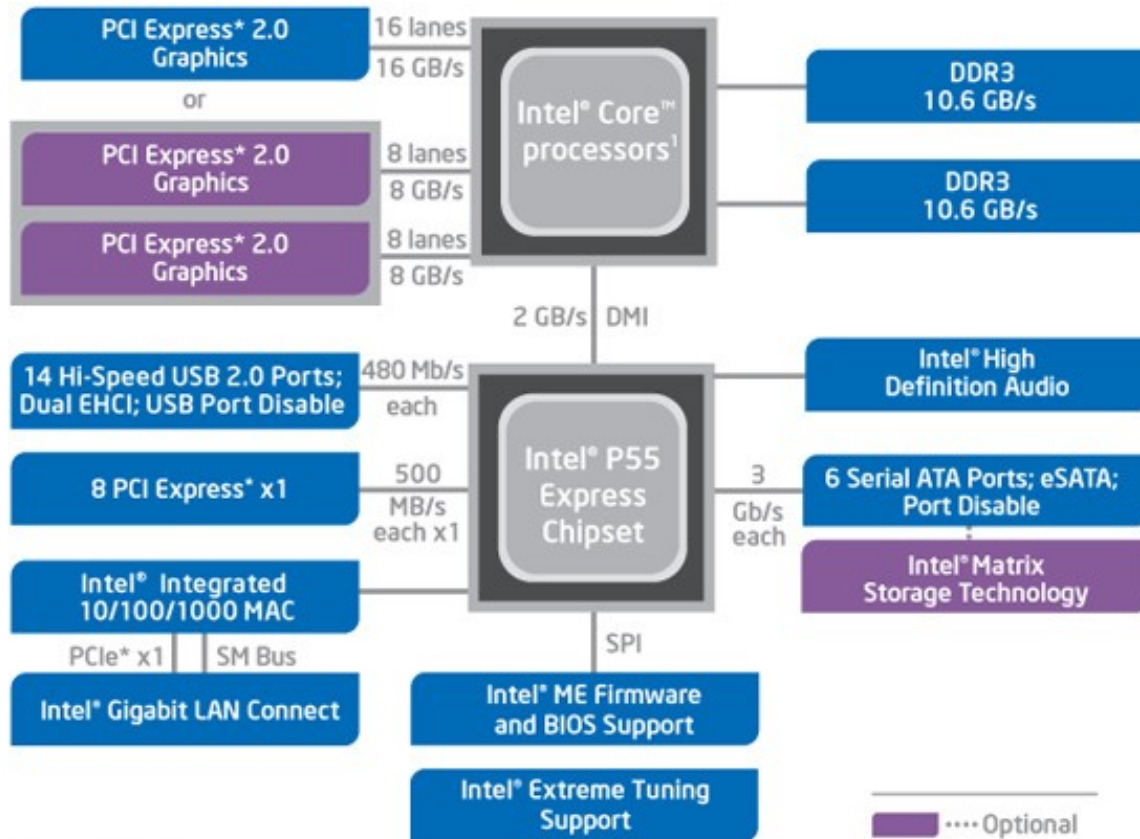
- Reason: Could not get beyond 2x 10GbE
- Motherboard: Asus P6T6 WS revolution
 - Two PCIe switches
 - X58 plus NVIDIA's NF200
 - NF200 connected via PCIe gen.1 x16
 - 32 Gbit/s -> overhead 26.88 Gbit/s
 - Avoid using the NF200 slots
 - scaling again...

Chipset X58: Core i7



- Bloomfield, Nehalem
- Socket: LGA-1366

Chipset P55: Core i5 and i7-800



- Lynnfield, Nehalem
- Socket: LGA-1156