



Attribution-Noncommercial-Share Alike 3.0 Unported

You are free:



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work

Under the following conditions:



Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).



Noncommercial. You may not use this work for commercial purposes.



Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

Attribution shall include Bengt Gördén, Olof Hagsand and Robert Olsson

Towards 10Gb/s open-source routing

Olof Hagsand (KTH)
Robert Olsson (UU)
Bengt Gördén (KTH)
FSCONS 2008

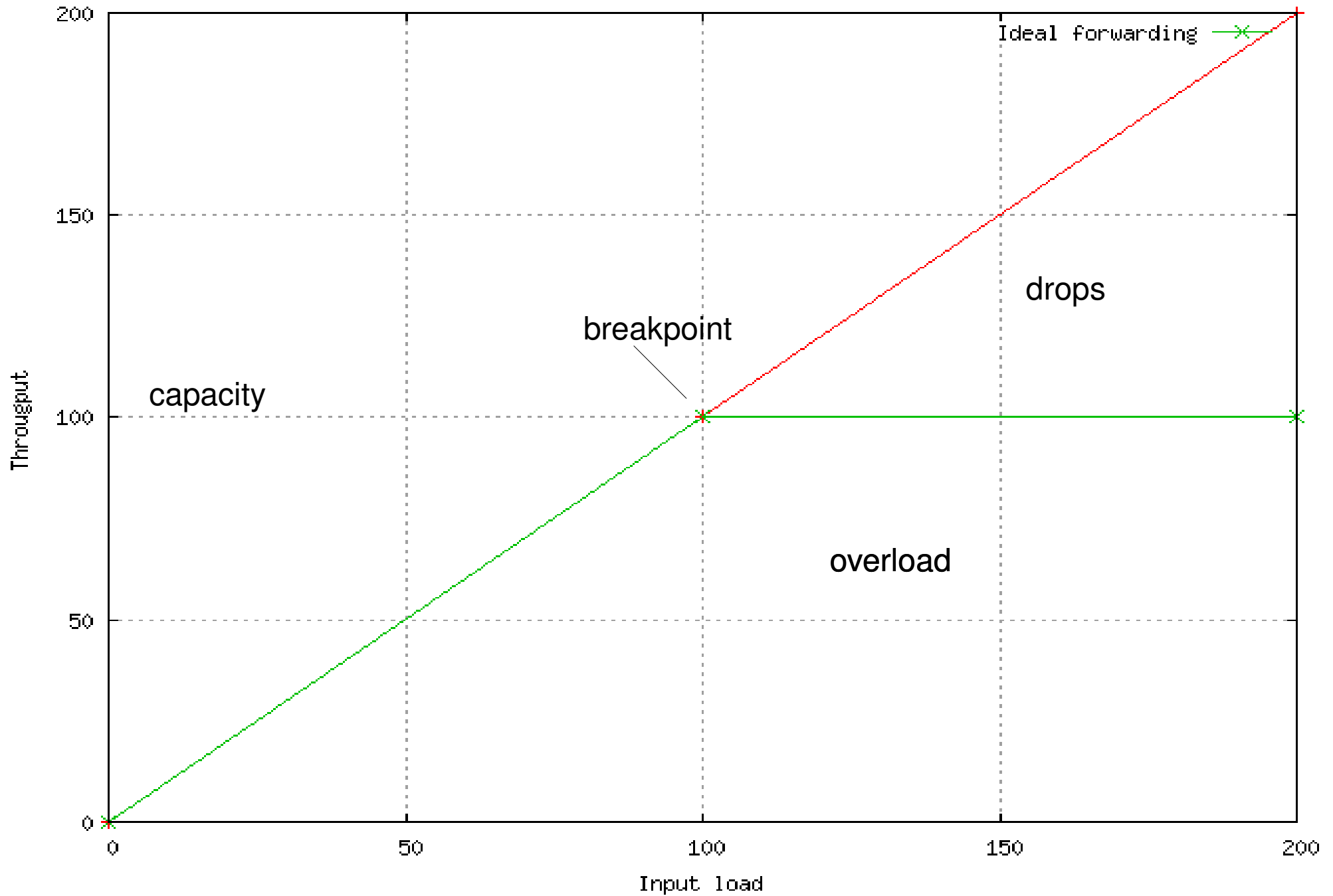
Introduction

- Investigate packet forwarding performance of new PC hardware:
 - Multi-core CPUs
 - Multiple PCI-e buses
 - 10G NICs
- Can we obtain enough performance to use open-source routing also in the 10Gb/s realm?

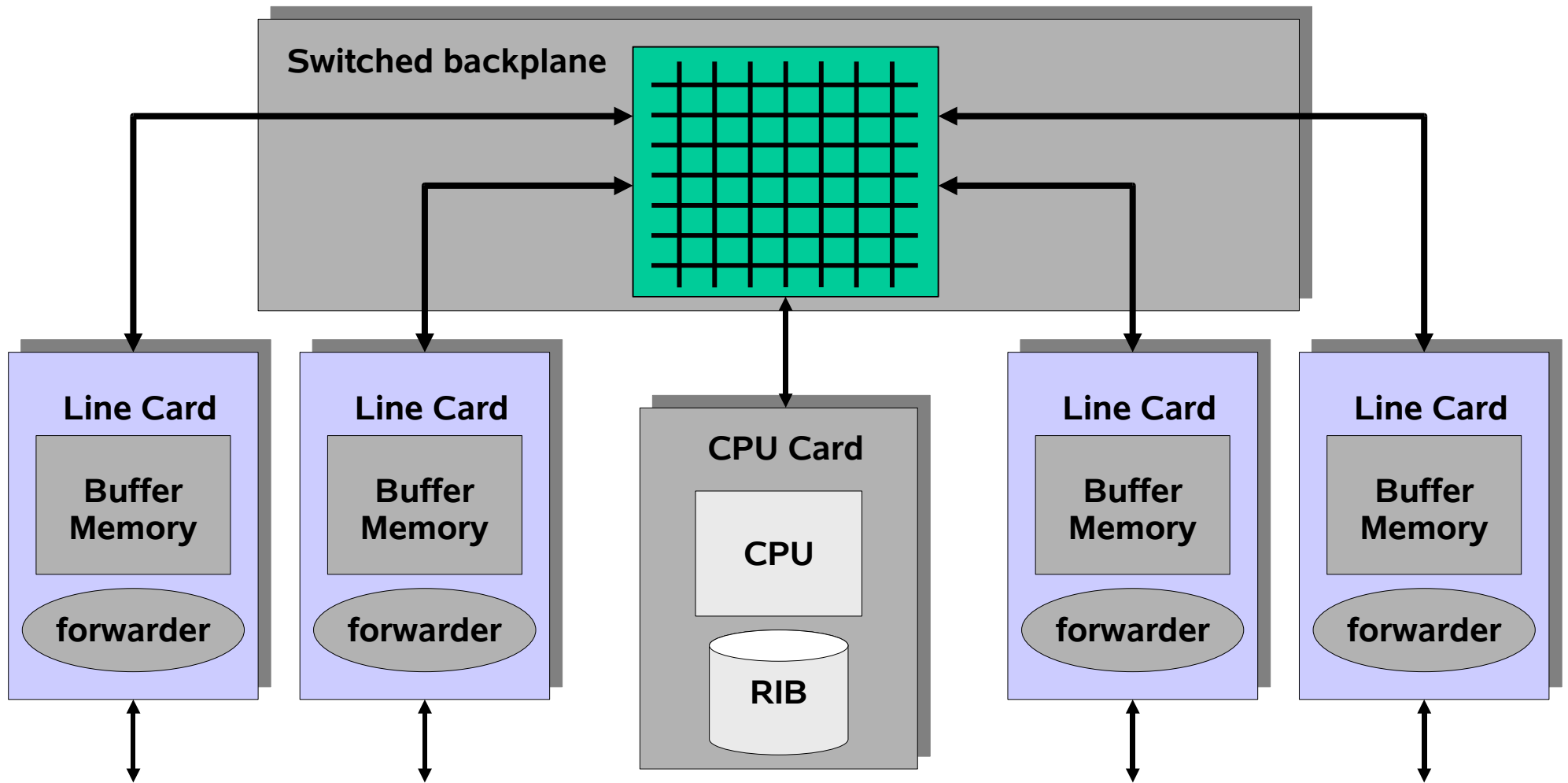
Measuring throughput

- Packet per second
 - Per-packet costs
 - CPU processing, I/O and memory latency, clock frequency
- Bandwidth
 - Per-byte costs
 - Bandwidth limitations of bus and memory

Measuring throughput

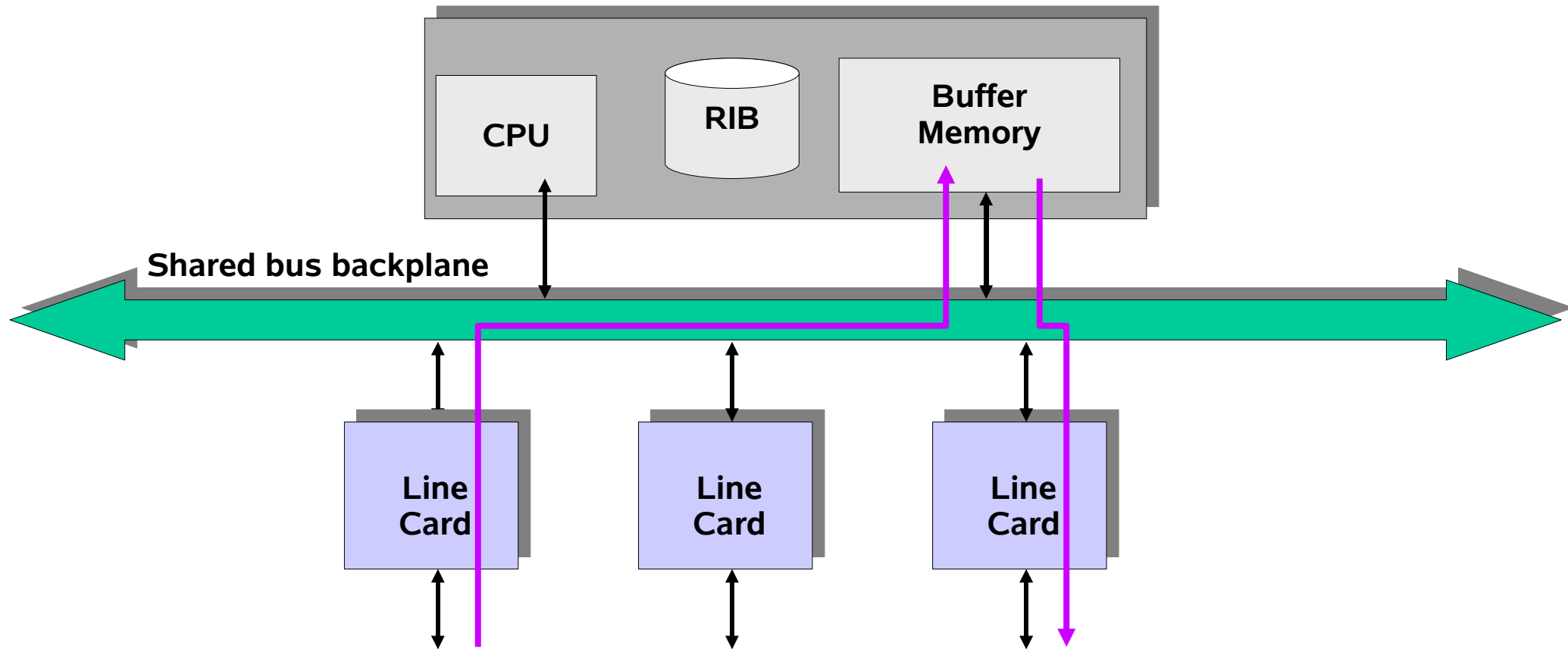


Inside a router, HW style



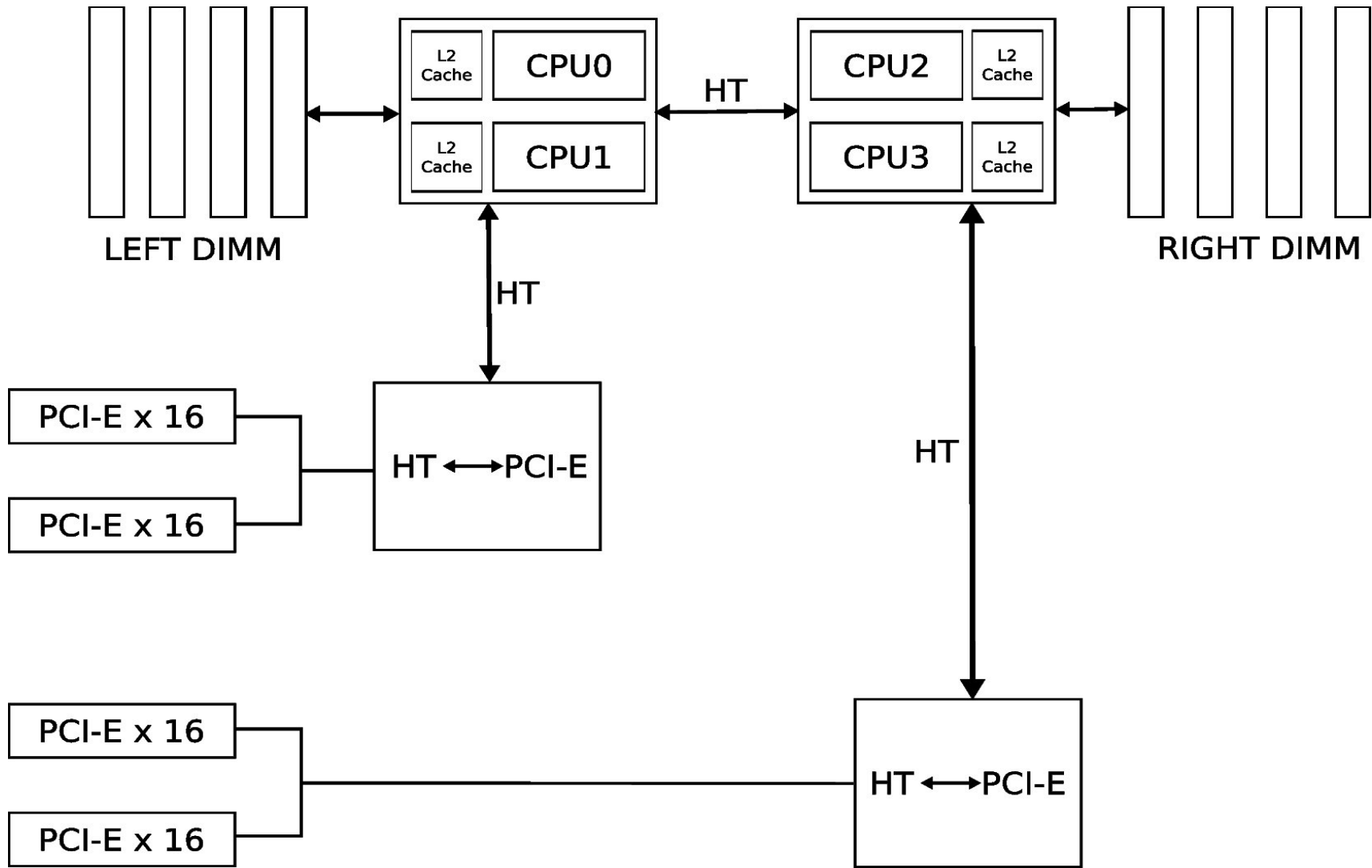
Specialized hardware: ASICs, NPUs, backplane with switching stages or crossbars

Inside a router, PC-style



- Every packet goes twice over shared bus to the CPU
- Cheap, but low performance
- But lets increase the # of CPUs and # of buses!

Block hw structure (set 1)

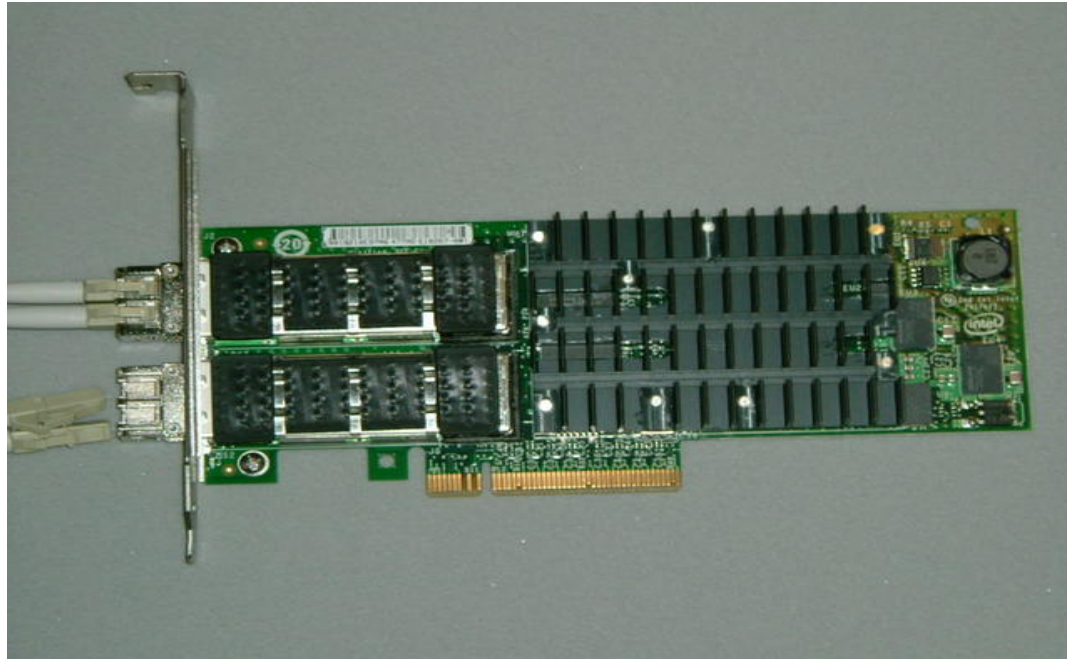


Hardware – Box (set 2)



AMD Opteron 2356 with one quad core 2.3GHz Barcelona CPUs on a TYAN 2927 Motherboard (2U)

Hardware - NIC



Intel 10g board Chipset 82598

Open chip specs. Thanks Intel!

Lab



creativecommons.org/licenses/by-nc-sa/3.0/

Equipment summary

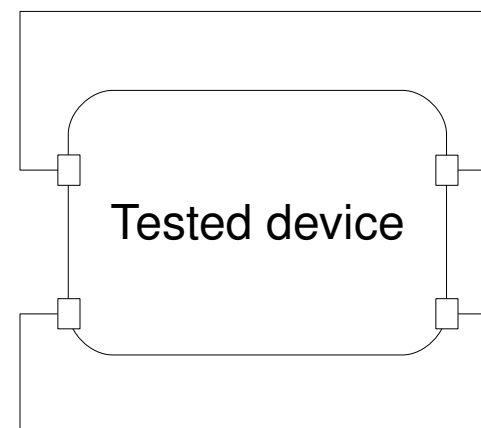
- Hardware needs to be carefully selected
- BifrostLinux on kernel 2.6.24rc7 with LC-trie forwarding
- Tweaked pktgen
- Set 1: AMD Opteron 2222 with two double core 3GHz CPUs on a Tyan Thunder n6650W(S2915) motherboard
- Set 2: AMD Opteron 2356 with one quad core 2.3GHz Barcelona CPUs on a TYAN 2927 Motherboard (2U)
- Dual PCIe buses
- 10GE network interface cards.
 - PCI Express x8 lanes based on Intel 82598 chipset

Experiments

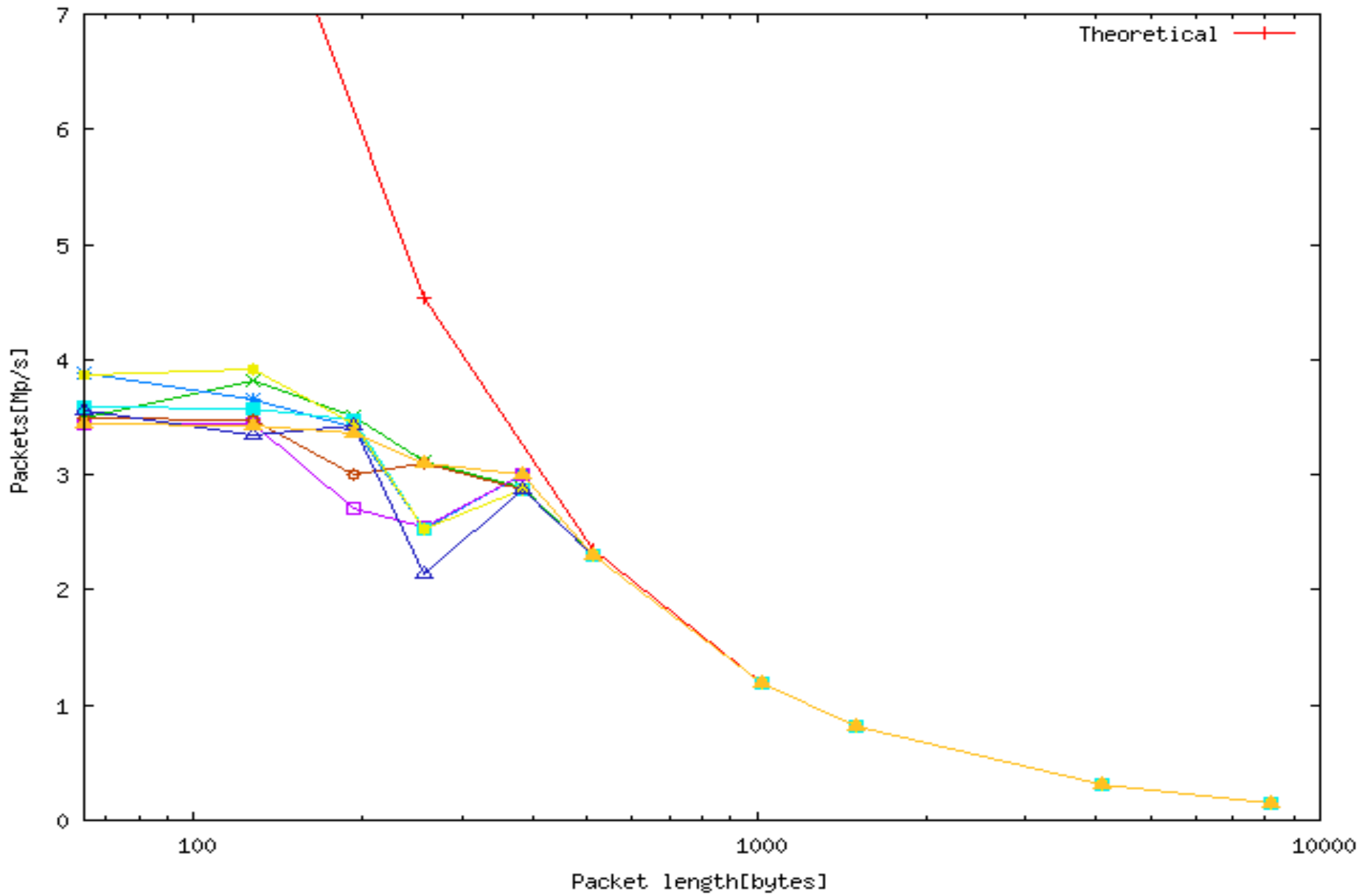
- Transmission(TX)
 - Upper limits on (hw) platform
- Forwarding experiments
 - Realistic forwarding performance

Tx Experiments

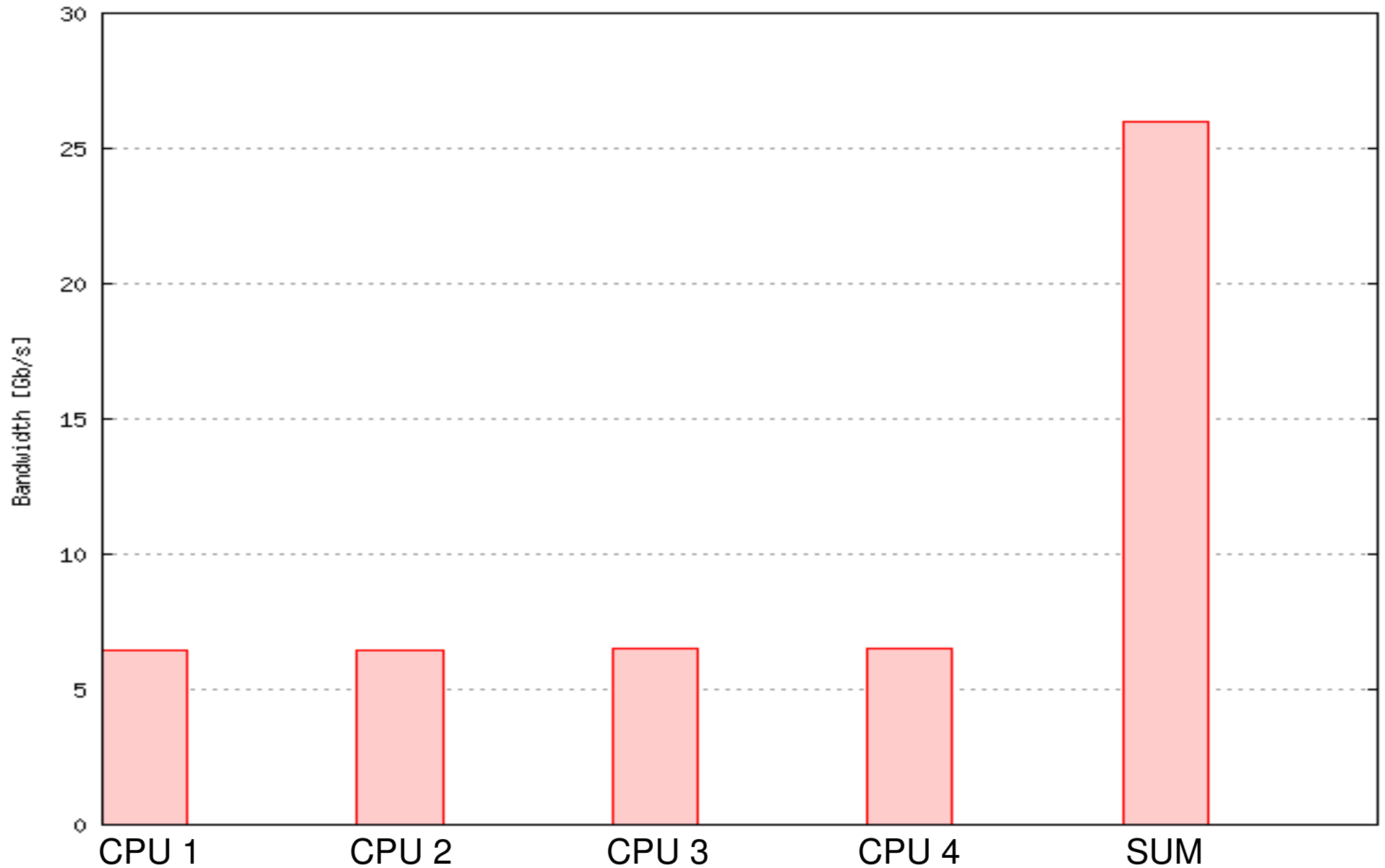
- Goal:
 - Just to see how much the hw can handle – upper limit
- Loopback tests over fibers
- Don't process RX packets just let MAC count them
- These numbers can give indication what forwarding capacity is possible
- Experiments:
 - Single CPU TX single interface
 - Four CPUs TX one interface each



Tx single sender: Packets per second



Tx - Four CPUs: Bandwidth



creativecommons.org/licenses/by-nc-sa/3.0/

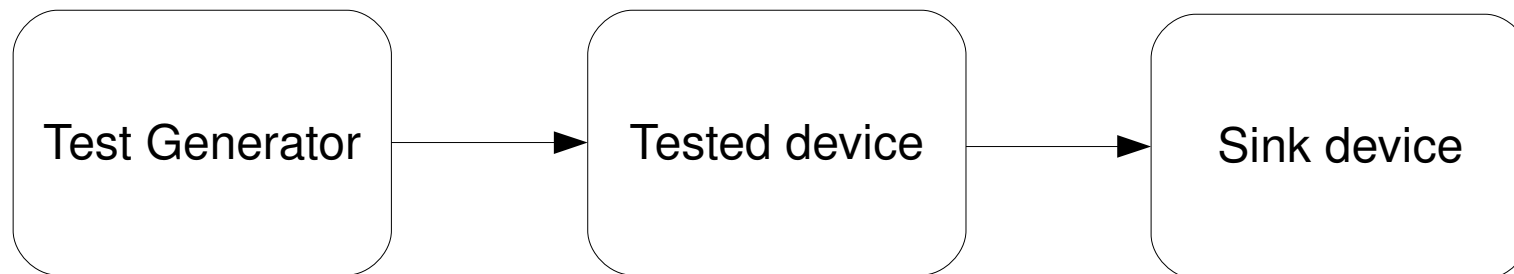
Packet length: 1500 bytes

TX experiments summary

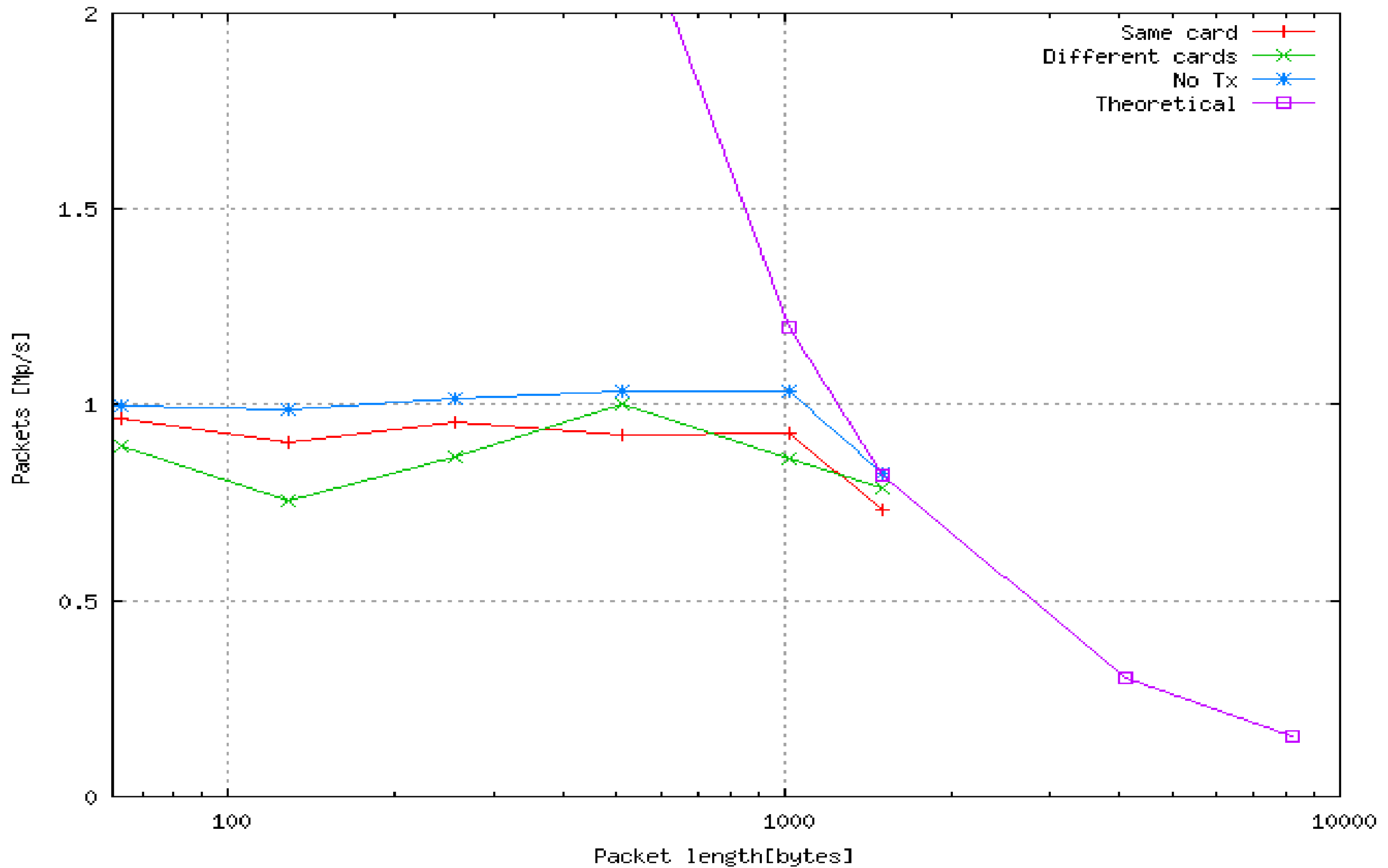
- Single Tx sender is primarily limited by PPS at around 3.5Mpps
- A bandwidth of 25.8 Gb/s and a packet rate of 10 Mp/s using four CPU cores and two PCIe buses
- This shows that the hw itself allows 10Gb/s performance
- We also see nice symmetric Tx between the CPU cores.

Forwarding experiments

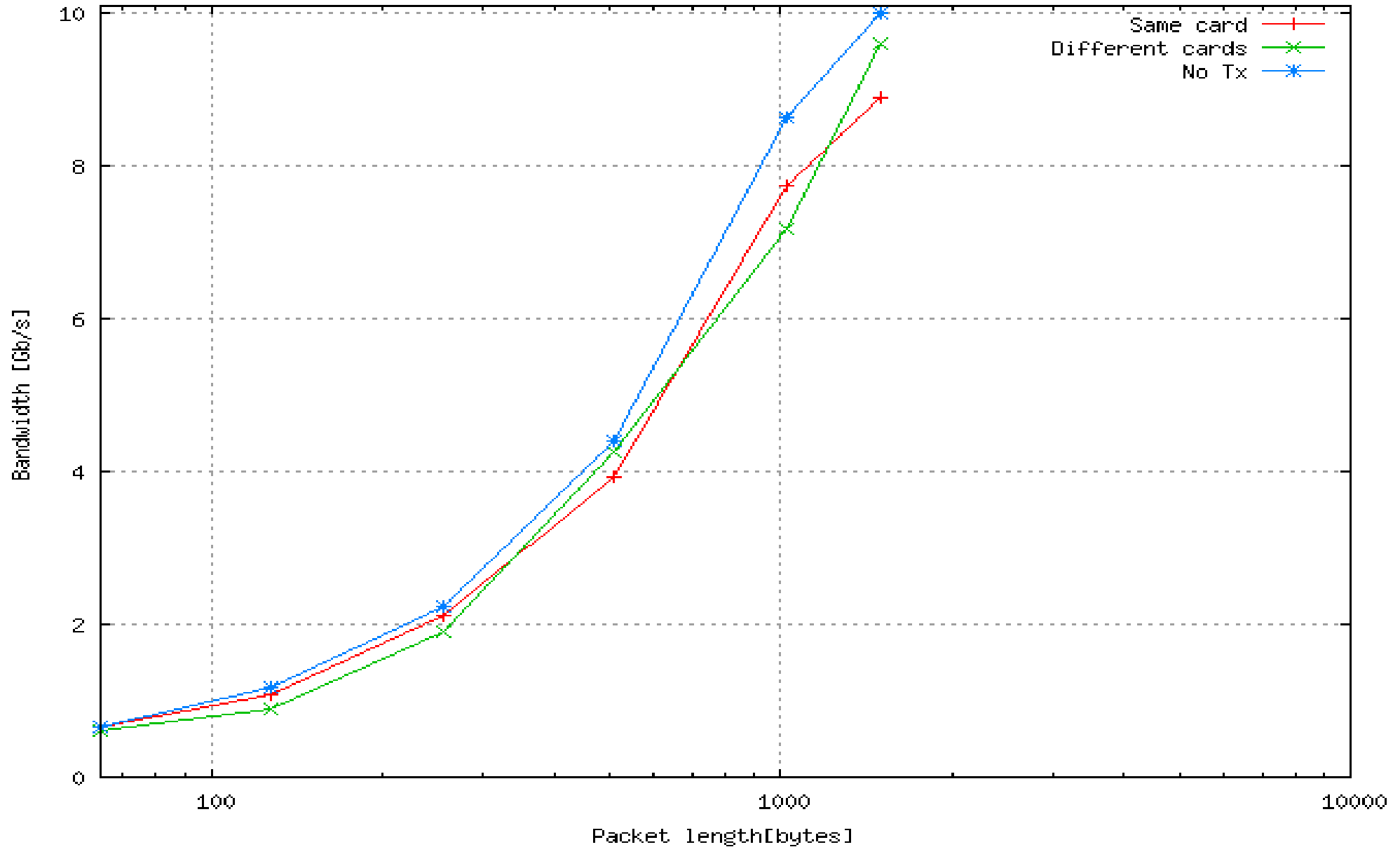
- Goal:
 - Realistic forwarding performance
- Overload measurements (packets are lost)
- Single forwarding path from one traffic source to one traffic sink
 - Single IP flow was forwarded using a single CPU.
 - Realistic multiple-flow stream with varying destination address and packet sizes using a single CPU.
 - Multi-queues on the interface cards were used to dispatch different flows to four different CPUs.



Single flow, single CPU: Packets per second



Single flow, single CPU: Bandwidth



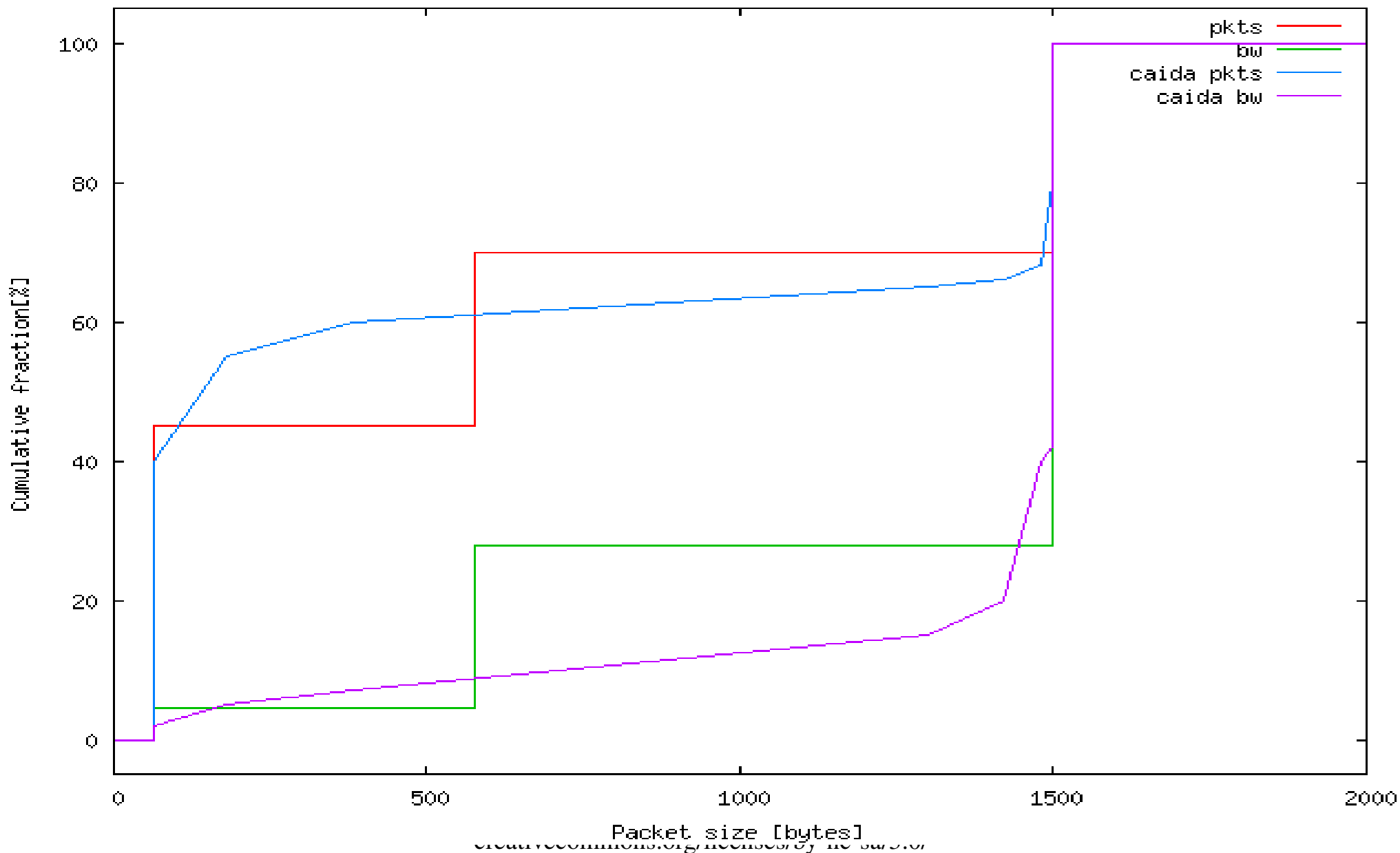
Single sender forwarding summary

- Virtually wire-speed for 1500-byte packets
- Little difference between forwarding on same card, different ports, or between different cards
 - Seems to be slightly better perf on same card, but not significant
- Primary limiting factor is pps, around 900Kpps
- TX has small effect on overall performance

Introducing realistic traffic

- For the rest of the experiments we introduce a more realistic traffic scenario
- Multiple packet sizes
 - Simple model based on realistic packet distribution data
- Multiple flows (multiple dst IP:s)
 - This is also necessary for multi-core experiments since NIC classification is made using hash algorithm on packet headers

Packet size distribution (cdf)

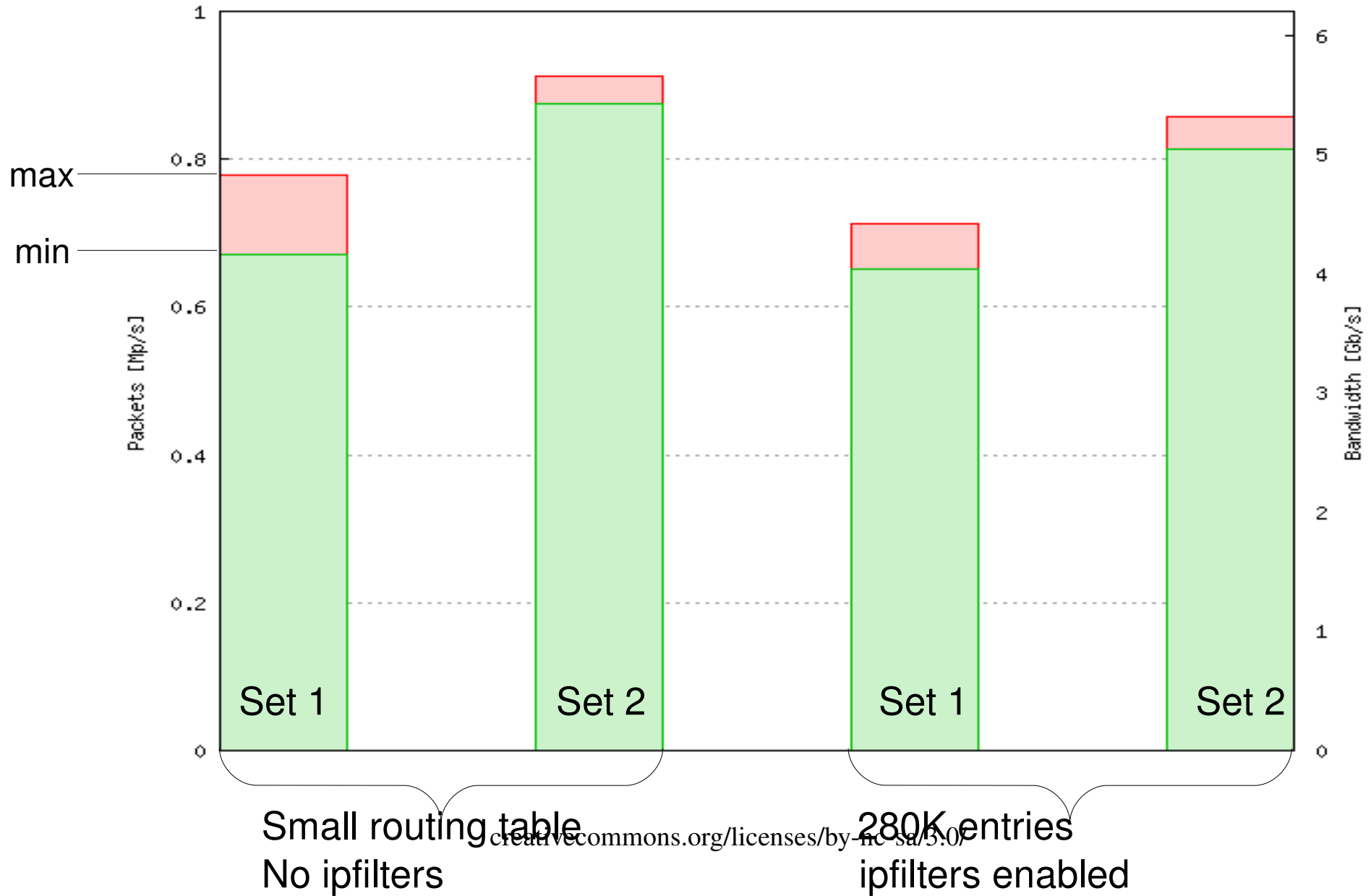


Real data from www.caida.org, Wide aug 2008

Flow distribution

- Flows have size and duration distributions
- 8000 simultaneous flows
- Each flow 30 packets long
 - Mean flow duration is 258 ms
- 31000 new flows per second
 - Measured by dst cache misses
- Destinations spread randomly over 11.0.0.0/8
- FIB contains ~ 280K entries
 - 64K entries in 11.0.0.0/8
- This flow distribution is relatively aggressive

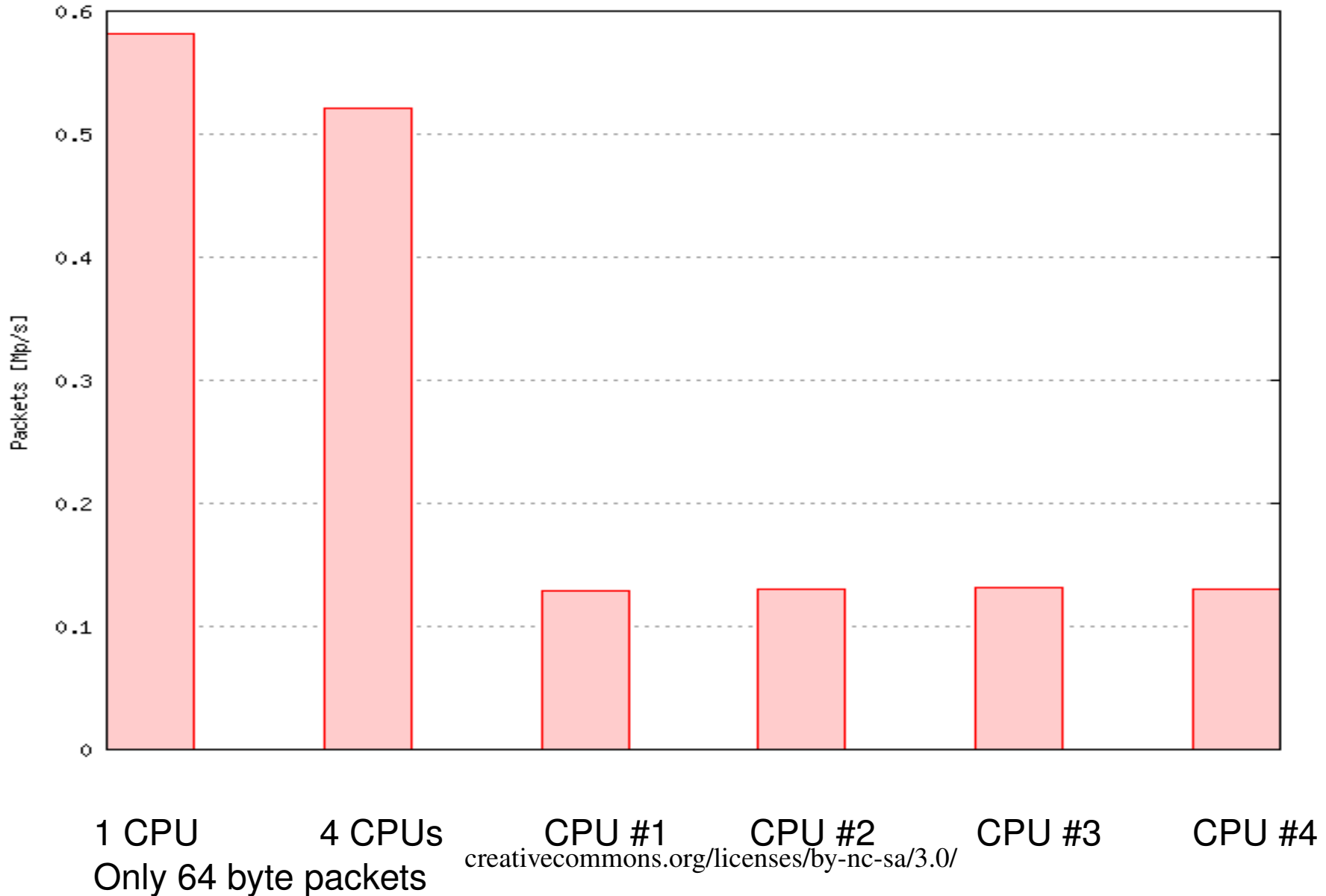
Multi-flow and single-CPU: PPS & BW



Multi-Q experiments

- Use more CPU cores to handle forwarding
- NIC classification (Receiver Side Scaling RSS) uses hash algorithm to select input queue
- Allocate several interrupt channels, one for each CPU.
- Flows are distributed evenly between CPUs
 - need aggregated traffic with multiple flows
- Questions:
 - Are processing of flows evenly dispatched ?
 - Will performance increase as CPUs are added?

Multi-flow and Multi-CPU (set 1)



Results MultiQ

- Packets are evenly distributed between the four CPUs.
- But forwarding using one CPU is better than using four CPUs!
- Why is this?

Profiling.

Single CPU

| samples | % | symbol name |
|---------|---------|----------------------|
| 396100 | 14.8714 | kfree |
| 390230 | 14.6510 | dev_kfree_skb_irq |
| 300715 | 11.2902 | skb_release_data |
| 156310 | 5.8686 | eth_type_trans |
| 142188 | 5.3384 | ip_rcv |
| 106848 | 4.0116 | __alloc_skb |
| 75677 | 2.8413 | raise_softirq_irqoff |
| 69924 | 2.6253 | nf_hook_slow |
| 69547 | 2.6111 | kmem_cache_free |
| 68244 | 2.5622 | netif_receive_skb |

Multiple CPUs

| samples | % | symbol name |
|---------|---------|--------------------|
| 1087576 | 22.0815 | dev_queue_xmit |
| 651777 | 13.2333 | __qdisc_run |
| 234205 | 4.7552 | eth_type_trans |
| 204177 | 4.1455 | dev_kfree_skb_irq |
| 174442 | 3.5418 | kfree |
| 158693 | 3.2220 | netif_receive_skb |
| 149875 | 3.0430 | pfifo_fast_enqueue |
| 116842 | 2.3723 | ip_finish_output |
| 114529 | 2.3253 | __netdev_alloc_skb |
| 110495 | 2.2434 | cache_alloc_refill |

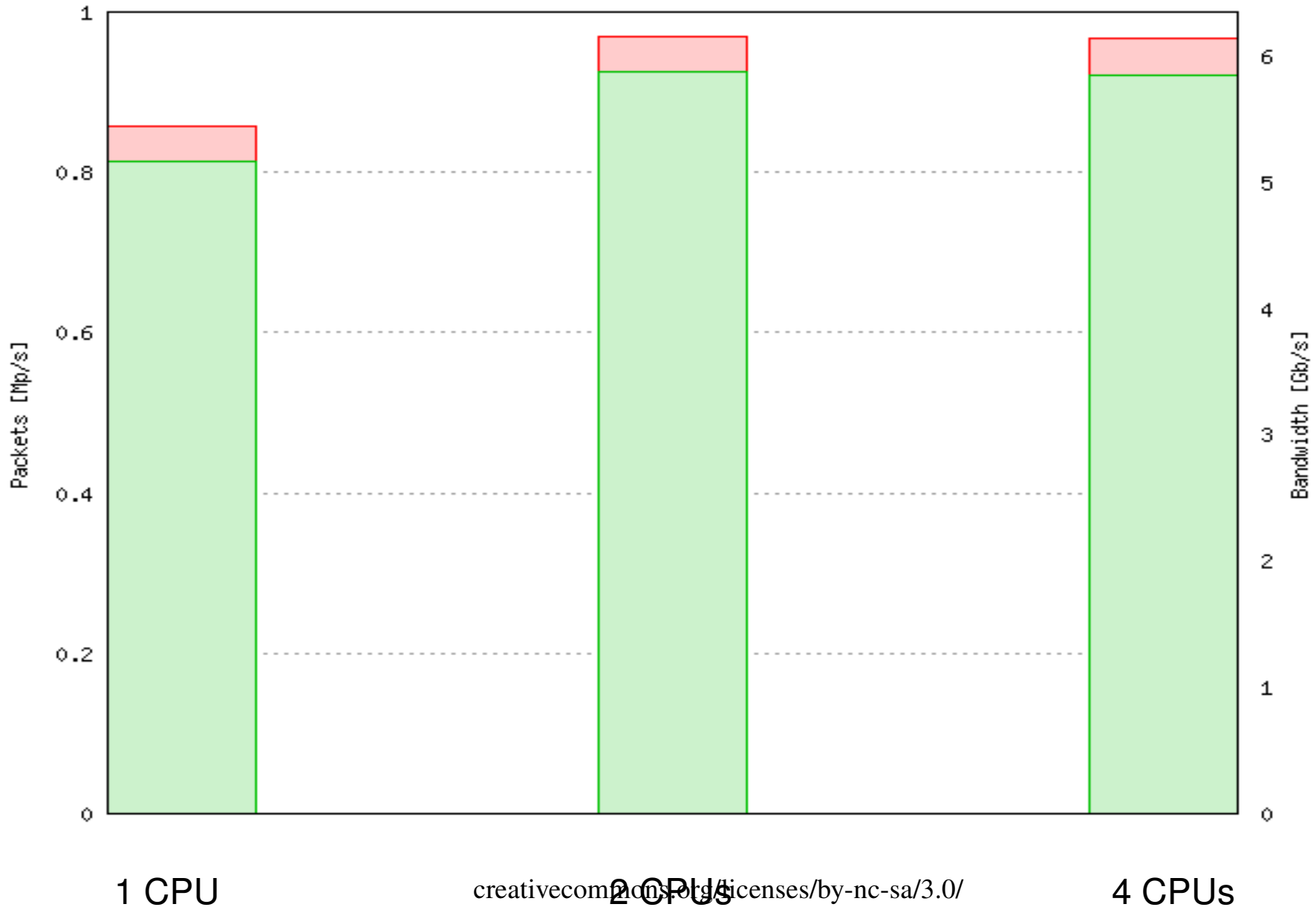
Multi-Q analysis

- With multiple CPUs: TX processing is using a large part of the CPU making using more CPUs sub-optimal
- It turns out that the Tx and Qdisc code needs to be adapted to scale up performance

MultiQ: Updated drivers

- We recently made new measurements (not in paper) using updated driver code
- We also used hw set 2 (Barcelona) to get better results
- We now see an actual improvement when we add one processor
- (More to come)

Multi-flow and Multi-CPU (set 2)



Conclusions

- Tx and forwarding results towards 10Gb/s performance using Linux and selected hardware
- For optimal results hw and sw must be carefully selected.
- >25Gb/s Tx performance
- Near 10Gb/s wirespeed forwarding for large packets
- Identified bottleneck for multi-q and multi-core forwarding.
- If this is removed, upscaling performance using several CPU cores is possible to 10Gb/s and beyond.