NaaS Workshop Dec'14

Rolf Neugebauer <rolf.neugebauer@netronome.com>

Netronome

- Fabless semi-conductor
- around 200 people in 4 main development sites
- Highly programmable network cards
  - originally derived from Intel's IXP
- NFP-32xx
  - Available since 2010(ish)
  - 2x10G, 40 cores (8 way HW threaded), PCIe Gen2 x8
- NFP-6xxx
  - Up to 2x100G, but typically 1 or 2x40G
  - Up to 4 PCIe gen3 x8 (typically 1 or 2 PCIe)
  - Up to 120 cores (8 way HW threaded) with high performance interconnect
  - Multiple memory units and specialised engines

Product lines

FlowNIC

- Open vSwitch Offload
- Standard Drivers and APIs
- Linux Virtualization Support

SDN Gateway

- Data Center to WAN
- OVSDB, OF-Config
- Open Daylight

Middlebox

- SDN-Controlled Applications
- PCIe IOV for VNFs
- Service Chaining

Intelligent ToR

- Disaggregated Server I/O
- Reduced vSwitch Instances
- Optimized PCIe Connectivity

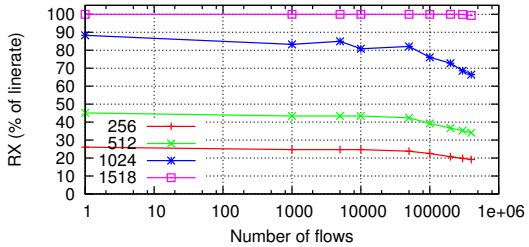
- Common: network co-processor to x86

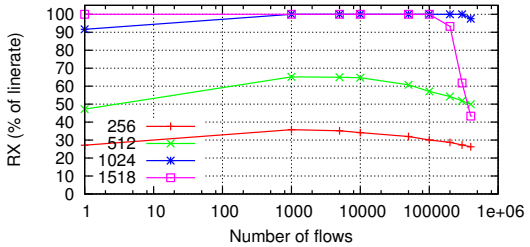
NETRONOME

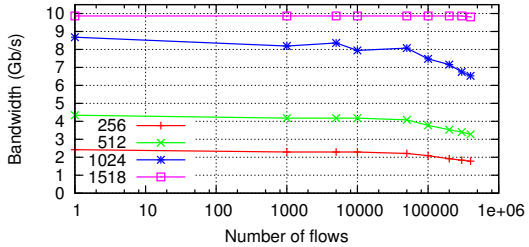
x86 overheads

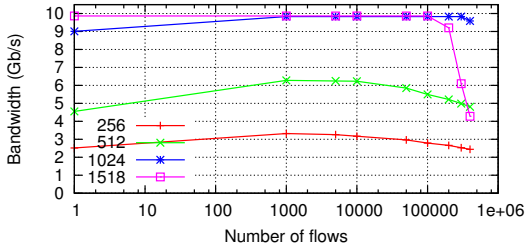
- Configure OVS to forward between two ports (gateway, middlebox)
- Multiple flows and multiple CPUs (with RSS)

- Single flow: max 1.2Mpps (out of 14.8Mpps)
- Increase #flows -> Perf drops
- Using RSS to fan out to multiple cores has little impact
- And we are just forwarding packets
- At “just” 10G!
- Without VMs!
- It gets worse…







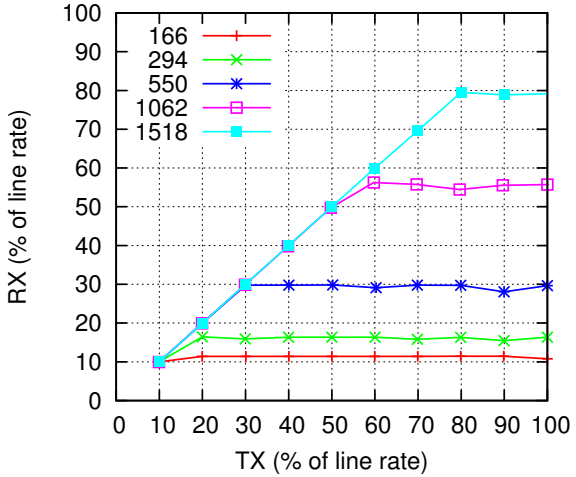


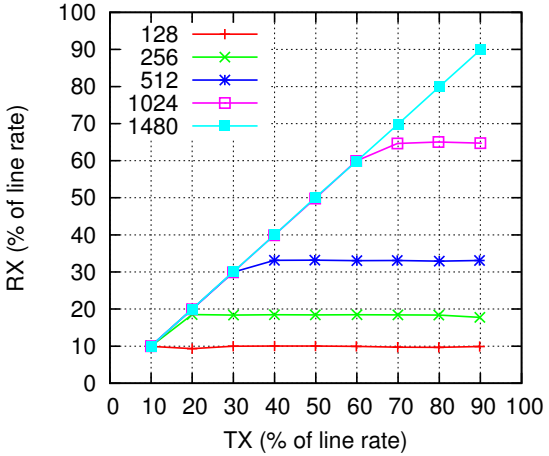
Intel Xeon E5-2360 (SNB), 2.3GHz, 16GB Mem, Intel 82599EB NIC, Ubuntu 12.04, OVS 1.7.x

NETRONOME

x86 Overheads (cont)

- Doing some work (GRE decap and encap)

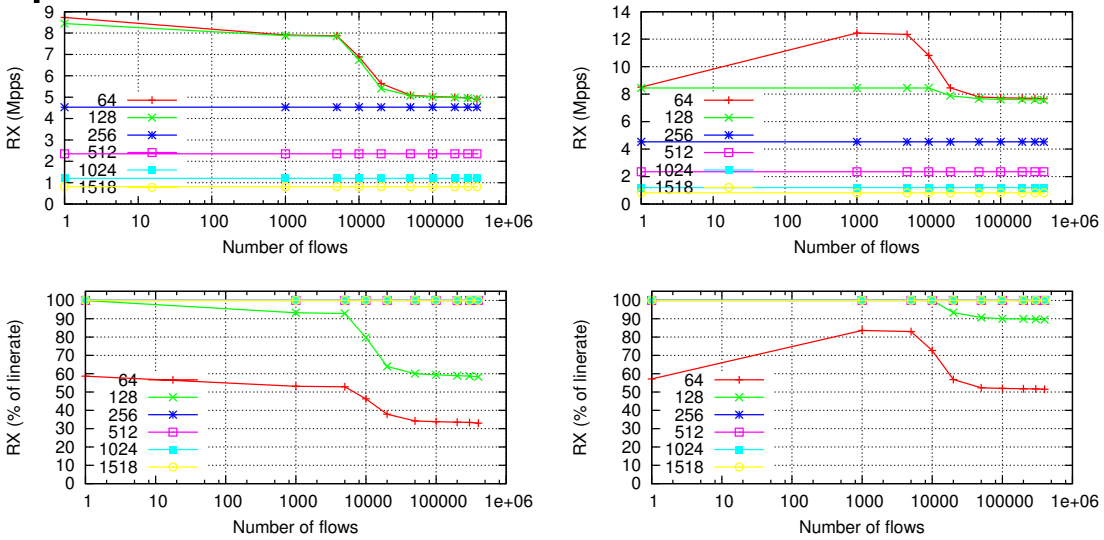




- Max out at 650-750Kpps
- In fairness OVS perf has improved a little since…

DPDK to the rescue (sorta, kinda)

- 10+G line rate forwarding on a single core! but…
- With per flow state:



# lcores	PPS
1	4.9Mpps
2	7.6Mpps
3	10.2Mpps
4	12.5Mpps
5	14.8Mpps

- “Just” 10G, no VMs yet, burning x86 cores…

How we address this

- Offload processing to the NIC
  - pre-process and filter on the NIC (OVS offload)
  - cut-through or drop packets on the NIC for middlebox apps
- High degree of concurrency to hide memory latency
  - up-to 120 x 8 threads
- Hierarchy of memories
  - from small and fast to large and slow
- Specialised engines close to memory
  - for Lookup, stats, etc
- Programmability to follow SW not HW product cycles

Challenges

- (Better) APIs
  - Allow MB applications to better control behaviour of switch/NIC
- Ease of Programming:
  - General purpose CPU >> NFP >> FPGA >> ASIC
  - Protocol oblivious, P4 etc
    - step in right direction, but somewhat limited
- Efficient VM connectivity