MoonGen

A Scriptable High-Speed Packet Generator

Paul Emmerich

January 31st, 2016
FOSDEM 2016

Chair for Network Architectures and Services
Department of Informatics
Technical University of Munich (TUM)

Outline

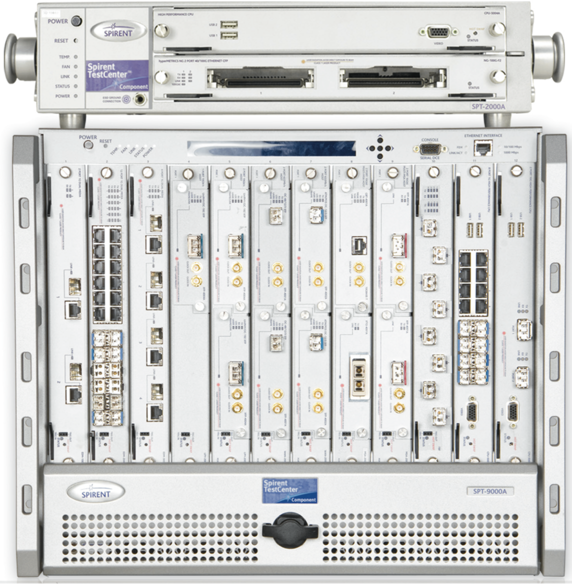
Hardware vs. Software Packet Generators

Architecture of MoonGen

Hardware Timestamping on Commodity NICs

Precise Rate Control

Example Measurements



Source: www.spirent.com

Challenges for software packet generators

- Hardware packet generators are
  - Precise
  - Accurate
  - Fast

Challenges for software packet generators

- Hardware packet generators are
  - Precise
  - Accurate
  - Fast
- Software packet generators
  - Run on cheap commodity hardware
  - Flexible

Challenges for software packet generators

- Hardware packet generators are
  - Precise
  - Accurate
  - Fast
- Software packet generators
  - Run on cheap commodity hardware
  - Flexible
- Key challenges for software packet generators
  - Rate control
  - Timestamping

Design goals

Design goal of MoonGen

Combine the advantages of both approaches while avoiding their disadvantages.

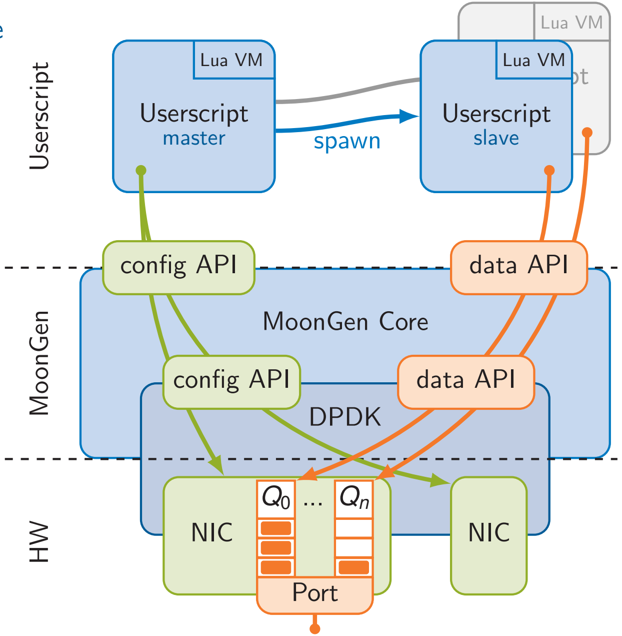
Design goals

Design goal of MoonGen

Combine the advantages of both approaches while avoiding their disadvantages.

- Fast: DPDK for packet I/O, explicit multi-core support
- Flexible: Craft all packets in user-controlled Lua scripts
- Timestamping: Utilize hardware features found on modern commodity NICs
- Rate control: Hardware features and a novel software approach

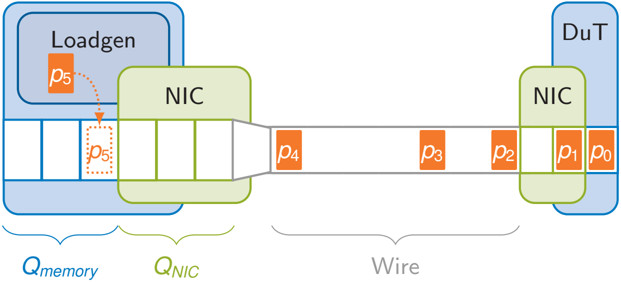
Architecture



Hardware timestamping

- NICs support PTP for precise clock synchronization
- PTP support requires hardware timestamping capabilities
- These can be (mis-)used for delay measurements
- Typical precision
  - ± 6.4 ns (Intel 10 GbE chips)
  - ± 32 ns (Intel GbE chips)
- Some restrictions
  - Packets must be UDP or PTP L2 protocol
  - Minimum UDP packet size is 84 bytes

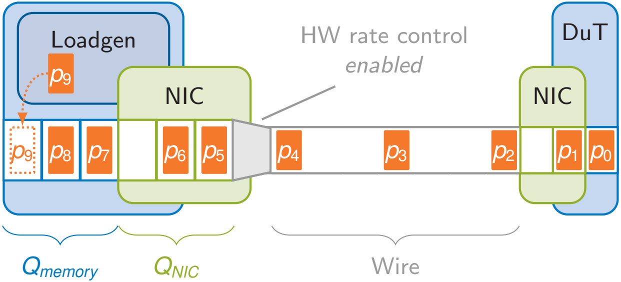
Software rate control in existing packet generators



- Software tries to push single packets to the NIC
- Queues cannot be used, no batch processing
- NICs work with an asynchronous push-pull model
- This can lead to micro-bursts

→ Unreliable, imprecise, and bad performance

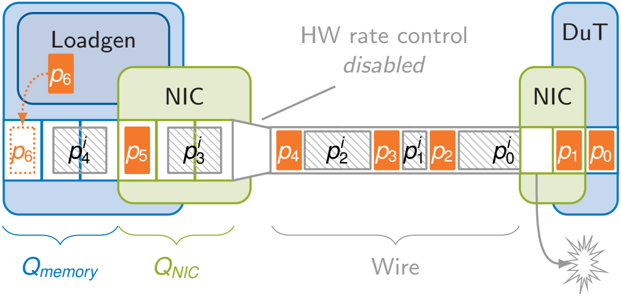
Hardware rate control



- Modern NICs support rate control in hardware
- Limited to constant bit rate and bursty traffic
- Precision controlled by the hardware

→ High performance as queues can be used, but inflexible

Software rate control based on invalid packets

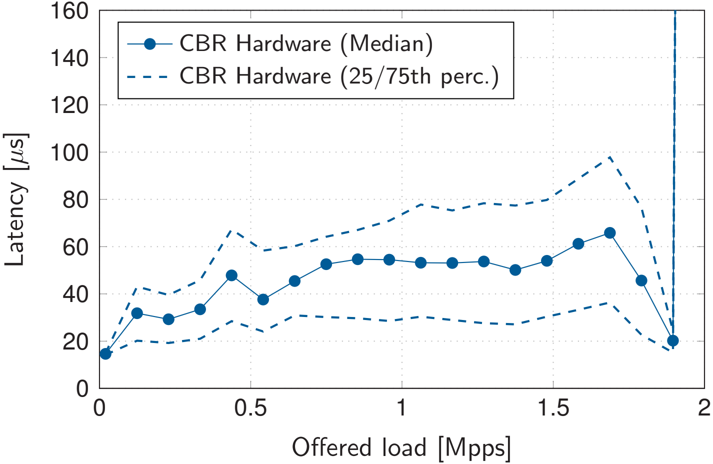


- Fill gaps with invalid packets p^i (e.g. bad CRC)
- NIC in the DuT drops invalid packets without side-effects
- Combines advantages of both approaches
- Precision limited by byte rate (0.8 ns per byte) and minimum packet size (33 byte)

→ High performance & high precision

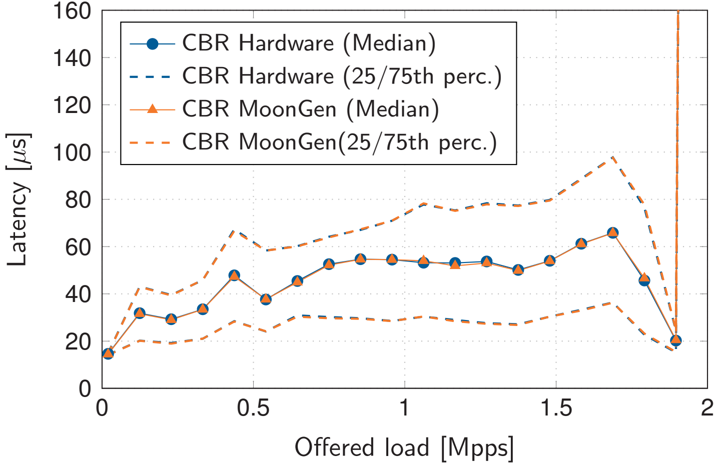
Does it work?

- Test setup: forward packets with Open vSwitch
- Measure the latency of the device under test



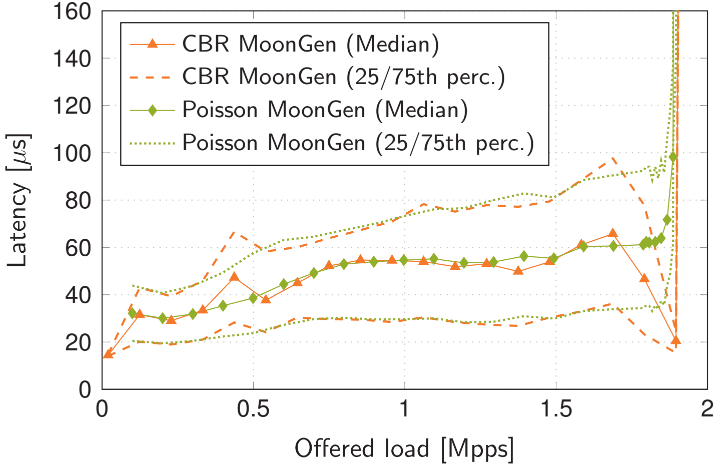
Does it work?

- Compare both rate control approaches
- Maximum deviation: 2%



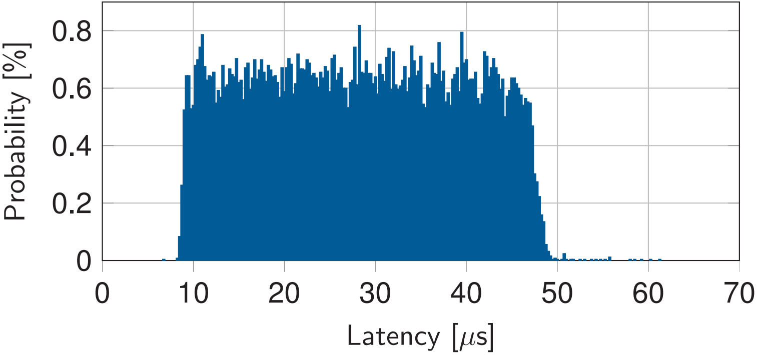
Does it matter?

- Compare CBR with Poisson traffic
- Different response from the device under test



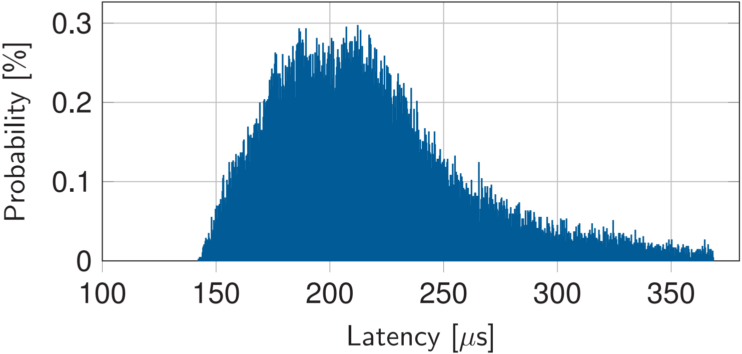
Example: Linux NAPI

- Open vSwitch on Linux
- Uniform distribution caused by interrupt throttling



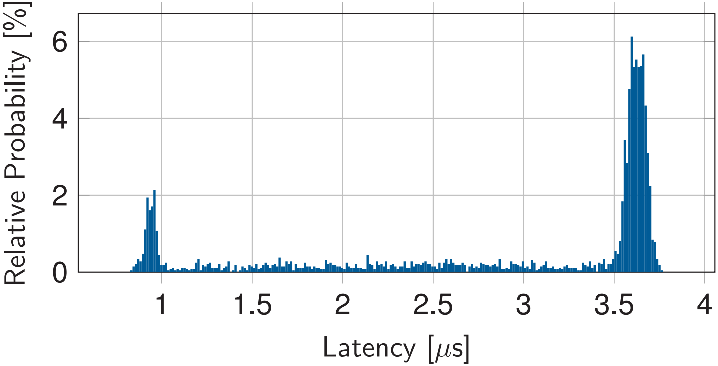
Example: Linux Virtualization (VirtIO)

- Open vSwitch forwarding through a VM
- Long tail distribution, typical for VMs



Example: Hardware Switch

- AS5712-54X 10/40 GbE OpenFlow switch
- Bimodal distribution caused by more input than output ports
  - Some packets are forwarded directly (cut-through switch)
  - Some packets are blocked by another flow and buffered



Summary

- Speeds of 10 Gbit/s per CPU core (64 byte packets)
- Sub-microsecond precision and accuracy
- Execute user-defined script code for each packet
- Easy to use

Q & A

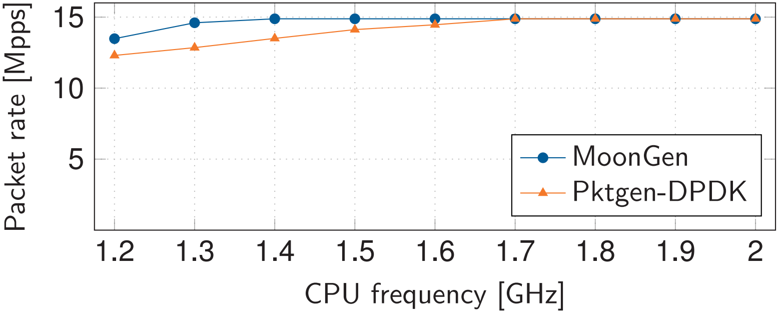
Try MoonGen yourself!

https://github.com/emmericp/MoonGen

Questions?

[Backup slide] Performance I: Lua can be faster than C

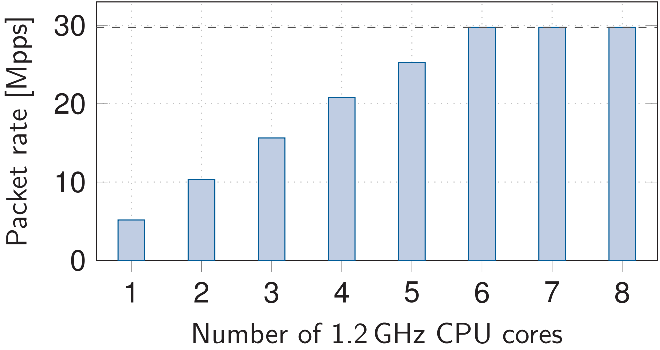
- UDP packets from varying source IP addresses



- Pktgen-DPDK needs a complicated main loop that covers all possibilites
- MoonGen can use a tight inner loop

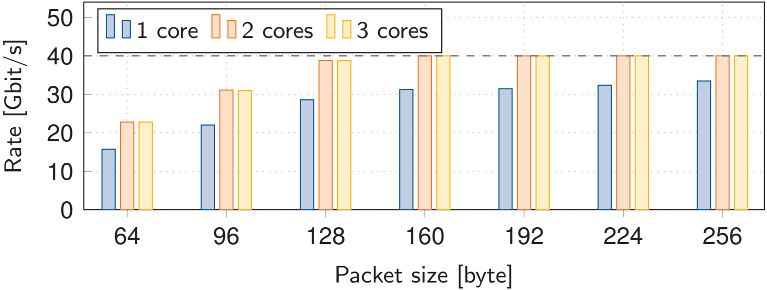
[Backup slide] Performance II: heavy workload and multi-core scaling

- Generate random UDP packets on 2 10 Gbit NICs
- 8 calls to Lua's standard `math.random` per packet
- CPUs artificially clocked down to 1.2 GHz

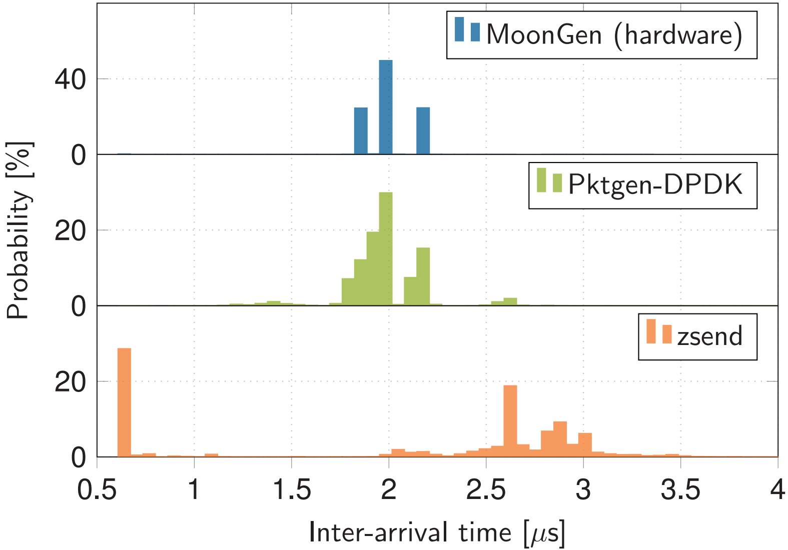


[Backup slide] Performance III: 40 GbE

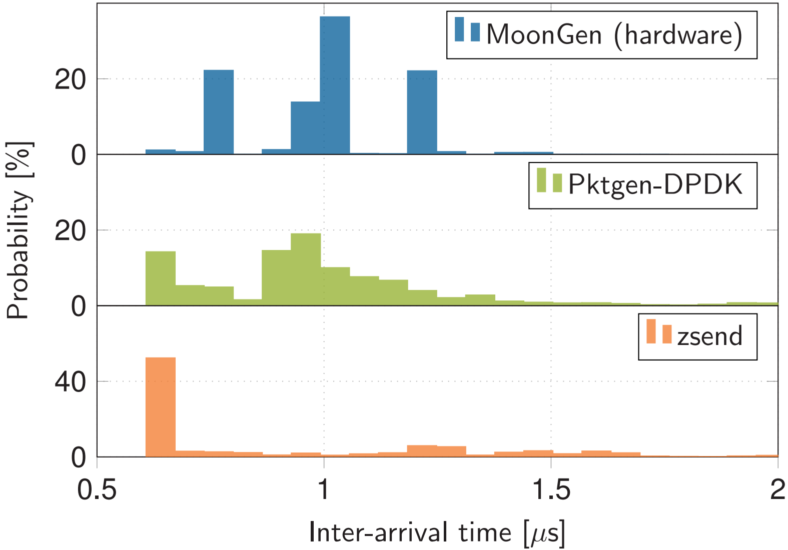
- Generate random UDP packets on 2 10 Gbit NICs
- 8 calls to Lua's standard `math.random` per packet
- CPUs artificially clocked down to 1.2 GHz



[Backup Slide] Rate control: 500 kpps



[Backup Slide] Rate control: 1,000 kpps

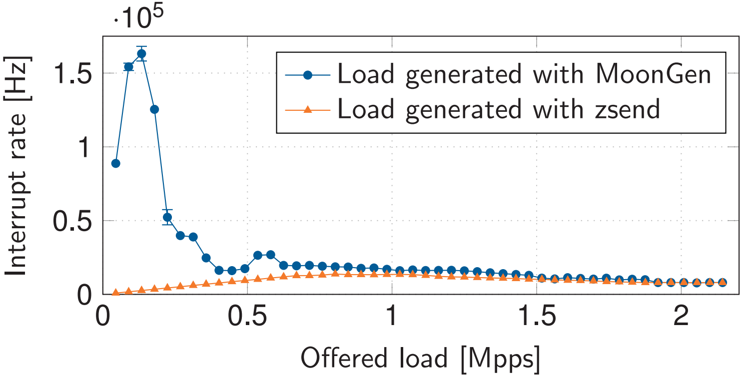


[Backup Slide] HW/SW rate control details

Rate	Software	Bursts	±64 ns	±128 ns	±256 ns	±512 ns
500 kpps	MoonGen	0.02%	**49.9%**	**74.9%**	**99.8%**	**99.8%**
	Pktgen-DPDK	**0.01%**	37.7%	72.3%	92%	94.5%
	zsend	28.6%	3.9%	5.4%	6.4%	13.8%
1000 kpps	MoonGen	**1.2%**	**50.5%**	52%	**97%**	**100%**
	Pktgen-DPDK	14.2%	36.7%	**58%**	70.6%	95.9%
	zsend	52%	4.6%	7.9%	24.2%	88.1%

[Backup slide] Effects of bad rate control

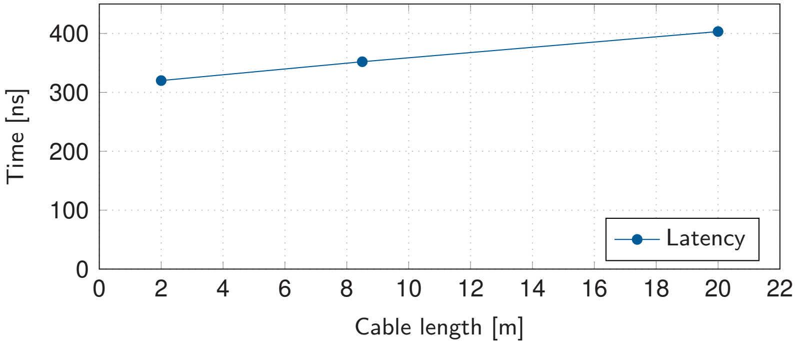
- Interrupt rate of an Open vSwitch packet forwarder



- Micro-bursts confuse dynamic interrupt throttling
- This affects latency (cannot be measured with zsend)

[Backup slide] Hardware timestamping precision and accuracy

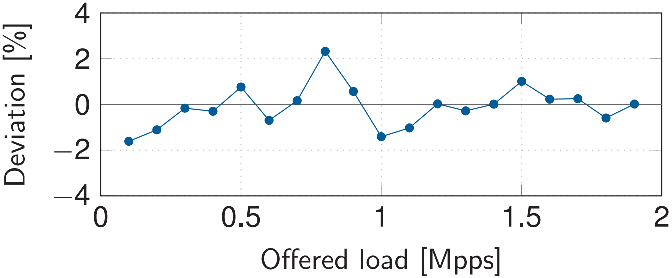
- Measure latencies of cables of various length
- Calculate encoding time k and propagation speed v_p



- Result for fiber cable: $k \approx 311$ns, $v_p = 0.72c \pm 0.056c$

[Backup slide] Effects of invalid packets

- Median latency of an Open vSwitch packet forwarder
- Packet rate controlled by hardware vs. invalid frames



- Minor modifications to the DuT (e.g. an active SSH session) result in a deviation of up to 15% with the same rate control mechanism