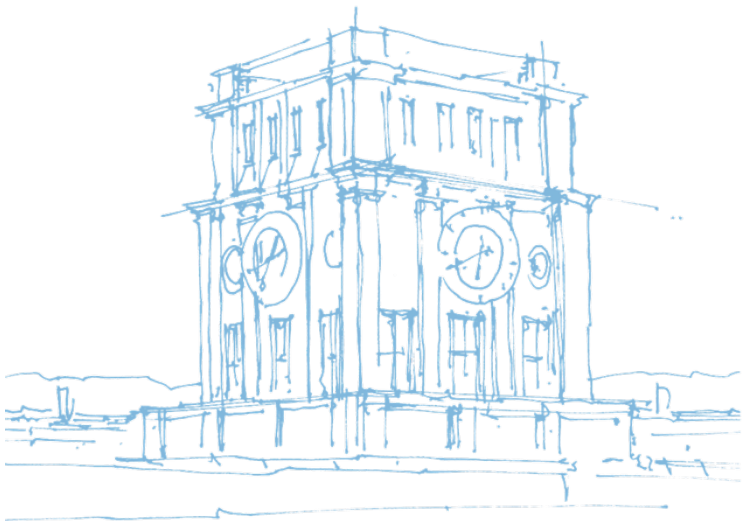
Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

# Measuring High-Performance Packet Processing

**Georg Carle,** Sebastian Gallenmüller, Dominik Scholz,
Florian Wohlfart, Quirin Scheitle and Paul Emmerich

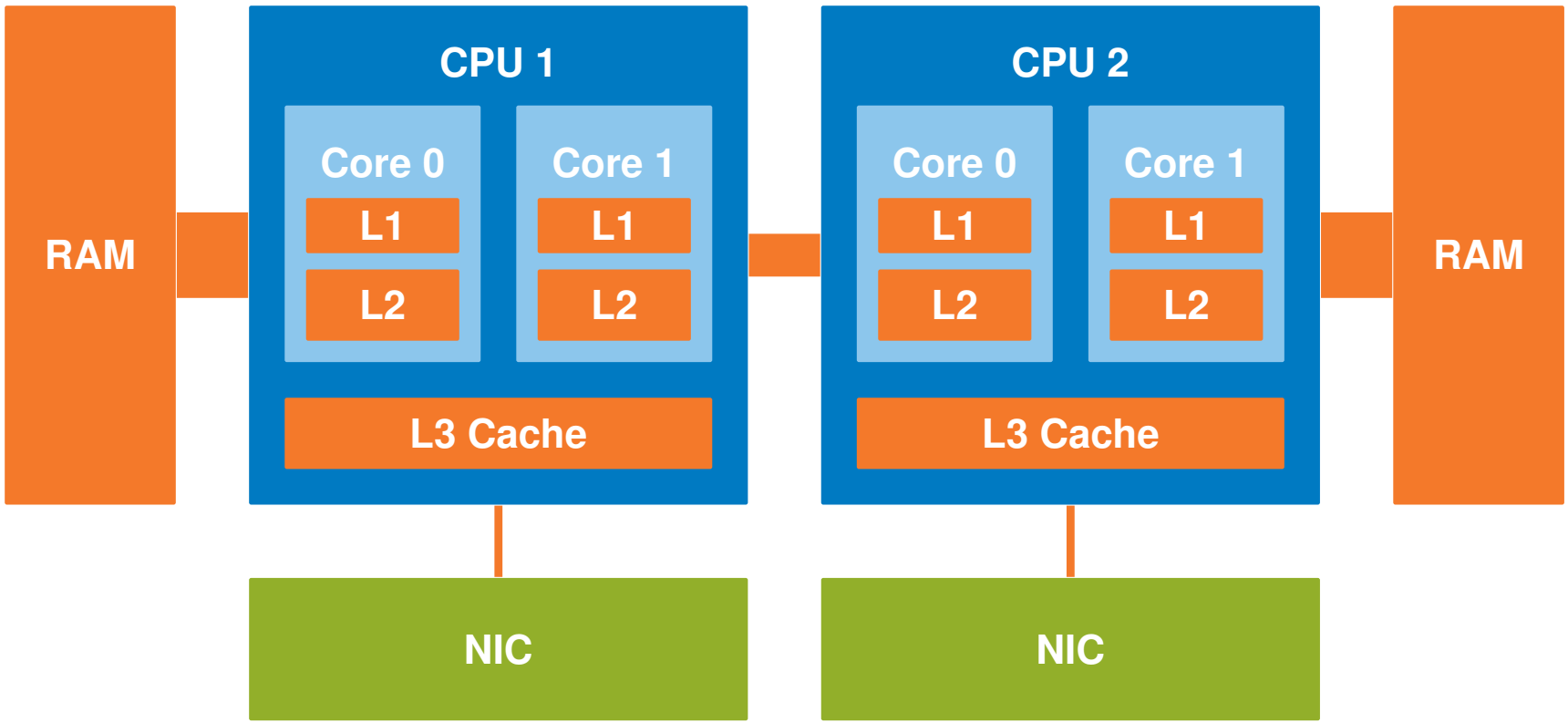
November 29, 2017

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



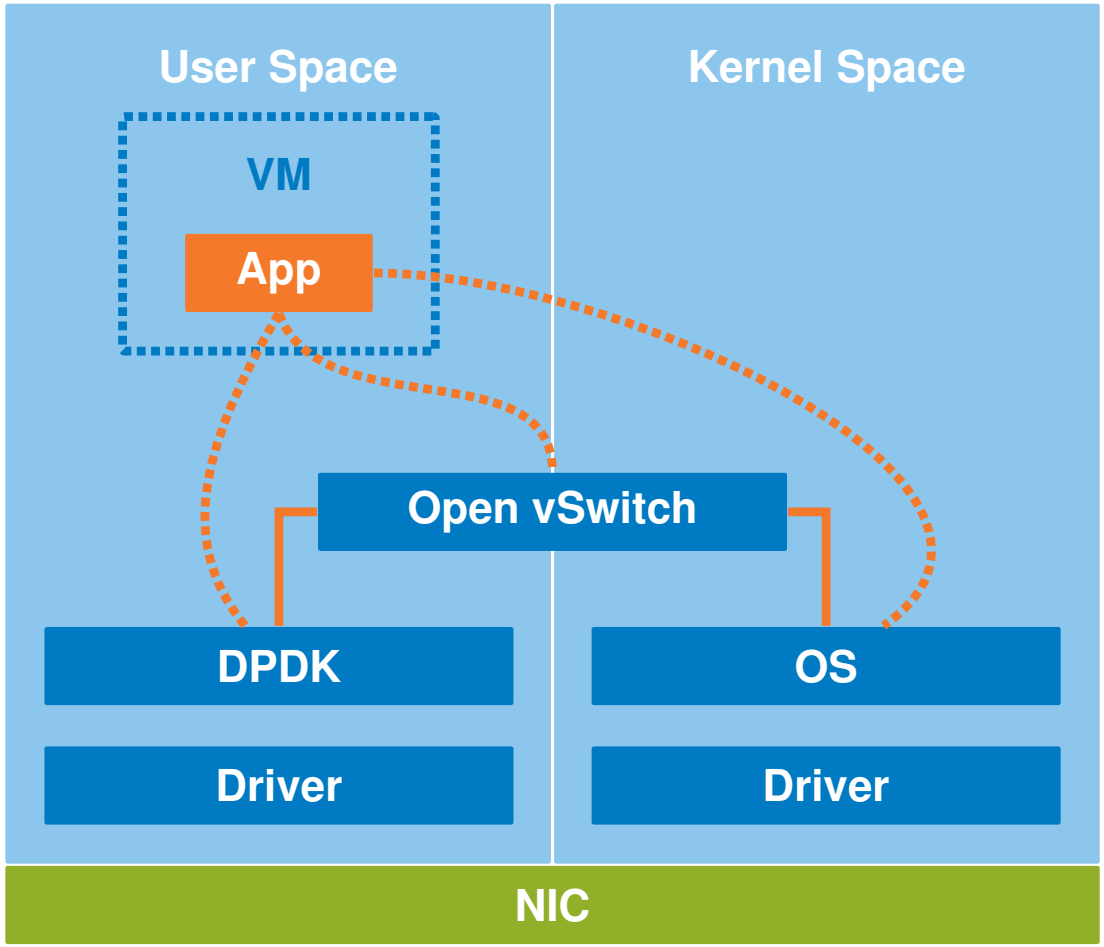
# Motivation

## Modern hardware architectures



# Motivation

## Modern software architectures



# Motivation

## Trends for computer networks

### Hardware trends

- Multi-core/many-core CPUs
- Multi-queue NICs

### Software trends

- High-performance packet processing frameworks: DPDK, netmap, Snabb, etc.
- Virtualization: Xen, KVM
- Containers: Kernel namespaces, cgroups

# Motivation

## Challenges

### Complexity of commmunication stack

- Protocols have different mechanisms and parameters
- Multiple layers, complex interaction
- Complex implementation in software and hardware

### Measurements

- Topology/configuration/state
- Automation of experiments
- Resource usage (CPU, RAM, energy) of different implementations

### Modeling

- Characterisation of wireless stack protocols
- Description of measurements
- Selection of models
- Coverage of relevant parameters

# Reproducibility

## MoMeTools - Workshop on Models, Methods and Tools for Reproducible Network Research


**Georg Carle, Hartmut Ritter, Klaus Wehrle:**
*"MoMeTools - Workshop on Models, Methods and Tools for Reproducible Network Research"*,
**Karlsruhe, Germany, August 2003**
in conjunction with ACM SIGCOMM 2003


Olivier Bonaventure, Luigi Iannone, Damien Saucez:
*"Reproducibility Workshop (Reproducibility'17)"*,
Los Angeles, USA, **August 2017**
in conjunction with ACM SIGCOMM 2017

Despite 14 years have passed the goals are still the same . . .

- Establish methodology for experiments on computer networks
- Discussion about current approaches and improvement
- Initiate a commonly accepted methodology for the scientific fields of computer networks

Q. Scheitle, M. Wählisch, O. Gasser, et al., „Towards an Ecosystem for Reproducible Research in Computer Networking“, in Proceedings of the Reproducibility Workshop, Reproducibility@SIGCOMM 2017, Los Angeles, CA, USA, August 25, 2017

# Reproducibility

## Open questions for network measurements

The problems discussed in both workshops are still relevant today.

- What influences the performance of packet processing?
- Which KPIs are relevant?
- How to measure these relevant KPIs?
- How to build experiment setups measuring these KPIs?
- How to measure in a **replicable** or even **reproducible** manner?

Classification according to ACM

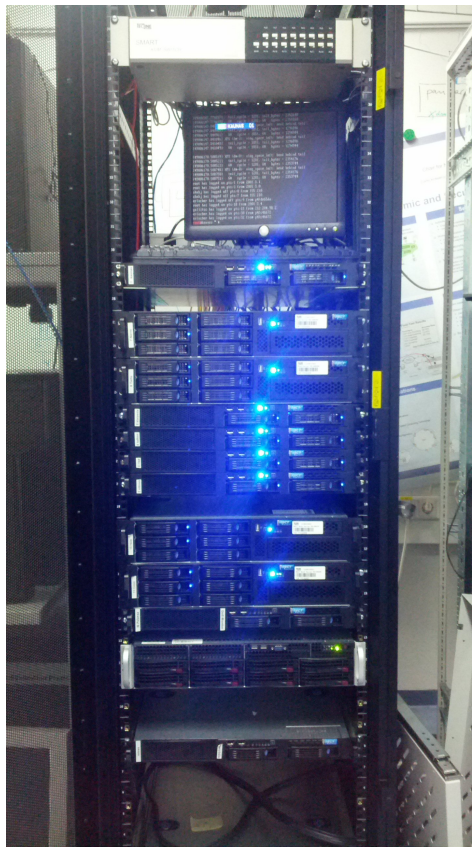
- **Repeatability:** Same people use same setup to repeat results.
- **Replicability:** Different people use same setup to replicate results.
  We provide “artifacts” (scripts, data, documentation) so any other team can easily replicate our work.
- **Reproducibility:** Different people use different setup to reproduce results.
  We provide a detailed documentation of our approach so other teams can reproduce our work without using our artifacts.

# Testbed for Reproducibility

## pos in Practice: the Baltikum Testbed

**Automated reproducible network experiments**

- Input: test configuration file
- Boot test machines
- Deploy system image via network
- Deploy host scripts
- Supervise test sequence
- Collection of results
- Output: measurement results



**Previous measurements include**

- Different wired network setups, also VM setups
- Energy consumption of network nodes
- Different OSes (Linux, FreeBSD, Windows)
- Automated device benchmarking
  (RFC 2544, OpenFlow)

# libmoon – High-Performance Packet Processing

## libmoon

What is libmoon?

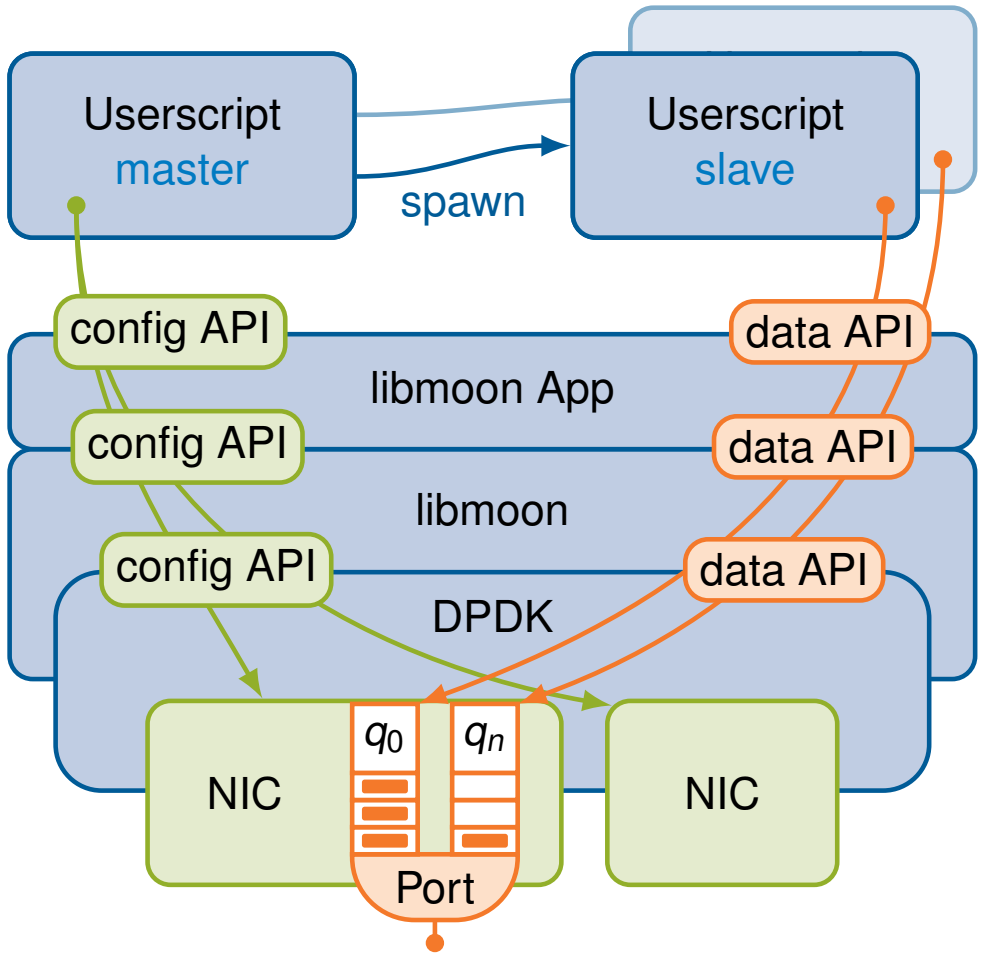
- High performance packet processing framework
- Flexible
- Lua wrapper based on DPDK
- Supporting libraries

Applications powered by libmoon:

- Traffic generation (MoonGen [IMC15])
- Flexible router (MoonRoute [ANCS17_2])
- Incident detection/flow analysis (Flowscope [IFIP17])
- High-performance stack
- ...

# libmoon – High-Performance Packet Processing

## Architecture of libmoon Apps

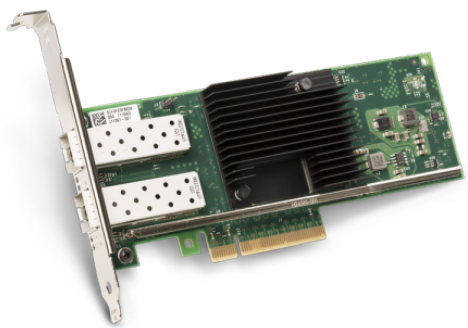


# MoonGen – High-Performance Packet Generation

## Software vs. Hardware Packet generators

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

Source: www.spirent.com



Source: www.intel.com

# MoonGen – High-Performance Packet Generation

## MoonGen features and publications

### Key features

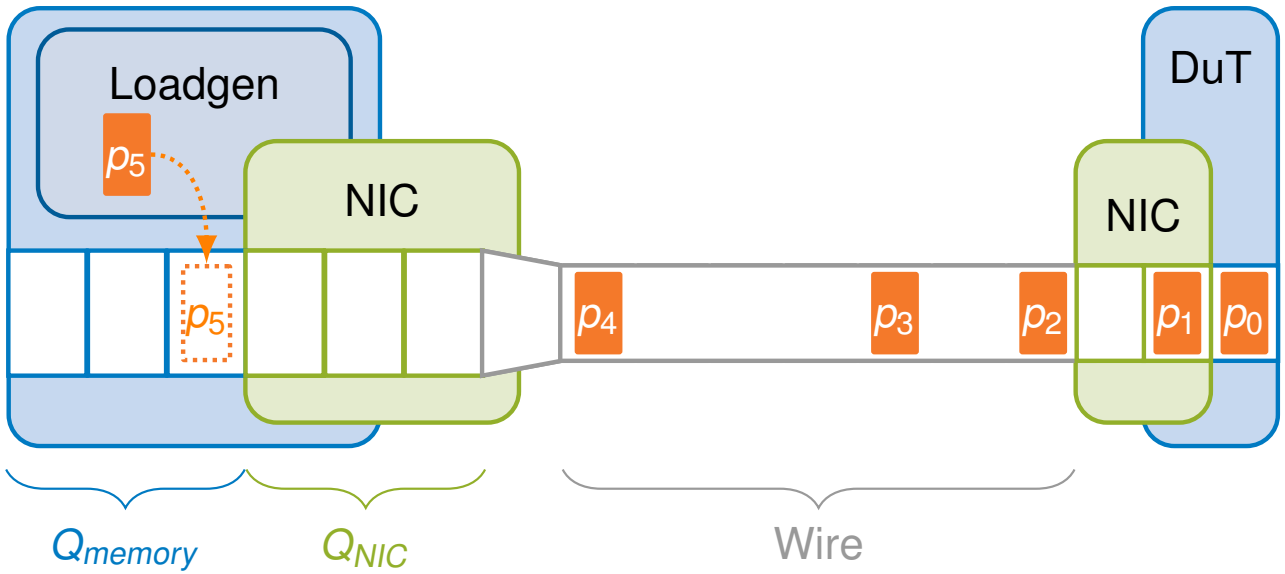
- Fast: DPDK for packet I/O, explicit multi-core support,
- Flexible: Craft all packets in user-controlled Lua scripts
- Timestamping: Utilize hardware features found on modern commodity NICs
- Rate control: Hardware features and a novel software approach

P. Emmerich, S. Gallenmüller, D. Raumer, et al., „MoonGen: A Scriptable High-Speed Packet Generator“, in Internet Measurement Conference (IMC) 2015, IRTF Applied Networking Research Prize 2017, Tokyo, Japan, Oct. 2015

P. Emmerich, S. Gallenmüller, G. Antichi, et al., „Mind the Gap – A Comparison of Software Packet Generators“, in ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2017), Beijing, China, May 2017

# MoonGen – High-Performance Packet Generation

## Rate control - pure software approach



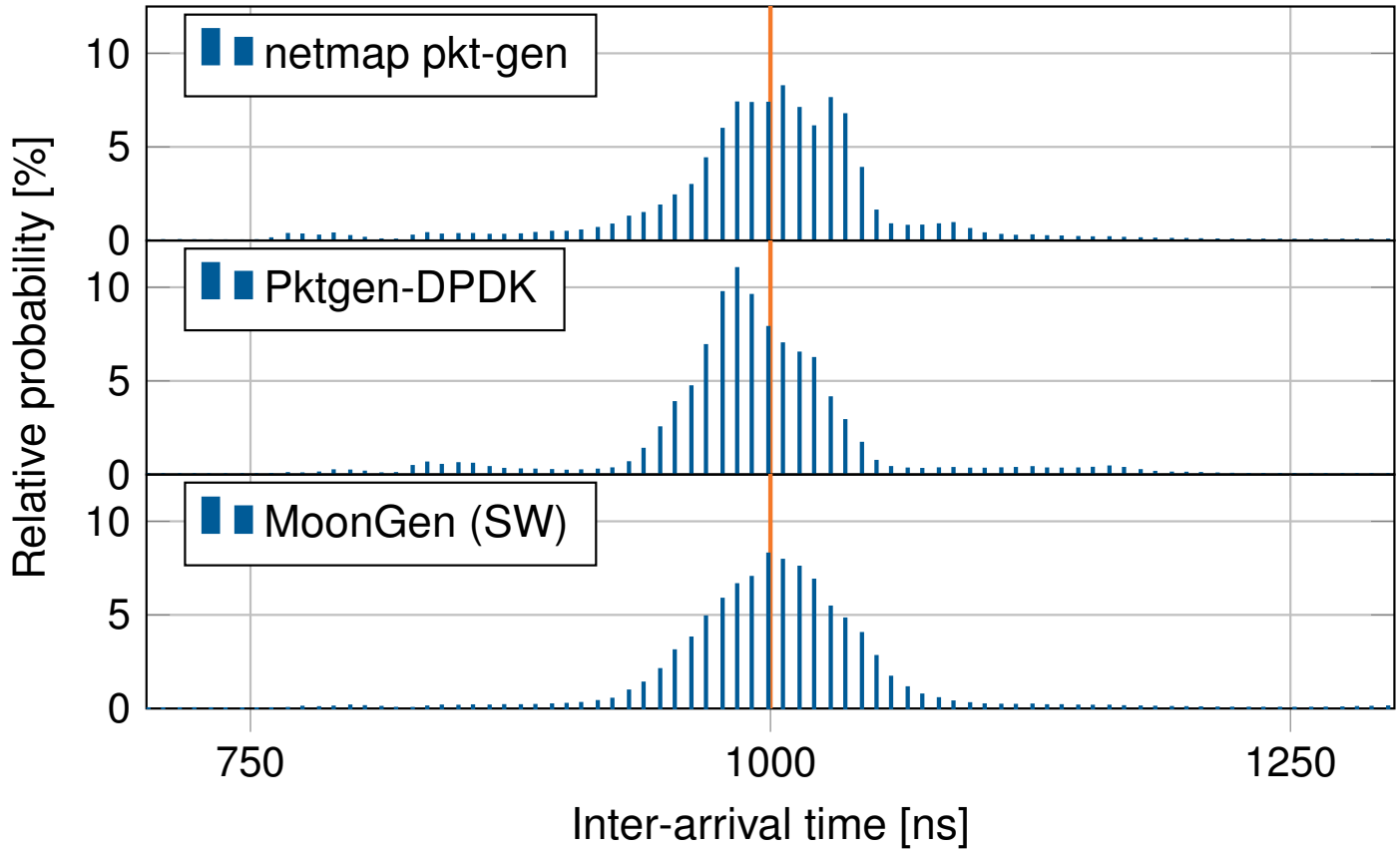

- Software pushes packets into buffer in RAM ($Q_{memory}$)
- NICs copy packets eventually from RAM to internal buffers ($Q_{NIC}$)
- NICs put packets stored in internal buffers on wire

→ Rate control by limiting the packets available in $Q_{memory}$

→ Packets are fed one-by-one to the NIC to achieve constant bit rate traffic

# MoonGen – High-Performance Packet Generation

## Rate control - pure software approach

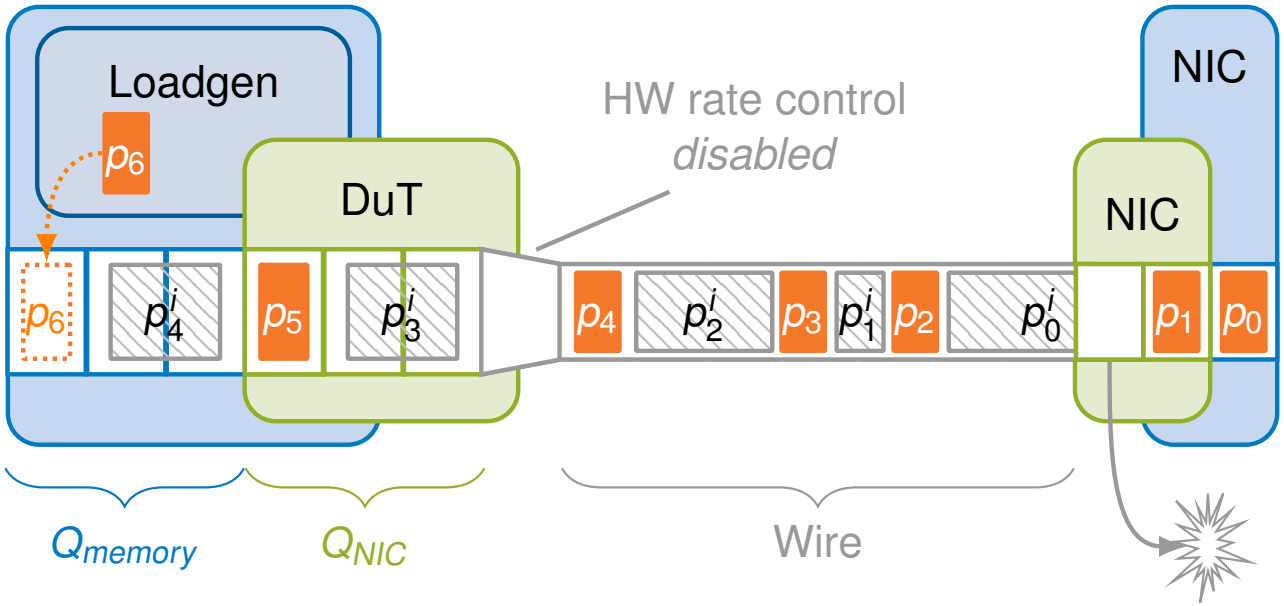



- Configured for a constant rate: 1 Mpps
- Measurement shows measured inter packet gap (ideal: 1000 ns)

➔ real IPG from 750 ns to 1250 ns

# MoonGen – High-Performance Packet Generation

## Rate control - corrupted CRC approach




- In between real packets, invalid packets with incorrect CRC checksums are inserted

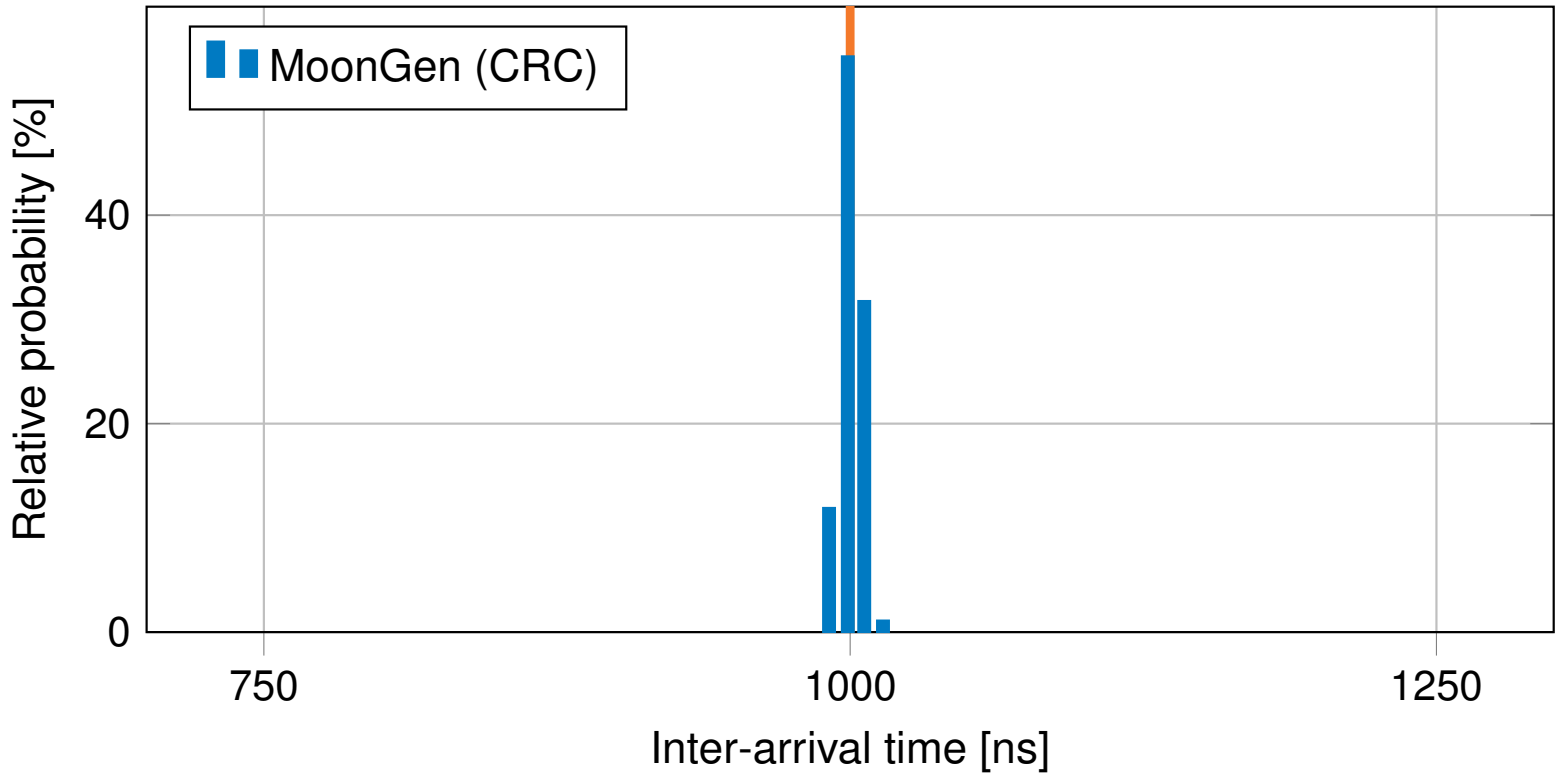
➔ Inter-packet gap controlled by size of corrupted CRC packets

➔ Packets can be fed in batches to the NIC

- Does not work with all devices under test

# MoonGen – High-Performance Packet Generation

## Rate control - corrupted CRC approach




- Configured for a constant rate: 1 Mpps
- Measurement shows measured inter packet gap (ideal: 1000 ns)

➔ real IPG closely meets ideal IPG

# MoonRoute – High-Performance Router

## MoonRoute – High-performance Router

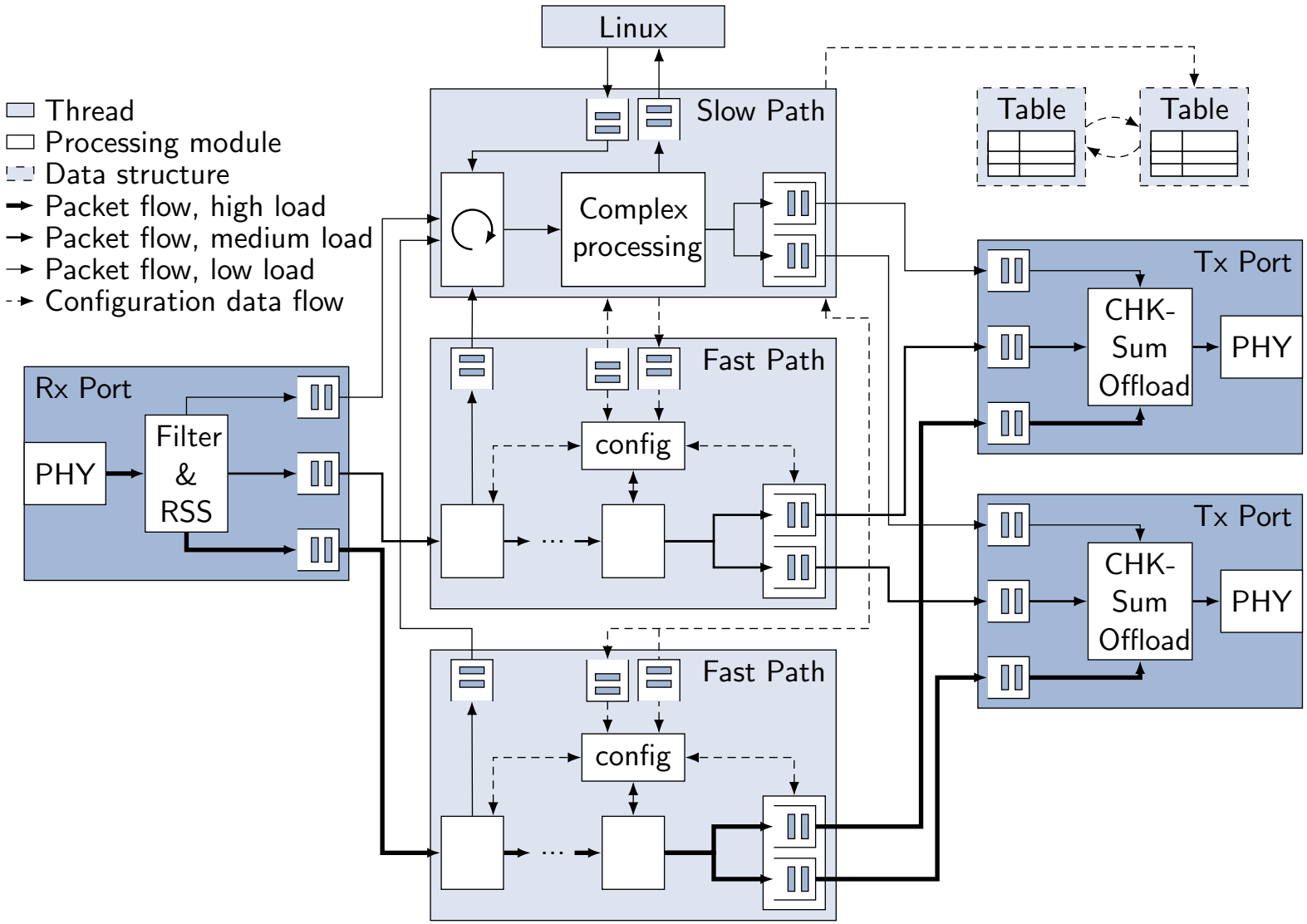
**Key features**

- Flexibility: Adding/removing code anywhere in the packet processing chain
- Code reuse: Mixing existing C libraries with Lua script code
- Performance: Saturating multiple 10 GbE ports with 64-byte packets on a single server
- Scalability: Multi/many-core architecture employing RSS, multi-queue & hardware filters
- Multiple processing paths: High-performance fast path, feature-rich slow path, or fallback to host OS routing

S. Gallenmüller, P. Emmerich, R. Schönberger, et al., „Building Fast but Flexible Software Routers“, in Proceedings of the Symposium on Architectures for Networking and Communications Systems, IEEE Press, 2017, pp. 101–102

# MoonRoute – High-Performance Router

## Applications of libmoon – MoonRoute



# MoonRoute – High-Performance Router

| Implementation | Src | Routing tbl. | CPU freq. [GHz] | Throughput [Mpps] reported | scaled | rel. |
|---|---|---|---|---|---|---|
| MoonRoute | — | 1 | 3.2 | 14.6 | 14.6 | 100% |
| MoonRoute | — | $2^{20}$ | 3.2 | 14.2 | 14.2 | 97% |
| MoonRoute | — | $2^{24}$ | 3.2 | 11.6 | 11.6 | 79% |
| 6WIND Turbo Router | [5] | Unknown | 2.8 | 9.6 | 11.0 | 75% |
| FastClick (DPDK 2.2) | – | 1 | 3.2 | 10.4 | 10.4 | 72% |
| FastClick (DPDK 2.2) | – | $2^{20}$ | 3.2 | 10.4 | 10.4 | 70% |
| Batching Click (PSIO) | [6] | Unknown | 8x 2.66 | 41.7 | 6.3 | 43% |
| Click (DPDK 2.2) | – | 1 | 3.2 | 4.3 | 4.3 | 29% |
| Click (DPDK 2.2) | – | $2^{20}$ | 3.2 | 4.2 | 4.2 | 28% |
| FreeBSD 11-routing | [7] | 2 | 8x 2.0 | 9.5 | 2.0 | 13% |
| Route Bricks | [8] | $2^{18}$ | 8x 2.8 | 12.0 | 2.0 | 12% |
| Linux 3.7 | – | 1 | 3.2 | 1.5 | 1.5 | 10% |
| Click | [9] | 8 | 0.7 | 0.4 | 1.6 | 11% |
| FreeBSD 10.2 | [10] | 8 | 4x 2.13 | 1.8 | 0.7 | 5% |
| Linux 2.2.14 | [9] | 8 | 0.7 | 0.1 | 0.4 | 3% |

## FlowScope – High-Performance Traffic Analysis

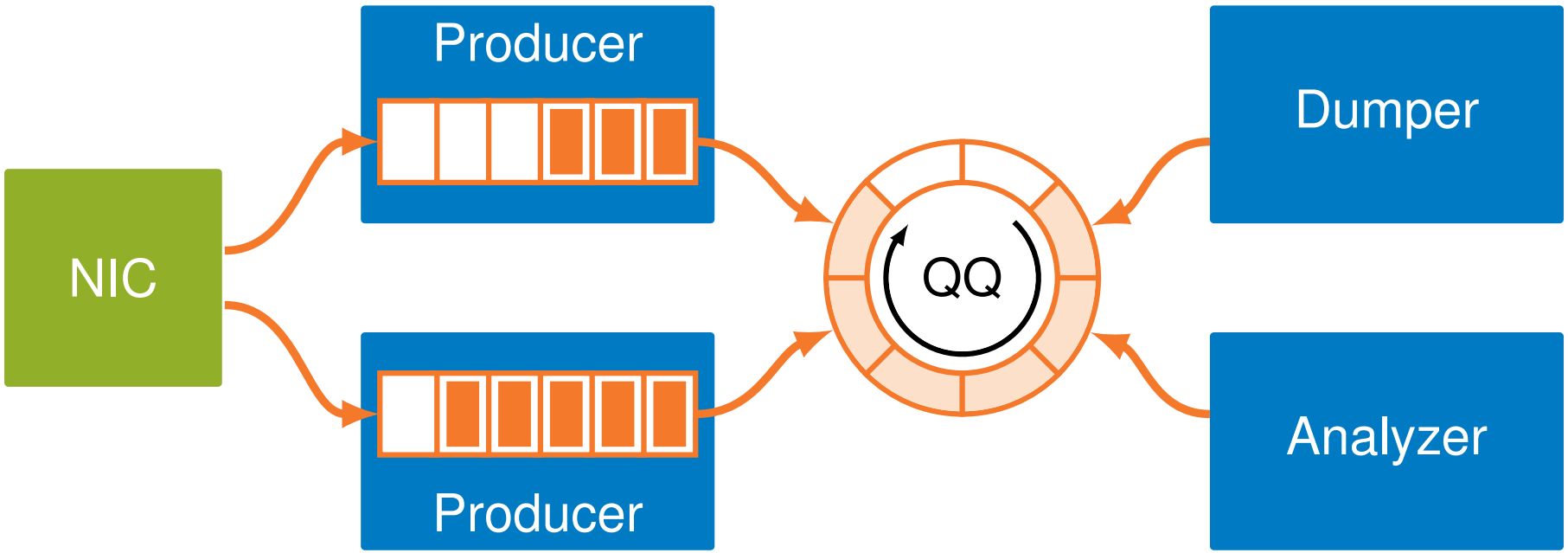
**Key features**

- High-Performance: Recording traffic at 100 Gbit/s or more
- Flexible Filtering: Monitoring/analyzing the traffic with Lua scripts or pcap filters
- Scalability: Multi-thread support by employing separate queues for writers
- Specialized Data Structure (QQ): Designed for high-throughput rather than low latency

P. Emmerich, M. Pudelko, S. Gallenmüller, et al., „FlowScope: Efficient Packet Capture and Storage in 100 Gbit/s Networks“, in Proceedings of the 16th International IFIP TC6 Networking Conference, IEEE, 2017

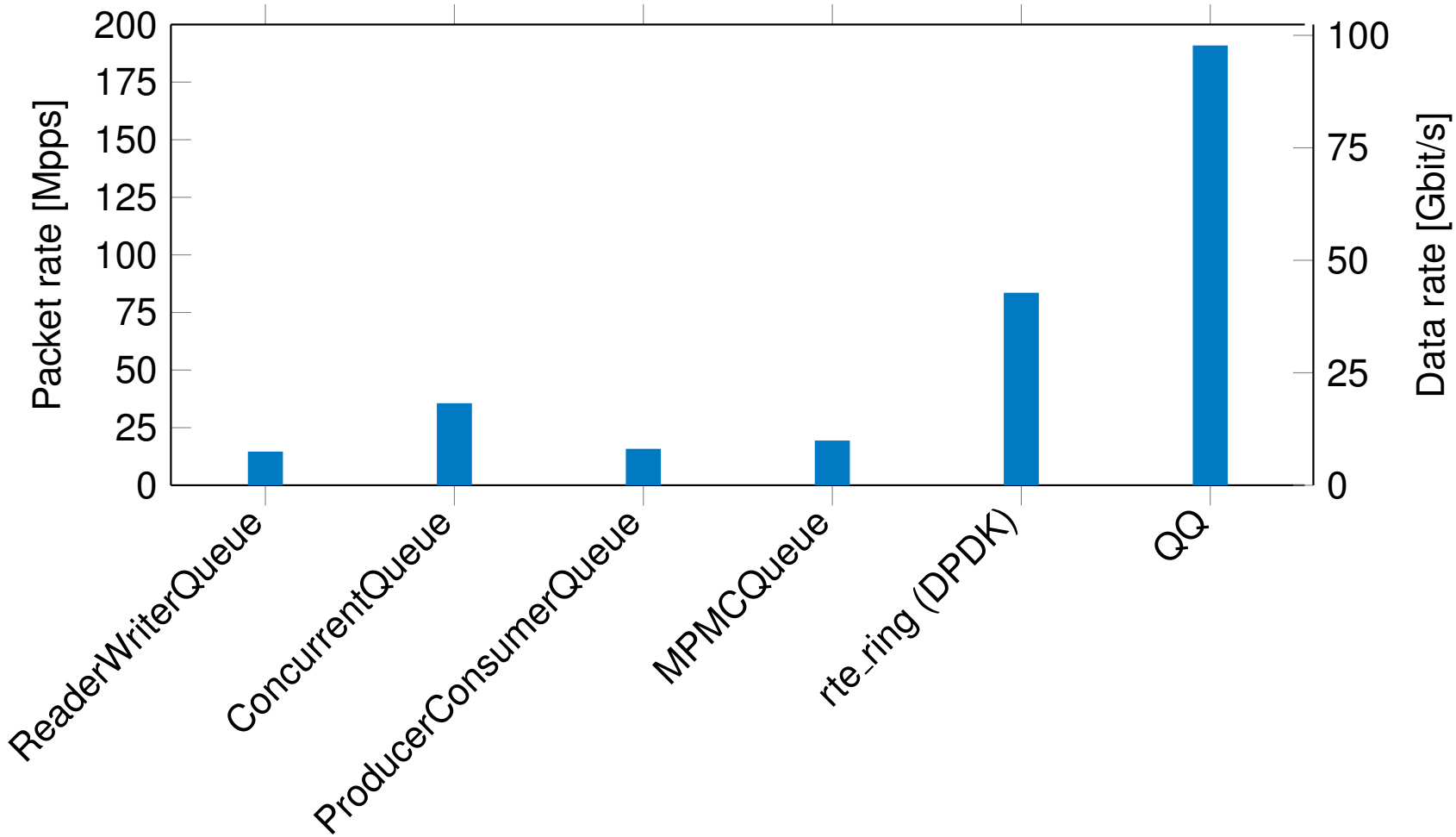
# FlowScope – High-Performance Traffic Analysis

## Architecture of FlowScope



# FlowScope – High-Performance Traffic Analysis

## Comparison of QQ to Other Queues



# Other Example Running on MoonGen/libmoon

| Name | Usage scenario | Publication |
|---|---|---|
| **High-performance applications:** | | |
| FlowScope | Tool for high-performance flow capture and analysis | [11], [12] |
| MoonRoute | Extensible high-performance router | [4], [13] |
| **Benchmarking tools:** | | |
| RFC 2544 | Modular benchmarking tool | [14], [15] |
| OPNFV VSPERF | Automated NFV testing framework | [16], [17] |
| FLOWer | High-performance switch benchmarking | [18], [19] |
| **Traffic & packet generation:** | | |
| DNS flood query generator | DNS implementation and flooding attack tool | [20], [21] |
| NFVnice | Throughput and latency measurements | [22] |
| Verified NAT | Throughput and latency measurements | [23] |
| PISCES | Throughput measurements | [24], [25] |
| **MoonGen / libmoon under test:** | | |
| MoonGen investigation | Precise and accurate rate control and timestamping | [3], [26], [27] |
| MoonGen timestamping | Investigation of timestamping for packet generators | [28] |
| **Additions to MoonGen / libmoon:** | | |
| MoonStack | Easy-to-use and efficient packet creation | [29] |

# Other Example Running on MoonGen/libmoon


[1] Q. Scheitle, M. Wählisch, O. Gasser, T. C. Schmidt, and G. Carle, „Towards an Ecosystem for Reproducible Research in Computer Networking“, in Proceedings of the Reproducibility Workshop, Reproducibility@SIGCOMM 2017, Los Angeles, CA, USA, August 25, 2017.

[2] P. Emmerich, S. Gallenmüller, D. Raumer, F. Wohlfart, and G. Carle, „MoonGen: A Scriptable High-Speed Packet Generator“, in Internet Measurement Conference (IMC) 2015, IRTF Applied Networking Research Prize 2017, Tokyo, Japan, Oct. 2015.

[3] P. Emmerich, S. Gallenmüller, G. Antichi, A. W. Moore, and G. Carle, „Mind the Gap – A Comparison of Software Packet Generators“, in ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS 2017), Beijing, China, May 2017.

[4] S. Gallenmüller, P. Emmerich, R. Schönberger, D. Raumer, and G. Carle, „Building Fast but Flexible Software Routers“, in Proceedings of the Symposium on Architectures for Networking and Communications Systems, IEEE Press, 2017, pp. 101–102.

[5] 6WIND, 6WIND Turbo Router, `http://www.6wind.com/products/6wind-turbo-router/`, Accessed: 2017-01-21.

[6] J. Kim, S. Huh, K. Jang, K. Park, and S. Moon, „The power of batching in the click modular router“, in Proceedings of the Asia-Pacific Workshop on Systems, ACM, 2012, p. 14.

# Other Example Running on MoonGen/libmoon


[7] O. Cochard-Labbé, fbsd11-routing.r287531, https://github.com/ocochard/netbenches/tree/master/Xeon_E5-2650-8Cores-Chelsio_T540-CR/fastforwarding-pf-ipfw/results/fbsd11-routing.r287531, Accessed: 2017-01-21, 2015.

[8] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, „RouteBricks: exploiting parallelism to scale software routers“, in Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, ACM, 2009, pp. 15–28.

[9] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, „The Click Modular Router“, ACM Trans. Comput. Syst., vol. 18, no. 3, pp. 263–297, Aug. 2000, ISSN: 0734-2071. DOI: 10.1145/354871.354874. [Online]. Available: http://doi.acm.org/10.1145/354871.354874.

[10] B. R. Project, Forwarding performance lab of an IBM System x3550 M3 with 10-Gigabit Intel 82599EB, http://bsdrp.net/documentation/examples/forwarding_performance_lab_of_an_ibm_system_x3550_m3_with_10-gigabit_intel_82599eb, Accessed: 2017-01-21, 2015.

[11] P. Emmerich, M. Pudelko, S. Gallenmüller, and G. Carle, „FlowScope: Efficient Packet Capture and Storage in 100 Gbit/s Networks“, in Proceedings of the 16th International IFIP TC6 Networking Conference, IEEE, 2017.

[12] FlowScope, https://github.com/emmericp/FlowScope.

# Other Example Running on MoonGen/libmoon


[13] MoonRoute, https://github.com/emmericp/MoonRoute-data/tree/51333dc648ca42f3740f6d09895e1ad4a9f67d69.

[14] D. Raumer, S. Gallenmüller, F. Wohlfart, P. Emmerich, P. Werneck, and G. Carle, „Revisiting Benchmarking Methodology for Interconnect Devices“, in The Applied Networking Research Workshop 2016 (ANRW '16), Berlin, Germany, Jul. 2016.

[15] „RFC 2544 benchmark tool“, Tech. Rep., https://github.com/emmericp/MoonGen/pull/98.

[16] „OPNFV VSPERF“,, http://artifacts.opnfv.org/vswitchperf/docs/index.html.

[17] „VSPERF code repository“,, https://git.opnfv.org/vswitchperf.

[18] P. Emmerich, S. Gallenmüller, and G. Carle, „FLOWer – Device Benchmarking Beyond 100 Gbit/s“, in IFIP Networking Conference (IFIP Networking) and Workshops, 2016, IEEE, 2016, pp. 109–116.

[19] FLOWer code, https://github.com/emmericp/FLOWer-scripts/tree/c5bd7cb25c3da1537dad7a44db84d043b442bbb9.

[20] S. R. Rincón, S. Vaton, and S. Bortzmeyer, „Reproducing DNS 10Gbps flooding attacks with commodity-hardware“, in Wireless Communications and Mobile Computing Conference (IWCMC), 2016 International, IEEE, 2016, pp. 510–515.

[21] MoonGen DNS code, https://github.com/emmericp/MoonGen/pull/118.

# Other Example Running on MoonGen/libmoon


[22] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu, „NFVnice: Dynamic Backpressure and Scheduling for NFV Service Chains“, in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 71–84.

[23] A. Zaostrovnykh, S. Pirelli, L. Pedrosa, K. Argyraki, and G. Candea, „A Formally Verified NAT“, in Proceedings of the Conference of the ACM Special Interest Group on Data Communication, ACM, 2017, pp. 141–154.

[24] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, „PISCES: A Programmable, Protocol-Independent Software Switch“, in Proceedings of the 2016 conference on ACM SIGCOMM 2016 Conference, ACM, 2016, pp. 525–538.

[25] PISCES experiment code, https://github.com/P4-vSwitch/vagrant/tree/ef197c77505252d1b255c3f1c83b976aae5d7fb7.

[26] MoonGen rate control methods, https://github.com/emmericp/MoonGen/blob/5388192fa1dc016797fabe8912bbcdfc7713756e/examples/rate-control-methods.lua.

[27] MoonGen timestamping, https://github.com/emmericp/MoonGen/blob/5388192fa1dc016797fabe8912bbcdfc7713756e/examples/timestamping-tests/.

[28] M. Primorac, E. Bugnion, and K. Argyraki, „How to Measure the Killer Microsecond“, in Proceedings of the Workshop on Kernel-Bypass Networks, ACM, 2017, pp. 37–42.

# Other Example Running on MoonGen/libmoon

[29] MoonStack, https://github.com/libmoon/libmoon/tree/f3013c9ced5d0e4f5344451d2d269233852eb96c/lua/proto.