Precise Rate Control for Software Packet Generators

Paul Emmerich, Sebastian Gallenmüller, and Georg Carle
Technische Universität München, Department of Computer Science
Chair for Network Architectures and Services
{emmericp|gallenmu|carle}@net.in.tum.de

Keywords

Packet generation; Traffic pattern; Interpacket gaps

1. INTRODUCTION

Precise control of interpacket gaps is an important feature of a packet generator as it is required to produce different traffic patterns. However, achieving the required precision is a challenging problem for software packet generators [1].

We evaluate existing software approaches and quantify their shortcomings. We then present a novel method for precise control of interpacket gaps in software.

https://youtu.be/hnxrnQY4zvI shows our demo.

1.1 Packet Generation in MoonGen

This work is based on our free and open source packet generator MoonGen[1]. MoonGen is a scriptable high-speed packet generator based on DPDK[2] and LuaJIT[3]. Beside the rate control feature discussed here, it also features latency measurements with sub-microsecond precision and accuracy by (mis-)using hardware features found on commodity NICs.

Interested readers are referred to a draft of our full paper about MoonGen [2]. A short discussion of MoonGen's architecture was presented as a poster/demo at NSDI 2015 [3]. This paper focuses on rate control in MoonGen.

2. SOFTWARE RATE CONTROL

Packet generators built on a traditional networking API of an operating system face a multitude of challenges including timing problems due to system calls and context switches which limits their precision significantly [1]. Software built on modern userspace networking frameworks like DPDK[2] or PF_RING ZC[4] eliminate most of these by moving the whole driver into the userspace.

[1]www.github.com/emmericp/MoonGen
[2]www.dpdk.org
[3]www.luajit.org
[4]www.ntop.org/products/pf_ring/pf_ring-zc-zero-copy

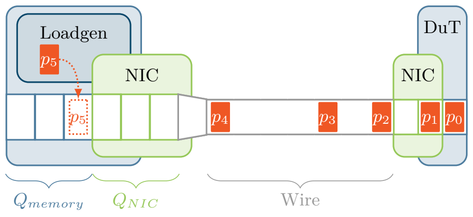

Figure 1: Software-based rate control

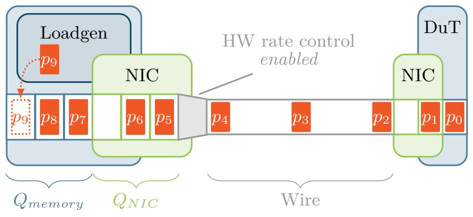

Figure 2: Hardware-based rate control

However, packet generators still face a fundamental problem when trying to insert a gap between packets. For example, Pktgen-DPDK[5] and zsend[6] simply wait for a certain time in a busy-wait loop. This assumes that the driver can push a packet to the NIC and send it immediately.

However, NICs use an asynchronous push-pull model [5]. Sending a packet only places it in a queue in main memory, the NIC fetches it asynchronously via DMA. The whole transmit operation includes two round-trips across the PCIe bus and a delay between informing the NIC about the new packet and the NIC fetching it. These delays add uncertainties to the operation and limit the precision.

Figure 1 shows this concept. Another drawback is that neither the queue in the main memory Q_{memory} nor the transmit queue Q_{NIC} on the NIC can be utilized as all packets in a queue would be sent out back-to-back. So a batch size of 1 must be used, this affects performance as batch processing is crucial for high-speed packet IO.

3. HARDWARE RATE CONTROL

Modern NICs, e.g. the Intel X540 used here, often support rate control in hardware. However, this feature is limited to constant bit-rate (CBR) traffic as it is designed to control bandwidth in multi-tenant virtualization environments and not for packet generators [5]. Figure 2 shows that this approach allows utilizing the queues.

[5]www.github.com/pktgen/Pktgen-DPDK
[6]An example application of PF_RING ZC

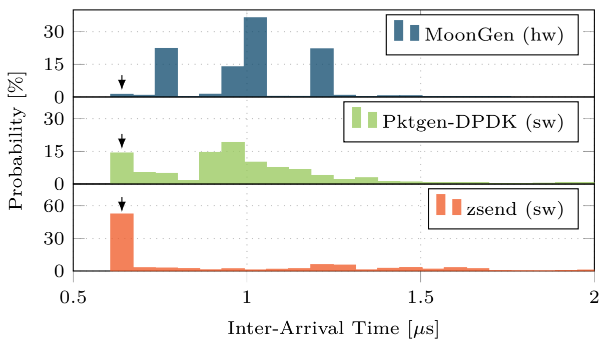

Figure 3: Histograms of inter-arrival times

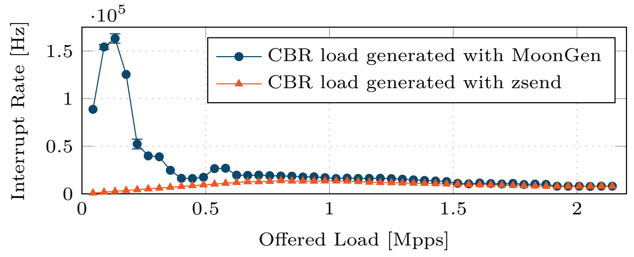

Figure 4: Interrupt rate with micro-bursts

MoonGen is, to our best knowledge, the first publicly available software packet generator that utilizes this feature for CBR and bursty traffic (by varying the rate control parameter periodically).

4. EVALUATION

We compare the two approaches by configuring the packet generators Pktgen-DPDK, zsend, and MoonGen (with hardware rate control) to generate 1 Mpps of 64 byte packets on a 1 Gbit/s link. Figure 3 shows the histograms (64 ns bins) of the inter-arrival times measured with an Intel 82580 which supports timestamping all received packets [4]. The expected inter-arrival time is 1 μs, the leftmost bucket marked with a black arrow contains back-to-back frames on the wire.

This measurement shows that software rate control already fails at relatively low rates. The hardware rate control of the Intel X540 NIC used here also oscillates around the target rate, but it avoids generating micro-bursts. zsend, on the other hand, generates bursty traffic, even though we explicitly configured it to generate CBR traffic.

Bursts have a measurable impact on the device under test (DuT), Figure 4 shows the interrupt rate of an Open vSwitch packet forwarder. The bursts trigger the interrupt moderation feature of the driver too early. Bad software rate control causes undesirable behavior in a tested system and affects the validity and reproducibility of experiments.

5. NOVEL SOFTWARE RATE CONTROL

We were unsatisfied with the precision of the software approach and the flexibility of the hardware solution on commodity NICs. Therefore, we present a new solution here: instead of generating gaps, we fill them with invalid packets. This allows us to produce arbitrary traffic patterns with a precision of 0.8 ns (length of a byte at 10 Gbit/s), this is even better than hardware rate control on the X540 NIC.

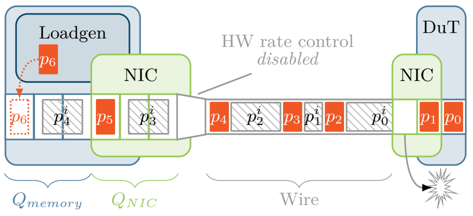

Figure 5: Arbitrary traffic patterns in MoonGen

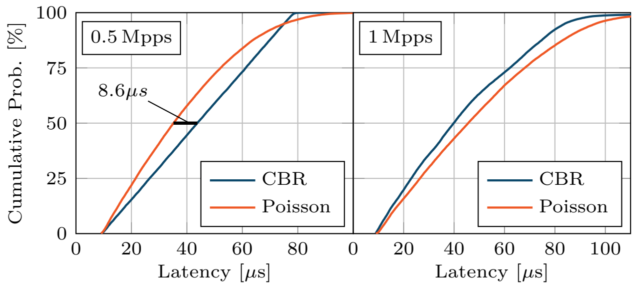

Figure 6: Latency of Open vSwitch with CBR and poisson traffic (cumulative density function)

Figure 5 shows this concept. Shaded packets p^i are sent with an invalid CRC checksum and an illegal length if necessary. The DuT drops these packets in hardware on reception without affecting the forwarding engine. We have tested and verified this with various NICs like the X540 used here. If a DuT is affected by invalid packets, a switch can be used to discard the packets before they reach the DuT.

Figure 6 shows the forwarding latency of Open vSwitch with CBR and poisson traffic at 0.5 Mpps and 1 Mpps. The median latency differs by 20% at a packet rate of 0.5 Mpps. This experiment shows that our approach can generate different traffic patterns with a measurable impact on the DuT. One reason for this influence are interrupt rate moderation algorithms on the DuT (cf. Figure 4).

Generating the CBR traffic in hardware and with our approach yields a virtually identical result ($\leq$ 2% deviation in median latency at all packet rates [2]), so the invalid packets do not affect the DuT.

6. LIVE DEMO

We will present a live demo of MoonGen's rate control features. Our demo loads a DuT with configurable traffic patterns and shows a live histogram of the latencies to visualize how a DuT responds to different traffic patterns. The video at https://youtu.be/hnxrnQY4zvI shows our demo.

7. REFERENCES

[1] Alessio Botta, Alberto Dainotti, and Antonio Pescapé. Do you trust your software-based traffic generator? *IEEE Communications Magazine*, 48(9):158–165, 2010.

[2] Paul Emmerich et al. MoonGen: A Scriptable High-Speed Packet Generator. `http://go.tum.de/566578`, 2015. *Draft*.

[3] Sebastian Gallenmüller et al. MoonGen: Software Packet Generation for 10 Gbit and Beyond. In *USENIX NSDI*, 2015.

[4] Intel 82580EB Gigabit Ethernet Controller Datasheet.

[5] Intel Ethernet Controller X540 Datasheet.

Demo Requirements

We will bring our portable testbed for the live demo.

Equipment to be Used

- Packet generator server: Barebone Mini-ITX PC
- Device under test 1: Barebone Mini-ITX PC
- Device under test 2: MikroTik 10 Gbit/s router
- Laptop
- Small projector

Space Needed

A table with a length of 2 meters should be sufficient since our portable testbed is relatively small.

We also need a surface to project the GUI of our live demo onto, ideally beside the poster. If this is not possible, then we require a large display, e.g. a TV, to show our demo in an appealing way.

Setup Time Required

Approximately 20 minutes.

Additional Facilities

Power supply:

Component	**Power**
Barebone PC 1	100 W
Barebone PC 2	100 W
MikroTik Router	80 W
Laptop	30 W
Small projector	90 W
Total	**400 W**

Some additional experiments and examples may be shown on our remote testbed. Therefore, we need a stable Internet connection, preferably via a cable.