HyperGen: High-Performance Flexible Packet Generator Using Programmable Switching ASIC

Zhaowei Xi, Yu Zhou, Dai Zhang, Jinqiu Wang, Sun Chen, Yangyang Wang, Xinrui Li, HaoMing Wang, Jianping Wu*
Tsinghua University

CCS CONCEPTS

• **Networks → Programmable networks**;

KEYWORDS

Programmable Switching ASIC, P4, Packet Generator

ACM Reference format:
Zhaowei Xi, Yu Zhou, Dai Zhang, Jinqiu Wang, Sun Chen, Yangyang Wang, Xinrui Li, HaoMing Wang, Jianping Wu. 2019. HyperGen: High-Performance Flexible Packet Generator Using Programmable Switching ASIC . In *Proceedings of SIGCOMM '19: ACM SIGCOMM 2019 Conference, Beijing, China, August 19–23, 2019 (SIGCOMM '19),* 3 pages. https://doi.org/10.1145/3342280.3342301

1 INTRODUCTION

Packet generators are vital in network researches and network operations. For network researchers, packet generators serve as essential tools to test the performance of researching prototypes. Network operators can also use packet generators to produce test traffic for latency measurement and failure troubleshooting [6]. Meanwhile, the development of today's network raises demands on packet generators from two perspectives. One is the ability of high performance traffic generation to meet the increasing network bandwidth (from 10GbE to 100GbE), the other is the ability of flexible packet customization with given properties to meet the constant appearance of new protocols and functions.

Existing packet generators can be categorized into hardware approaches and software approaches. Commodity packet generators based on proprietary hardware can achieve high rate packet generation up to above 1Tbps per device. Nevertheless, hardware packet generators [1] have limited flexibility. For instance, it is hard for users to customize proprietary hardware to test new protocols or functions. Besides, hardware packet generators are typically expensive, and a two 10GbE port packet generation module can cost as much as $25,000 [2]. Software approaches [5], on the contrary, enable users to customize packet generation logic, yielding high flexibility. However, software approaches have to make an undesirable trade-off between performance and cost. A 8-core commodity server can only achieve less than 100Gbps [5]. If operators have higher requirements on generation rates, more servers are needed, which makes the cost rise rapidly.

Table 1: Comparison of current approaches.

Metrics (1Tbps)	Device Required	Flexibility	Cost
Hardware	1	limited	$100,000
Software	≈ 10	high	$30,000
HyperGen	1	**high**	**$3,000**

Motivated by the limitations of existing approaches, we present ***HyperGen***, a high-performance, flexible, and low-cost packet generator using programmable switching ASIC in this paper. HyperGen can produce above 1Tbps traffic, and supports rate control with high accuracy and high precision. Users can flexibly reconfigure HyperGen to execute various tasks like throughput testing, latency measurement, denial-of-service attack emulation and so on. HyperGen can be implemented in a single programmable switch whose cost per Tbps is much lower than the proprietary hardware and mid-range server. Table 1 summarizes the comparison between HyperGen and existing approaches.

The design of HyperGen is non-trivial, and the main challenge is that there is no straight way to generate high rate, task-specific traffic in programmable switching ASICs that are limited in programmablity and resources. We solve the challenge from two perspectives. Firstly, we present template-based packet generation and take advantage of switch CPU to improve the flexibility of the switching ASIC. More concretely, we use switch CPU to generate the control logic and template packets with predefined properties (*e.g.*, packet size and payload), which are hard for the switching ASIC to customize. Secondly, we present a new pipeline design to improve the capacity of packet generation in the switching ASIC. The pipeline performs a series of operations sequentially including acceleration, replication and edition on template packets to generate testing packets for various tasks.

2 DESIGN

Figures 1 shows the architecture overview of HyperGen. The workflow can be described into two steps:

(1) Compiling tasks to generate testing packets. HyperGen compiles testing tasks into three elements: switch configurations (*e.g.*, port settings and table entries), template packets, and a P4[4] program. It is hard for the switching ASIC to customize some properties (*e.g.*, packet size and payload). Therefore, instead of directly generating packets in the switching ASIC, we present template-based packet generation that generate template packets with predefined properties in switch CPU, and forward template packets to the switching ASIC for further processing. The P4 program defines the packet generation logic in the switching ASIC, which can be

*This work is supported by National Key R&D Program of China (2017YFB0801701) and the National Science Foundation of China (No.61872426).

SIGCOMM '19, Beijing, China
ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342301

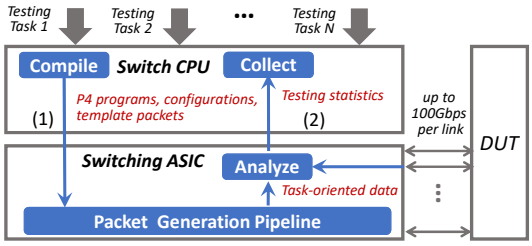


Figure 1: Architecture overview of HyperGen.

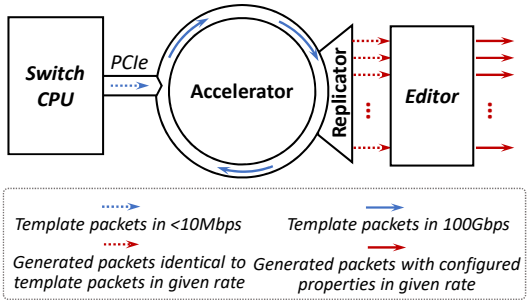


Figure 2: Design of Packet Generation Pipeline.

described as the packet generation pipeline. §2.1 presents the design of the pipeline.

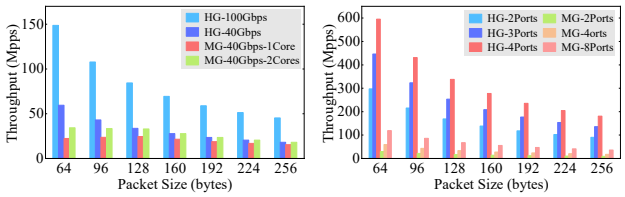
(2) Testing statistics analysis and collection. HyperGen can analyze and record task-oriented data while generating packets and receiving packets from devices under test (DUT). Recorded data can be per-packet data like a particular field value, or statistical properties like packet count, throughput, delay and so on. Switch CPU could attain aforementioned data via pulling from data plane counters or receiving digests pushed by the switching ASIC. Then, switch CPU will feed back testing statistics to testing tasks.

2.1 Packet Generation Pipeline

Packet generation pipeline consists of three components that execute operations sequentially on template packets. Figure 2 shows the pipeline design. We will present the accelerator and replicator in detail in the following parts, since the editor's main function is to modify generated packets from replicator to produce specific traffic(*e.g.*, choosing forwarding port, using specific fields for state storage), and all operations needed are well supported by P4-programmable hardware.

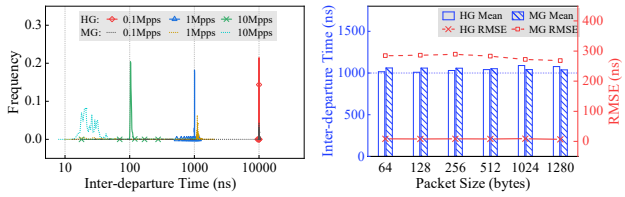
Accelerator. Using the accelerator, we can accelerate template packets from switch CPU to 100GbE full line rate in negligible time. The accelerator forwards template packets into a loop via *recirculation*, which is a general primitive supported by P4-programmable hardware. Our experiments testify that Tofino [3] can recirculate packets at a speed of no less than 100Gbps, and round trip time of 64-byte template packets is about 600ns, which means the maxmium number of template packets that one loop-back port can accelerate sequentially is nearly one hundred and increasing loop-back ports linearly improves the maxmium number. Through recirculation, the accelerator act as a stable high rate packet source for the replicator.

Replicator. Taking the looping template packets in the accelerator as input, the replicator performs conditional packet replication at given rates. The replicator implements rate control via adjusting



(a) Single-port rate (b) Multi-port rate

Figure 3: Packet generation rate in Mpps.



(a) Inter-departure time distribution (b) Precision

Figure 4: Accuracy and precision comparison.

inter-departure time. More concretely, the replicator maintains a nanosecond-level periodic timer and performs replication as soon as the timer expires. Through changing the timer threshold, the replicator manages to adjust inter-departure time of replication. The replicator implements replication on template packets through *multicast*, which is a general primitive supported by switching ASIC and can replicate packets to multiple ports simultaneously.

3 EVALUATION

We implement a prototype of HyperGen (*HG*) on Wedge 100BF32X equipped with Tofino and 32 100GbE ports. We use 1 port for recirculation and 4 ports for multicasting in our testbed for the consideration of reserving ports for other use in practical circumstance. More ports for recirculation increases the maxmium supported number of template packets and more ports for multicasting guarantee higher performance. We use a server with 64GB RAM and 8 2.10GHz CPU cores to run MoonGen (*MG*) [5] as the countermeasure. We evaluate HyperGen with the following two objectives.

Packet generation rate. Figure 3 shows the experiment results. We compare HyperGen and MoonGen's rate under different packet sizes using single-port (Figure 3(a)) and muti-port configurations (Figure 3(b)). As the results show, in our testbed HyperGen can achieve 4-port 100GbE full line rate, which is higher than MoonGen.

Accuracy and precision. We configure HyperGen and MoonGen to generate 64-byte packets in different rates. HyperGen keeps accurate at all rates, while MoonGen becomes inaccurate when rate grows high (Figure 4(a)). Besides, we test the average inter-departure time of different packet sizes at a rate of 1Mpps. HyperGen has better accuracy when packet size is relatively small, and has much lower Root Mean Squared Error(RMSE) (Figure 4(b)).

4 FUTURE WORK

Our future work will focus on further analyzing generated packets and building a general network testing system. Besides, developing a convenient way to specify network testing intents of operators is another important work we keep studying.

REFERENCES

[1] A Keysight Business. 2019. IXIA test modules. Website. (2019). https://www.ixiacom.com/products.

[2] Gianni Antichi, Muhammad Shahbaz, Yilong Geng, Noa Zilberman, Adam Covington, Marc Bruyere, Nick McKeown, Nick Feamster, Bob Felderman, Michaela Blott, and others. 2014. OSNT: Open source network tester. *IEEE Network Magazine* 28, 5 (2014), 6–12.

[3] Barefoot Networks. 2019. Barefoot Tofino Switch. Website. (2019). https://barefootnetworks.com/technology/.

[4] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and others. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[5] Paul Emmerich, Sebastian Gallenmüller, Daniel Raumer, Florian Wohlfart, and Georg Carle. 2015. Moongen: A scriptable high-speed packet generator. In *Proceedings of the 2015 Internet Measurement Conference*. ACM, 275–287.

[6] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, and others. 2015. Pingmesh: A large-scale system for data center network latency measurement and analysis. *ACM SIGCOMM Computer Communication Review* 45, 4 (2015), 139–152.