Accepted Manuscript

A scalable and accurate distributed traffic generator with Fourier transformed distribution over multiple commodity platforms

Ching-Hao Chang, Ying-Dar Lin, Yu-Kuen Lai, Yuan-Cheng Lai

PII: S1084-8045(19)30225-5

DOI: https://doi.org/10.1016/j.jnca.2019.07.001

Reference: YJNCA 2400

To appear in: *Journal of Network and Computer Applications*

Received Date: 15 November 2018

Revised Date: 1 May 2019

Accepted Date: 2 July 2019

Please cite this article as: Chang, C.-H., Lin, Y.-D., Lai, Y.-K., Lai, Y.-C., A scalable and accurate distributed traffic generator with Fourier transformed distribution over multiple commodity platforms, *Journal of Network and Computer Applications* (2019), doi: https://doi.org/10.1016/j.jnca.2019.07.001.

A Scalable and Accurate Distributed Traffic Generator with Fourier Transformed Distribution over Multiple Commodity Platforms

Ching-Hao Chang[a], Ying-Dar Lin[a], Yu-Kuen Lai[b], Yuan-Cheng Lai[c]

[a]*Department of Computer Science, National Chiao Tung University, Hsinchu County 300, Taiwan*
[b]*Department of Electrical Engineerin, Chung Yuan Christian University, Taoyuan City 320, Taiwan*
[c]*Department of Information Management, National Taiwan University of Science and Technology, Taipei County 106, Taiwan*

Abstract

The rapid growth of high-speed computer networks poses a challenge in the design of testing and verification equipment. The hardware-based packet generators that are often used in the verification process are accurate but costly. Software-based packet generators, on the other hand, are relatively low-cost but have a limited performance with low accuracy. This paper proposes a Fourier-based profile decomposition and formulation methodology for a distributed packet generating system, featuring good horizontal scalability and high accuracy. Different from traditional software-based packet generators, this proposed system extracts the traffic components from a specific traffic distribution applying Fourier transformation to generate traffic components. These traffic components are distributed to one or more worker nodes for packet generation, and thus achieving higher aggregated traffic rate in any given distribution. The system design is based on the Data Plane Development Kits (DPDK) framework to maximize the traffic generation performance. The accuracy and performance of the proposed system scale according to the number of worker nodes used. Currently, with multiple CPU cores and five workers, the proposed system can generate aggregated traffic of more than 40 Gbps in a Poisson distribution.

Keywords: Packet Generator, DPDK, Fourier Transformation, Commercial off-the-shelf Packet Generator.

1. Introduction

The latest trend in network bandwidth is shifting from 1 Gbps towards 10 Gbps in access networks and is rapidly moving toward 100 Gbps and beyond in the core networks and data centers.

According to the timeline provided by the IEEE P802.3cd Task Force [3], the standard of the optical and electrical signaling of 50 Gbps for both 200 Gbps and 400 Gbps [4] transmission rates is scheduled to be released by the end of 2018. As the development of new testing tools often lags behind, there is a strong demand for measurement tools and testing instruments that can provide an accurate evaluation of the performance of network equipment [14] while satisfying the growing demand of increasing bandwidths.

Packet generators, commonly used to generate synthetic traffic for performance evaluation on network equipment, are implemented for both hardware and software platforms. Hardware-based packet generators are built with specific ASIC chips where precision and performance are optimized, but it has a high price and limited flexibility. Commonly seen models of hardware-based packet generators are manufactured by Spirent [7], IXIA [5] and XENA [10]. Hardware-based packet generators are designed to generate predefined packet streams and perform network device validation according to RFC2544 [20]. Software-based packet generators, on the other hand, are accessible and have a much lower cost. Most of the software-based packet generators are open-source and can be readily downloaded, built and executed on commodity personal computers and servers [36]. Nevertheless, the accuracy of software-based packet generators is often not good enough, especially when conducting experiments that require the generation of a high packet rate and achieving accurate traffic profiles [19].

1.1. Motivation

It is known that the performance of state-of-the-art software-based packet generators has improved in various ways [32]. For example, Pktgen [28], a packet generator that runs in the kernel-space instead of userspace to increase the performance of packet generation. Pktgen-DPDK [2] developed by Wind-river using the DPDK [9] framework to provide line-rate traffic generation and receiving. MoonGen [22] is another packet generator based on the DPDK that support fast per packet customization with LuaJIT [6]. A. Botta *et al.* proposed the D-ITG [16], a distributed packet generator that generates and receives packets from multiple nodes. However, there are no software-based

packet generators which are capable of generating traffic profiles with various stochastic-processes in a distributed fashion.

This has motivated us to design a packet generator which is accurate, scalable, and still highly cost-effective. Leveraging the techniques of dividing the desired traffic profile into multiple traffic components in a frequency domain, we propose a distributed packet generator system that is capable of synthesizing a specific traffic profile based on various stochastic processes. By generating packets based on each traffic component in multiple nodes, traffic can be aggregated with demanded traffic profiles.

This packet generating system consists of a controller node and one or more worker nodes. The controller node processes the desired traffic profile within a time domain. Based on the domain transformation techniques, the major traffic components in the frequency domain are extracted to establish the packet generation logic, and is then sent to the worker nodes. Based on the control messages sent from the controller, the worker nodes accept the packet generation logic and generate packets in a synchronized and distributed fashion. For each worker node, the DPDK packet processing framework, commonly used in modern high-performance networking services, is adopted to achieve the generation of highest packet rate precisely. Therefore, the performance of the packet generator can be scaled horizontally across a cluster of worker nodes, achieving very high throughput of aggregating traffic.

The main contributions of this work are summarized as follows.

- A distributed packet generator system is developed based on a controller-agent architecture.
- A novel Fourier-based profile decomposition and formulation methodology consisting steps of *domain transformation*, *traffic component selection* and *reconstruction* is developed, so that specific traffic profiles can be decomposed into multiple frequency components for remote worker nodes.
- A control message protocol is proposed based on the publish-subscribe model along with Precision Time Protocol (PTP) [30] for time synchronization among workers and controller. The controller can control remote worker nodes with accuracy in the sub-microsecond range to generate specific profiles of networking traffic at a high aggregated rate.

- The system can be extended with multiple workers on multiple commodity PCs equipped with the off-the-shelf commodity network interface cards (NICs).

The structure of this paper is as follows. Section 2 presents a general introduction to current high-speed packet processing frameworks that are used to speed up the throughput of packet generators. The common bottlenecks of such software-based packet generators are discussed. Domain transformation techniques and comparisons with other similar efforts are also discussed. Section 3 outlines the problem addressed in this paper. Section 4 describes the proposed design and implementations in terms of traffic feature extraction and reconstruction processes; Section 5 discusses the overall system architecture of the proposed design, and the testing setup and experiment results are covered in section 6. Finally, in section 7, we summarize the work presented and make suggestions regarding future work.

2. Background and Related Work

2.1. High-speed Packet Processing Frameworks

In spite of state-of-the-art CPU architecture with booming computing power, achieving full line-rate packet processing performance continues to be difficult to achieve [33, 34], without taking complex packet handling operations into consideration [23]. This is primarily due to the processing overhead of network protocol stack implemented at the kernel of the operating system. The design of the Linux network stack is optimized for an operating system which is focusing on general purpose use, rather than applications such as high-speed packet generation and capture. A majority of packet capture and analysis applications were designed based on the Pcap library with the limited scalability due to the lack of multi-core support. Bonelli *et al.* [18] proposed an extended version of the Pcap library that enables application-level parallelism. Supports of packet fan-out to the original Pcap library along with extended APIs were provided. The goal was to offload packet reception workloads to multiple cores and increase the scalability of the system.

Modern high-speed packet processing frameworks, such as Netmap [31], Intel DPDK [9], and PF_RING Zero Copy [8] brought unprecedented performance enhancement to packet processing by using a variety of techniques such as zero-copy, kernel-bypass, polling and interrupt coalescing [32]. WireCAP [35] presented two novel mechanisms of ring-buffer-pool and buddy-

group-based offloading featuring lossless zero-copy packet capture and delivery that exploit the multi-queue NICs and multi-core architecture. The conventional way of receiving and transmitting data through network interfaces involves not only DMA transactions between the NIC and kernel-space buffer, but also memory-to-memory copying between kernel-space buffers and user-space applications. High-speed packet processing frameworks eliminate such inefficiencies by allocating a user-space memory pool sharing across NICs and user-space applications. These frameworks provide a stripped-down alternative to the Linux network stack so that the user-space applications can entirely bypass the kernel, avoid the overheads induced by a kernel networking stack and manipulate a raw packet buffer directly. For example, both Intel DPDK and PF_RING Zero Copy utilize zero-copy and kernel-bypass techniques. Their performance outperforms that of Netmap, allowing fast packet generation of minimum-sized packets. We compare and summarize these general aspects of the packet processing frameworks in Table 1. Still another high-speed packet processing technology that is capable of line-rate packet processing is NetFPGA[26]. NetFPGA is an open platform which employs programmable hardware and implements the packet processing logic within the field programmable gate array (FPGA), with the host implements only the controlling software. G. Antichi *et al.* [15] proposed a system based on NetFPGA that features high precision packet generation and timestamping. Nevertheless, the cost of NetFPGA solution still far exceeds COTS solutions and the flexibility is limited compared to a pure software-based solution.

Table 1: Comparison of packet processing frameworks. PF_RING ZC is a proprietary software and requires license to be purchased.

Name	**Intel DPDK**[9]	**PF_RING ZC**[8]	**Netmap**[31]
Type	User space	User space	Kernel+User space
Hardware Dependency	High	High	Low
Transparent	No	Yes	Yes
License	BSD	Non-free	BSD
Performance	Higher	Higher	Lower

2.2. Software-based Packet Generators

Researchers and engineers widely adopt software-based packet generators [29, 36] for performing benchmarking and system validation. They can

generally be classified into three categories: application-level, flow-level, and packet-level based on the types of traffic generated.

Application-level packet generators produce traffic of a specific application protocol by emulating the protocol's behavior. This type of traffic generator is commonly utilized to generate the workload for performance evaluation for various application servers. For instance, an HTTP workload generator behaves like multiple HTTP clients and generates a massive amount of HTTP requests simultaneously to stress the loading of the web server under test. Flow-level packet generators, on the other hand, produce application-independent IP flows characterized by the number of packets, bytes transferred and flow duration. Packet-level traffic generators are the most common among the three types. This type of packet generators can produce packets not only with determined inter-packet delay (IPD) and packet size (PS), but also for any given desired stochastic distribution.

Apart from lower cost, researchers generally opt for software-based packet generators for their flexibility. Software-based packet generators are often designed to support sophisticated customization of packets, which enable users to test and verify new network protocols and services. Hardware-based packet generators, by contrast, have difficulty generating arbitrary packets and thus are not appropriate for this sort of application.

In order to evaluate the performance of targets under test and to provide reproducible test results, packet generators are supposed to be accurate. Unfortunately, this is rarely the case for software-based packet generators. There is still a trade-off between performance and flexibility, and that is probably why the bare-metal hardware-based packet generators exist in the first place.

There are several software-based packet generators available with different implementation approaches. D-ITG, developed by Avallone *et al.* [16], features multi-node deployment. It provides central management utilities to command the remote senders and receivers. It also supports various modes of packet generation based on different stochastic processes for inter-packet delay and packet size. Angrisani *et al.* [13] measure the inter-departure time of packets generated by the packet generator D-ITG with constant bitrate traffic of various inter-departure times (IDTs) configured. The experiment results show that the variability of the packet generation is mainly contributed by non-deterministic OS system calls instead of memory access time and computational time as they are almost deterministic. Besides, the experimental result shows that the variation of IDT becomes much higher when the packet rate exceeds 1000 packets per second (PPS). Pktgen-DPDK, de-

veloped by Keith Wiles *et al.* [2], is designed with Intel DPDK framework. It features 10G line-rate capability on COTS hardware. MoonGen, developed by Emmerich, Paul *et al.* [22], features LuaJIT [6] for efficient per-packet customization. It provides a novel way of generating accurate inter-packet delay by adding deliberately corrupted packets into packet batch.

Among the works listed above, Pktgen-DPDK only supports constant bit-rate packet generation. MoonGen, on the other hand, supports per-packet customization and thus can be used to generate customized traffic profile. MoonGen does not provide a mechanism for central management, and therefore lacks the ability to generate packets in a distributed manner. D-ITG provides support for distributed packet generation. However, it supports only the generation of per-flow traffic profiles. The comparisons of these various proposals are shown in Table 2.

The state-of-the-art software-based packet generator performs well with the advance of computer architecture and the support of various packet processing acceleration frameworks with traffic under 10 Gbps. Nevertheless, state-of-the-art software-based packet generators put focus on extracting the maximum performance from a single COTS server and therefore the performance is capped by the server's physical resource. This has immensely limited the traffic generator's scalability to keep up with the performance demand when the network traffic and device performance reach the scale of 40 Gbps and beyond, especially when non-uniform traffic profiles are enforced. Researchers and testers have no choice but to fall back to costly hardware-based traffic generators. In our work, we explore the problem from a different angle by viewing the clustered workers as a whole and develop a way to scale out software-based traffic generator that satisfy the need for benchmarking advanced network devices. The major advantage of our work compared to the state-of-the-art is the ability to effectively decompose a given traffic profile and reconstruct the traffic in a distributed manner, which allows us to effectively scale out and increase the maximum capacity of the system.

2.3. Short-time Fourier Transformation

The technique of domain transformation is widely used for anomaly detection, network traffic analysis and measurement [17, 25] in the frequency domain. A short-time Fourier transform (STFT) [27] is a variation of Fourier transformation that is used to determine the magnitude, frequency and phase

Table 2: Comparison of Software-based Packet Generators

Name	**Accel.**	**10G Line-rate**	**Multi-node**	**Traffic Distribution**
D-ITG	None	No	Yes	Yes
ZSend	PFRING-ZC	Yes	No	No
Pktgen-DPDK	DPDK	Yes	No	No
MoonGen	DPDK	Yes	No	Manual

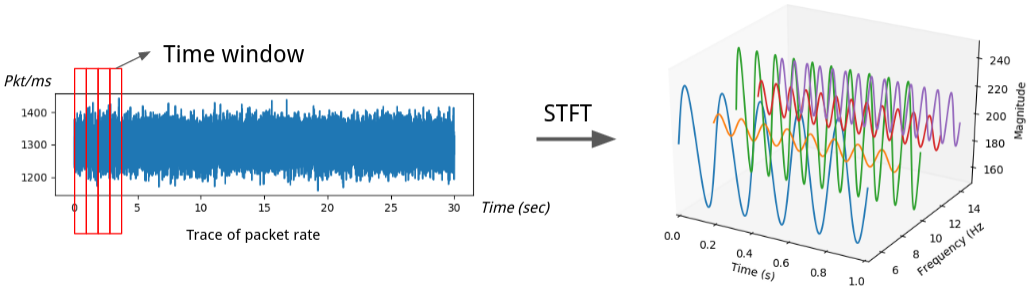


Figure 1: Short-time Fourier transform

of sinusoidal components in a short segment of a signal. This is done by slicing a long, time-based signal into multiple shorter segments with equal time intervals called windows and then to compute the Fourier transform of each window. The result is a frequency spectrum of each segment.

Short-time Fourier transform is applied in our work (as shown in Figure 1), to provide a generic methodology for analyzing the desired traffic distribution and efficiently decompose the components from the traffic. The reason why we select short-time Fourier transform over general Fourier transform is that, short-time Fourier transform provides temporal resolution while general Fourier transform provides purely frequency resolution. While there are other time-frequency domain analysis techniques such as wavelet transform, short-time Fourier transformation features simplicity of implementation and its performance can scale out effectively in multiple-worker scenario, and thus is suitable for real-time traffic reconstruction in our system.

3. Problem Statement

3.1. Notations

As shown in Table 3, the distribution types of inter-packet delay and packet size are denoted by $D^{ipd} \cdot type$ and $D^{ps} \cdot type$. We further denote the distribution of inter-packet delay and packet size by $D^{ipd}(x^{ipd}, \sigma^{ipd^2})$ and $D^{ps}(x^{ps}, \sigma^{ps^2})$ with x^{ipd}, x^{ps}, σ^{ipd2^2} and σ^{ps^2} representing the mean and variance of inter-packet delay and packet size. The requested packet count and precision is denoted by n and p while the total available worker count is denoted by m. We denote the frequency components of the traffic profile by $C_{A,f,\phi}$, with A being the amplitude, f being the frequency and ϕ being the phase. Given the $worker_j$ in set of $HOST$, D_j^{ipd}, D_j^{ps} and pc_j denote the distribution of inter-packet delay, size and packet count received from $worker_j$.

3.2. Problem Description

Given the distribution type of inter-packet delay $D^{ipd} \cdot type$ and packet size $D^{ps} \cdot type$, distribution of inter-packet delay $D^{ipd}(x^{ipd}, \sigma^{ipd^2})$ and packet size $D^{ps}(x^{ps}, \sigma^{ps^2})$, packet count n and total available worker count m, the system can derive the frequency components of traffic profile: $C_{A,f,\phi}$ of each workers $worker_j$ such that the summation of the received inter-packet delay distribution: $\sum_{j=1}^{m} D_j^{ipd}$, summation of packet size distribution $\sum_{j=1}^{m} D_j^{ps}$ and total packet count $\sum_{j=1}^{m} pc_j$ approximate to $D^{ipd}(x^{ipd}, \sigma^{ipd^2})$, $D^{ps}(x^{ps}, \sigma^{ps^2})$ and n respectively.

The object of the work is to generate a traffic profile with multi-gigabit traffic bandwidth. The design of the system is based on commodity hardware such as servers or PCs with off-the-shelf network interface cards of 10 Gbps and 40 Gbps.

As an example, shown in Figure 2, when given a traffic profile with its D^{ipd} as a Poisson distribution with mean and variance X, its D^{ps} being constant ps, the demanded test duration $t_{duration}$ and available worker $WORKER$, the system divides the traffic into multiple components and assign each of them to $worker_j$, such that the distribution of $PKTSEQ_j$: D_j^{ipd} sums up to the original distribution of requested traffic profile.

Table 3: Table of notations.

Classification	**Notation**	**Description**
Traffic Profile	$D^{ipd} \cdot type$, $D^{ps} \cdot type$	Distribution types for inter-packet delay and packet size
	$D^{ipd}(x^{ipd}, \sigma^{ipd^2})$	Inter-packet delay distribution with mean x^{ipd} and variance σ^{ipd^2}
	$D^{ps}(x^{ps}, \sigma^{ps^2})$	Packet size distribution with mean x^{ps} and variance σ^{ps^2}
	n	Packet count
	$t_{duration}$	Test duration
	t_{tos}	Test start time
	$t_{elapsed}$	Elapsed time
	t_{now}	Current time
Traffic Components	C_{A_i,f_i,ϕ_i}	The i_{th} frequency component with amplitude A_i, frequency f_i, and phase ϕ_i
	$COMP_j$	Traffic component set assigned to $worker_j$
	t_{sample}	Sampling interval
Entities	$worker_j$	The j_{th} worker
	m	Total available worker count
	$WORKER$	Worker set $WORKER = \{worker_j \mid 1 \leq j \leq m\}$
Results	D_j^{ipd}, D_j^{ps}	Inter-packet delay distribution and packet size distribution of $worker_j$
	pc_j	The count of packets from $worker_j$
	$PKTSEQ_j$	The packet sequence from $worker_j$

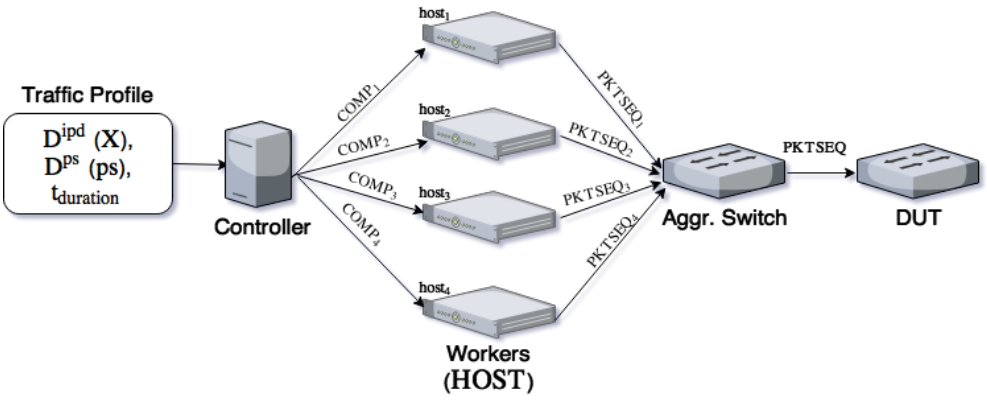


Figure 2: Packet generation example.

4. Fourier-based Profile Decomposition and Formulation

4.1. Overview

The flowchart of the proposed system is shown in Figure 3. It can be roughly divided into three main stages: domain transformation, traffic component selection, and traffic reconstruction.

The procedure for packet generation starts with the traffic profile gathered from the user input. This profile consists of the distribution type of the inter-frame gap, its duration and the stochastic properties, such as the mean and variance of the distribution. The system first carries out traffic synthesis based on the given traffic profile and produces a simulated traffic trace. By sampling the simulated traffic trace, the system can extract the packet rate changes of the actual traffic profile. Domain transformation on the packet rate changes is then carried out to obtain the frequency components of the traffic.

At the second stage, as shown in Figure 3, the most significant frequency components are selected by applying a low-pass filter to the resulting frequency components. Finally, at the third stage, the system reconstructs the traffic traces by utilizing inverse domain transformation on the remaining frequency components. The traces of the stochastic properties of the generated packet sequences then resemble the original traffic profile.

4.2. Domain Transformation on Traffic Profile

One of the core contributions of this work is the generation of distributed traffic traces with a specific profile, allowing the total bandwidth of aggre-

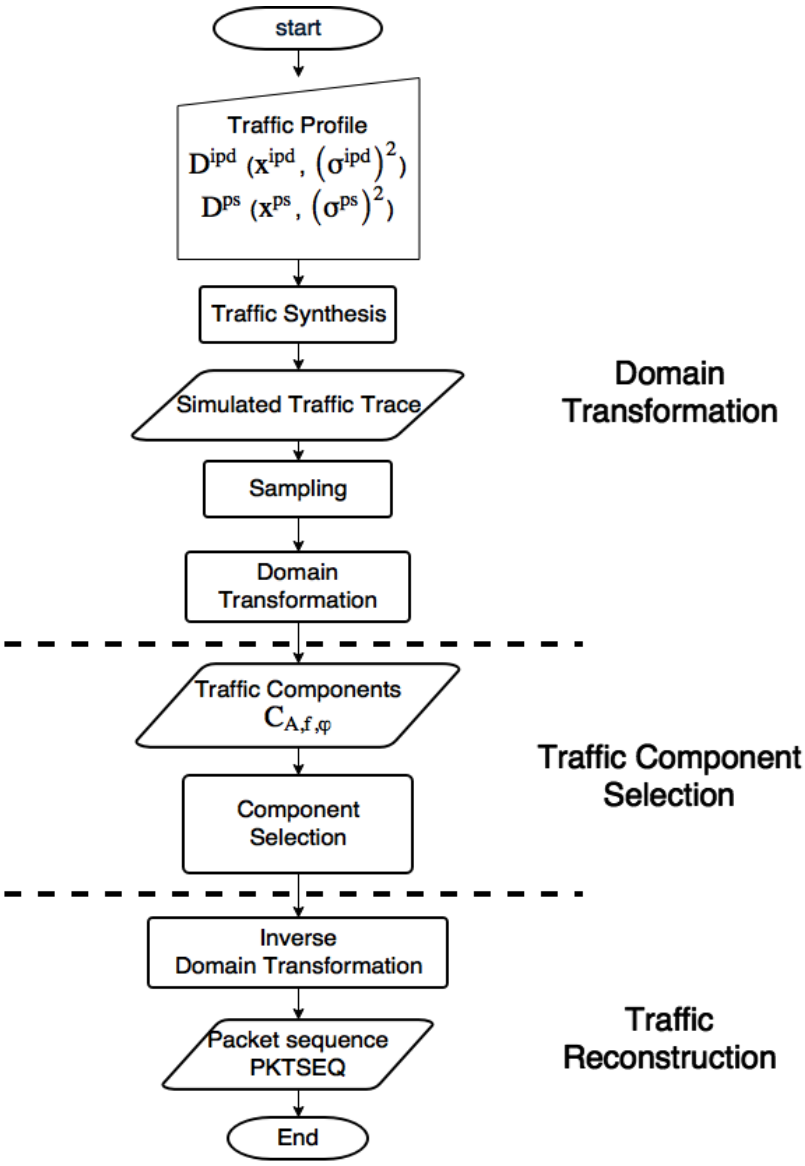


Figure 3: System flowchart.

gated traffic to scale up to multiple gigabits per second. The key to the distributed generation of a traffic profile is the decomposition of the traffic into multiple frequency components. This is achieved with the following domain transformation techniques.

Before the domain transformation process, the desired traffic profile from

the user by the distribution type of inter-packet delay ($D_{ipd} \cdot type$) is gathered along with its corresponding stochastic properties of mean (x^{ipd}) and variance (σ^{ipd^2}). These properties are then used to synthesize the desired traffic traces, or more specifically, the packet rate changes of the given traffic profile.

Notice that, in addition to the inter-packet delay distribution, the system also takes the packet size distribution ($D^{ps}(x^{ps}, \sigma^{ps^2})$) from the user. However, unlike inter-packet delay, where its accurate reconstruction is a challenging task, packet size reconstruction is rather trivial and accurate. The packet size distribution is thus not involved in the process of domain transformation.

The synthesized traffic trace is further sampled with a fixed time interval t_{sample}. The choice of sampling time t_{sample} affects the resolution of the generated traffic. Choosing a long sampling time results in aliasing of the traffic profile, while choosing a smaller one results in a significant number of data sets but higher accuracy. In this work, the default sampling time is one millisecond.

A Discrete Fourier Transform (DFT) process is then applied to analyze the simulated trace and extract the frequency components. DFT is defined as

$$A_k = \sum_{m=0}^{n-1} a_m e^{-2\pi i \frac{mk}{n}},\ k = 0, \cdots, n-1. \tag{1}$$

We specifically apply a Fast Fourier Transform (FFT), a computational-friendly variation of DFT, to speed up the process of domain transformation. Since the sampling outputs, that is, the packet rate changes, are entirely real, the component of a specific frequency is just the complex conjugate of the negative counterpart, which means there is no information in the negative frequency component that is not already available from the positive frequency components. We then use this symmetry and compute only the positive frequency components. The resulting frequency components are shown as

$$C_{A_i, f_i, \phi_i},\ i = 0, \cdots, \frac{n}{2} - 1, \tag{2}$$

where $f_{\frac{n}{2}-1}$ is the Nyquist Frequency.

4.3. Traffic Component Selection

With n sampling points, we can derive $\frac{n}{2}$ frequency components with frequency up to the Nyquist frequency. As a result, with longer duration of

generated traffic and higher sampling frequency, the amount of frequency components increases correspondingly. The processing overhead becomes larger with an increasing number of sample points n. More frequency components take up more computing power and network resources for reconstruction. In order to decrease the amount of frequency components, the system further differentiates the impact of the lower and higher frequency components. We discovered that lower frequency components determine an approximation of the traffic profile, while higher frequency components contribute to the preciseness of the counterpart. Thus, it is feasible to suppress some of the frequency components by removing higher frequency parts, since exact precision may not always be required in practice.

4.4. Traffic Reconstruction

The traffic reconstruction process is the inverse of domain transformation as shown in the first stage. In this phase, the system inversely transform the frequency components back to a time domain traffic trace. This is accomplished by applying an inverse FFT, which is defined as

$$a_m = \frac{1}{n} \sum_{k=0}^{n-1} A_k e^{2\pi i \frac{mk}{n}},\ m = 0, 1, \cdots, n-1. \tag{3}$$

The traffic reconstruction process is carried out in a distributed manner. Before the start of the generation process, the frequency components are distributed evenly to the available packet generation nodes. The frequency components are encapsulated in a task, and the duration of the generation process ($t_{duration}$) and the start time (t_{tos}) are also stored in this task. The start time is chosen to be long enough to propagate each task to each worker. At the starting time, each worker calculates the number of packets to send per t_{sample} and to put it into the sending bucket. The packet count is taken from the summation of the amplitude of each frequency component. The traffic generation flowchart of each task is shown in Figure 4. Note that the reconstructions are deliberately converted to approximate the property of sinusoidal signals. By offsetting each frequency components (except the zero frequency component), the packet rate is distributed within the range of $[0, A_i]$. This allows us to simplify the generation process and also better utilize the hardware rate control capability of network interfaces when supported.

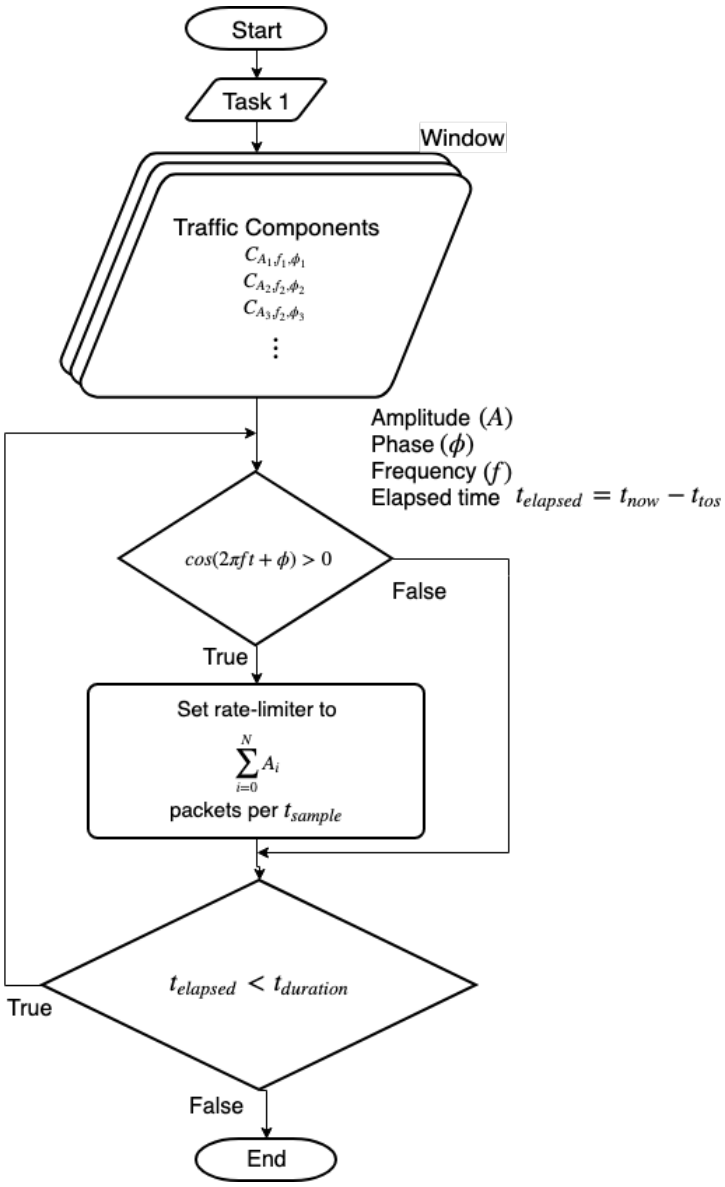


Figure 4: Flowchart of traffic reconstruction.

5. System Implementation

5.1. System Architecture

The system's architecture is shown in Figure 5 and consists of a controller and one or more workers. Each worker must be equipped with at least

two network interfaces, with one being the management port and the other being the actual packet generating ports. The workers initialize a control session with the controller through the management port during setup and perform the packet generation task under controller's command. The packet generating ports of the workers are bound to the userspace I/O (UIO) driver for high-speed packet processing with DPDK beforehand and can be directly attached to the device under test (DUT). The packet generating ports can alternatively be attached to an aggregate switch for traffic merging before the DUT.

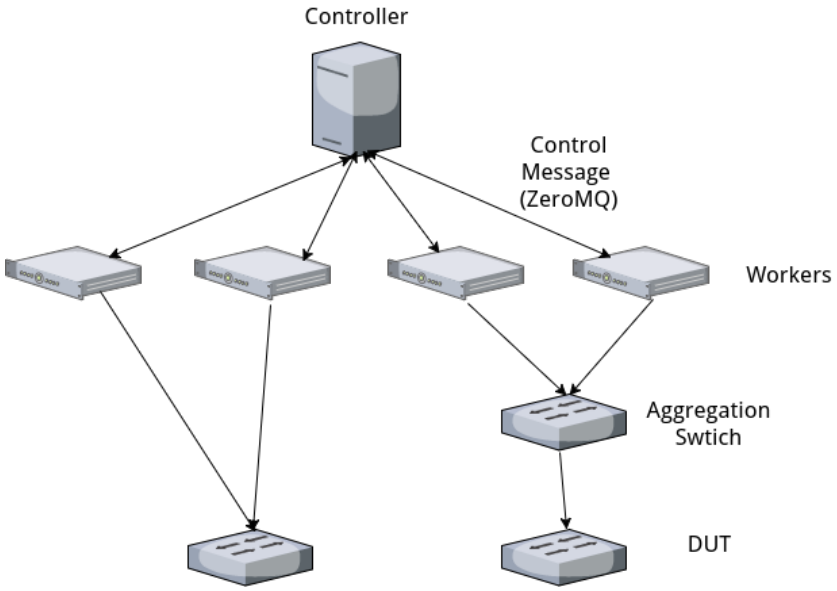


Figure 5: The proposed system architecture which consists of a controller node and four worker nodes.

5.2. Single-worker Traffic Generation

The system can be operated for a single-worker traffic generation scenario. In this mode, all of the frequency components are processed by a single worker node with limited resources. Depending on the number of packet generating ports available to the worker, each packet generating port is in charge of one or more frequency components. Currently, most of the modern network adapters are embraced with multi-queue configuration where more than one queue can be enabled on both transmission and receive side. This allows us

to generate packets with multiple CPU cores at one port. Therefore, for a worker with limited per-core computation power, multiple CPU cores can be utilized simultaneously to increase traffic throughput. Some of the 10Gbps network adapters, such as Intel 82599 and Intel X540, can even support more advanced features such as hardware-based per queue rate scheduling. Once this feature is enabled, the system can dispatch frequency components into various transmit queues with each queue set to a fixed rate of A_i. By periodically refilling the sending bucket, the system is able to craft multiple traffic streams with an accurate packet rate.

5.3. Multi-worker Traffic Generation

A multi-worker traffic generation mode, similar to that of a single-worker mode, consists of two more processes of time synchronization and port mapping. The packet generator relies on a time stamp counter (TSC)[1] which is built inside the CPU to determine when to send a packet and how many to send. In the early generation of multi-core CPU, the TSC may be used across different cores, and may even be used with SpeedStep[2] or TurboBoost[3] enabled. Modern CPU employs invariant-TSC[4], where the TSCs are synchronized across all cores and does not vary with SpeedStep or TurboBoost. Thus, for single-worker traffic generation, time synchronization would not pose a problem. This is, however, not true for multi-worker traffic generation which incorporates multiple workers with varied TSCs.

The way that the system deals with the time synchronization problem is to introduce a time correction factor t_{offset} into the generation process so that the elapsed time t becomes $t_{now} - t_{tos} + t_{offset}$. The time difference, t_{offset} between a worker and the controller, is measured based on the IEEE 1588 PTP protocol [30] with the support of hardware timestamping in the network adapter.

[1]The Time Stamp Counter (TSC) is a 64-bit register present on all x86 processors since the Pentium. It counts the number of cycles since reset. It can be read via the instruction RDTSC

[2]SpeedStep is a technology built into some Intel microprocessors that allow the clock speed of the processor to be dynamically changed by software

[3]Intel Turbo Boost is a technology implemented by Intel in certain versions of its processors that enables the processor to run above its base operating frequency via dynamic control of the processor's clock rate

[4]The invariant TSC will run at a constant rate in all ACPI P-, C-, and T-states. It is first introduced on Nehalem Intel processor

Some of the modern network interface cards such as Intel 82574 [11] and Intel 82580 [12] provides a hardware timestamping feature for PTP packets. This can significantly improve the time synchronization to sub-microsecond accuracy.

A PTP time synchronization process starts with a grandmaster that synchronizes its clock to the connected slave and boundary clocks. In principle, hardware timestamping features on the network card is used to measure the accurate jitters of the network. There is a hardware clock (PHC) within each network card. All slave NIC interfaces receive PTP packets from the grandmaster and synchronize its hardware clock to that of the grandmaster. An additional process is in charge of transforming the PHC clock to the system clock (CLOCK_REALTIME).

Figure 6 shows the port id synchronization process. To globally synchronize the port id information among all workers with the controller, the controller has to collect port id mapping from all of the worker nodes and remap them accordingly. Finally, the controller publishes the global port id map to each worker.

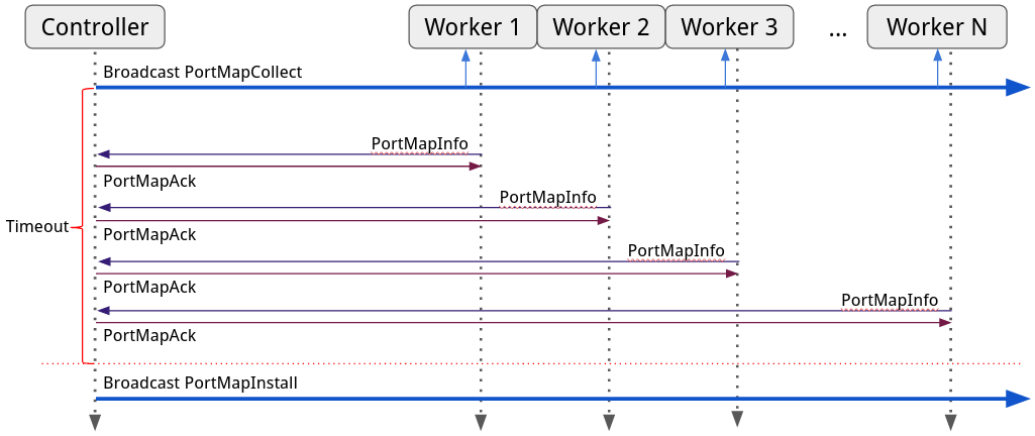


Figure 6: Port ID synchronization process.

5.4. Control Message Design

The control messages play an important role in communication between controller and workers. Therefore, the control message protocol is designed with portability and scalability in mind so that worker nodes can expand from current x86-based server to a variety of heterogeneous computing platforms.

The design uses ZeroMQ[24] and Protobuf[1] together. Both of these have ample support for various languages, and that makes them portable on many platforms. Using ZeroMQ helps us simplify the connection setup and management process, and avoids rebuilding the wheel by utilizing commonly used connection patterns. Protobuf, though adds an overhead to the protocol, helps ensure the portability of the protocol layer by its efficient and flexible serialization feature. The worker nodes and controller communicate in an out-of-band manner, in which the control messages are sent and received via management ports to avoid mixing with testing traffic. The control messages are mostly sent and received at the setup phase with only a few periodical statistic update messages during the packet generation process with an average traffic below 100 kbps and has little impact on the performance of the system. On top of that, the control message handler is isolated from the worker threads and is pinned to the master core to prevent interference of any kind to the packet generation process.

The control message architecture of our work is shown in Figure 7. Each worker listens to two types of socket, the subscriber socket and the request socket, while the controller listens to the publisher socket and the reply socket. The control messages are classified into the broadcast message and unicast message. The broadcast messages are unidirectional, and are initiated by the controller. The unicast messages, on the other hand, are bidirectional and initiated by the worker.

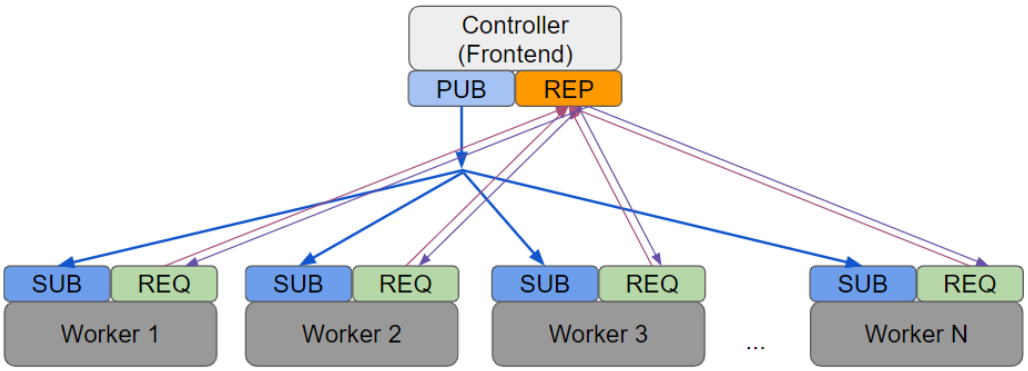


Figure 7: The proposed control message architecture. Each worker listens to two types of socket, the subscriber socket and the request socket, and the controller listens to the publisher socket and the reply socket.

6. Experiments and Testing Results

The experiments conducted to evaluate the performance of the proposed system are grouped into tests of scalability and accuracy. The purpose of a scalability test is to evaluate the system such that the maximum traffic rate can be achieved with multiple worker nodes. In an accuracy test, the similarity of traffic rate distribution between the generated and desired traffic profile is verified in various setups.

Table 4: List of equipment for the experiments conducted in the system performance evaluation.

Category	**Model**	**RAM**	**NIC**	**Amount**
Worker (Ubuntu 16.04.2)	Intel Core i7-2600 @ 3.40Ghz	16G	Intel x520 (Dual-port) DPDK 17.05.1	5
Controller (Ubuntu 16.04.2)	Intel Core i3-2120 @ 3.30Ghz	32G	Endace DAG 9.2x2	1
Aggr. Switch	Quanta LB6M	N/A	24x10Gbps	1

6.1. Experiment Setup

The equipment and the configuration of the software environment used in the experiments are listed in Table 4. Five Intel Core i7-2600 PCs were used as the worker nodes for packet generation task. Each worker was equipped with 16GB of RAM and a dual-port Intel X520 network interface. Note that the hyper-threading feature of the CPU was deliberately disabled as suggested in DPDK documentation. The hyper-threading mechanism contributed additional overhead to the system and decreased the system performance. An Intel Core i3-2120 server equipped with an Endace DAG 9.2X2 dual ports DAG card was used for the controller. The controller was in charge of worker orchestration and packet reception. The Endace DAG 9.2X2 was equipped with a 2GB packet capture buffer. It was also capable of recording packets with hardware timestamping at nanosecond precision and performing line-rate packet capture with zero packet loss.

In the scalability test, the goal was to measure the maximum achievable throughput under different packet sizes. The test was conducted based on the

scenario of single-worker and multiple-worker with hardware rate-limiting on and off. For the scalability test, the network topology was arranged as shown in Figure 5. In this test, multiple CPU cores were used to generate traffic as fast as possible by using a dual-port Intel X520 NIC card. One transmit queue was enabled for each port and is designated to one distinct CPU core.

Table 5: List of enforced traffic profile in the accuracy tests.

Inter-packet delay distribution	**Bitrate**	**Packet size distribution**	**Packet size**	**Test duration**
Poisson	1.31 Gbps	Constant	64 bytes	5 sec
	6.72 Gbps			3 sec
	9.19 Gbps			3 sec

In the accuracy test, the goal was to figure out the ability of the system to replicate the desired traffic profile. The experiments were conducted with single computing core at various combinations of single-worker, multi-worker, a different number of transmit queues, software-based rate limiting, and hardware-based rate limiting. The enforced traffic profiles are listed in Table 5.

6.2. Experiment Results

6.2.1. Scalability Test

In order to test the performance of the traffic generator, a constant packet rate traffic was generated with packet sizes of 64 bytes, 512 bytes, and 1518 bytes. The theoretical rate boundary each packet size is calculated by:

$$Maximum\ Rate = \frac{10\ Gbps}{8 \times (Packet\ size + Frame\ overhead)} \tag{4}$$

with the frame overhead being 20 bytes (12 bytes inter-frame gap and 8 bytes preamble). The calculated rate boundaries of each sizes are 14.88 Mpps, 2.35 Mpps, and 0.81 Mpps. The system was easily able to achieve the aggregated throughput of 20 Gbps with the 99.99% line-rate in each CPU core. The experiment was further extended based on two worker nodes and the system was able to generate 40 Gbps traffic as anticipated. The test results are

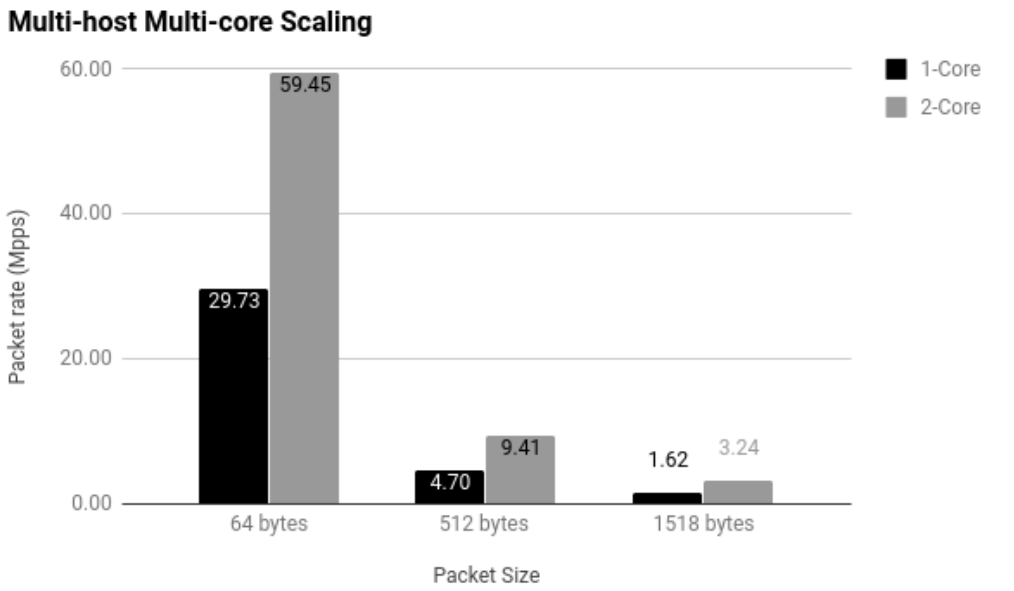


Figure 8: Scalability test results for multi-worker multi-core setups. Two worker nodes are used to generate 40Gbps traffic.

shown in Figure 8. A token-bucket based rate limiter was provided as a fallback position when hardware-based rate limiting was not supported. A Poisson distribution traffic with various workloads from 1.31 Gbps to 9.19 Gbps was enforced. We also evaluated test scenarios with different numbers of transmit queues, and each transmit queue was designated to one distinct CPU core.

6.2.2. Accuracy Test for Single-Worker

We first enforced the 1.31 Gbps Poisson traffic with a single worker, with software-based rate limiting under various numbers of transmit queues. The experiment result is shown in Figure 9. The more number of transmit queues used, the lower the mean squared error reached. The mean squared error (MSE) decreases as the number of transmit queues increases. The MSE of generated traffic with three transmit queues decreases by 72% compared to that of one transmit queue.

The tests were further conducted by enabling the hardware-based rate limiting feature. Compared to that of software-based rate limiting, as shown

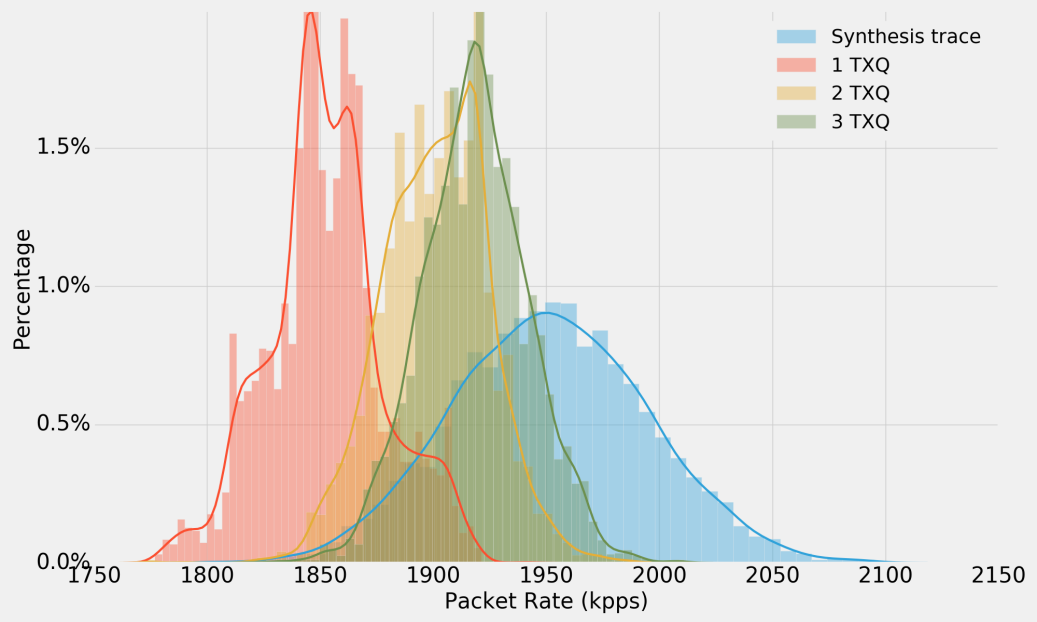


Figure 9: Histogram of packet rate with the feature of *software-based* rate limiting. A Poisson traffic is generated at the rate of 1.31 Gbps in a single worker configuration with different number of transmit queues. Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (1 TXQ) is 12,362. The Pearson correlation coefficients are 0.21, 0.37 and 0.33 for 1 TXQ, 2 TXQ and 3 TXQ.

in Figure 10, the MSE decreased by 74% even with only one transmit queue used. It can be seen on both the histogram and the CDF that the generated traffic came significantly closer to the target traffic profile.

The experiment result of the 6.72 Gbps traffic profile with single-worker configuration is shown in Figure 11. With profile, the MSE was amplified as the volume of traffic increased. We perceived a similar trend to that of 1.31 Gbps. In single-worker configuration, the MSE decreased as the number of transmit queues increased. We further increased the volume of the traffic profile to 9.19 Gbps in order to explore the limits of our system. As shown in Figure 12, it is obvious that we reached the limit, and that the single-worker configuration could no longer keep up with the desired traffic volume. We can see from the CDF of the packet rate that the generated packet rate was capped at around 12.95 Mpps.

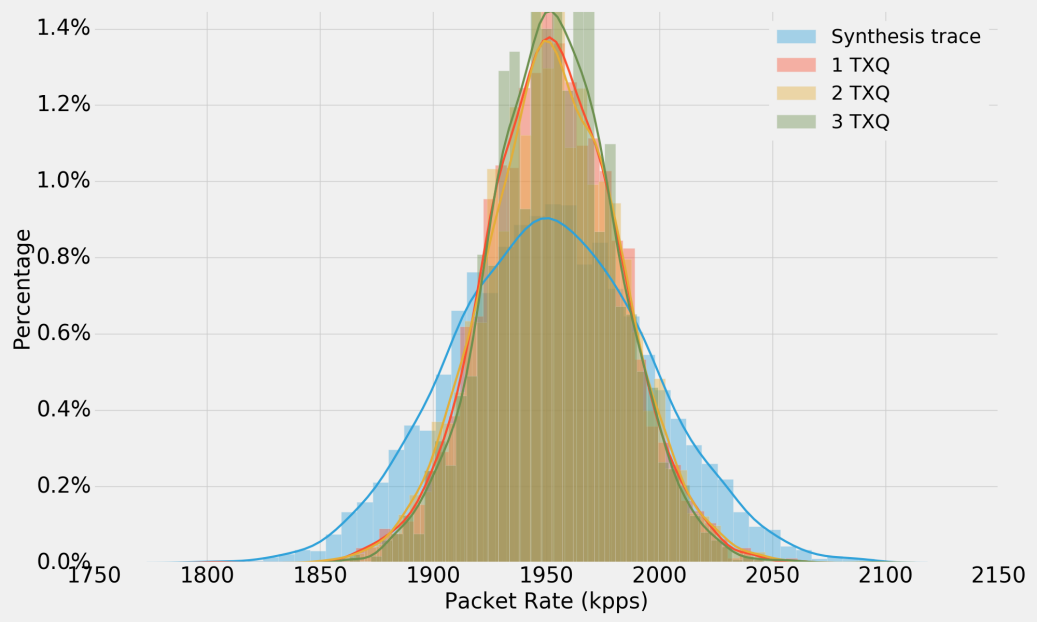


Figure 10: Histogram of packet rate with the feature of *hardware-based* rate limiting. A Poisson traffic is generated at the rate of 1.31 Gbps in a single worker configuration. Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (1 TXQ) is 3,195. The Pearson correlation coefficients are 0.73, 0.82 and 0.61 for 1 TXQ, 2 TXQ and 3 TXQ.

6.2.3. Accuracy Test for Multi-Worker

Thus far, experiments were conducted under a single-worker configuration. We were keen to determine the impact of a multi-worker configuration with different transmit queues. Compared to that of the single-worker configuration with a traffic volume of 1.31Gbps, the average MSE of the multi-worker configuration was increased by 24% and the mean packet rate was dropped by 1%. The main reason was due to a time synchronization error which directly leads to a decrease in the average packet rate. In the multi-worker configuration, time synchronization was critical as all the workers were configured to start packet generation at a scheduled point-in-time.

The experiment result of 6.72 Gbps traffic profile with multi-worker configuration is shown in Figure 13. As shown in the figure, the decrease of MSE is obvious as the number of transmit queues increased. The volume of the traffic profile was further increased to 9.19 Gbps in order to explore the limits of our system. As shown in Figure 12, it was obvious that we

had reached the limit, and that at 9.19 Gbps the single-worker configuration could no longer keep up with the desired traffic volume. The desired traffic profile was offloaded to multiple workers and the result shown in Figure 14. The limitation of traffic generation in a single-worker setup can be overcome with the multi-worker configuration. A scenario of perfect time synchronization by manually realigning the packet streams from each worker is shown in Figure 14. The MSE decreased by 70% compared to that of the original trace.

6.2.4. Reproducibility of the Experiment Results

To show the reproducibility of the experiment results, we replicate CBR and Poisson traffic generation using 5 workers with HRL enabled. The maximum measurable throughput of our capture card is 10 Gbps; thus, we select two traffic volume for each traffic profile: a lower one which is close to 50% of the measurable traffic, and a higher one which is around 90% of the measurable limit. The Poisson traffic generation is generated with a fixed random seed. The NMSE of the result is shown in Table 6, and the estimated intervals are calculated with 95% confidence level.

Table 6: Reproducibility Test Result.

Traffic Type	**Bitrate**	**Replications**	**NMSE**
CBR	5 Gbps	10	$2.70E-05 \pm 2.03E-05$
	9 Gbps	10	$4.70E-05 \pm 2.03E-05$
Poisson	6.72 Gbps	10	$4.91E-05 \pm 2.91E-05$
	9.19 Gbps	10	$7.04E-05 \pm 2.83E-05$

6.3. Discussion

The system utilizes the hardware rate limiting feature of a network interfaces card to provide better accuracy for the software-based rate-limiting mechanism. However, the test results show that enabling hardware-based rate-limiting may affect the throughput of the traffic generated by the system. When conducting a test that requires maximum throughput, optionally disable hardware-based rate-limiting helps the system extract the maximum performance from the hardware. On the other hand, when performing packet

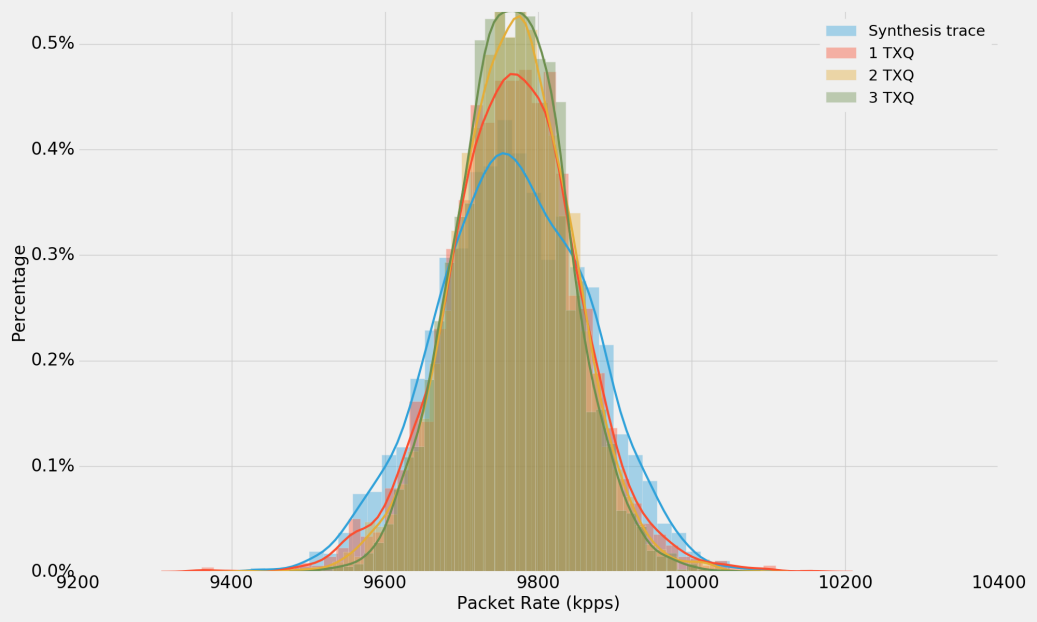


Figure 11: Histogram of packet rate at 6.72 Gbps (Poisson traffic) with single worker of hardware-based rate limiting. Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (1 TXQ) is 8,221. Pearson correlation coefficients are 0.64, 0.72 and 0.57 for 1 TXQ, 2 TXQ and 3 TXQ.

generation of specific traffic distribution with multiple nodes, the accuracy can be increased by enabling hardware-based rate-limiting, as each node only generates part of the total traffic components, The accuracy of our system under a multi-worker configuration highly correlate to the accuracy of time synchronization. With more workers being added to the system, the error of generated traffic distribution increased correspondingly as a result of time synchronization. On top of that, the accuracy of the system could be increased by increasing the number of transmit queues, especially when a hardware rate-limiting feature was enabled.

The results of the experiment provide a guideline for configuring the system. First of all, the hardware-based rate-limiting is preferred over software-based rate limiting for accuracy. While enabling HRL can lower the throughput of the network interface, this can be partly mitigated by increasing the number of configured transmit queues. Secondly, the number of workers required depends on the capacity of the worker. The rule of thumb is to have a total system capacity exceeding the maximum traffic of the traffic profile to

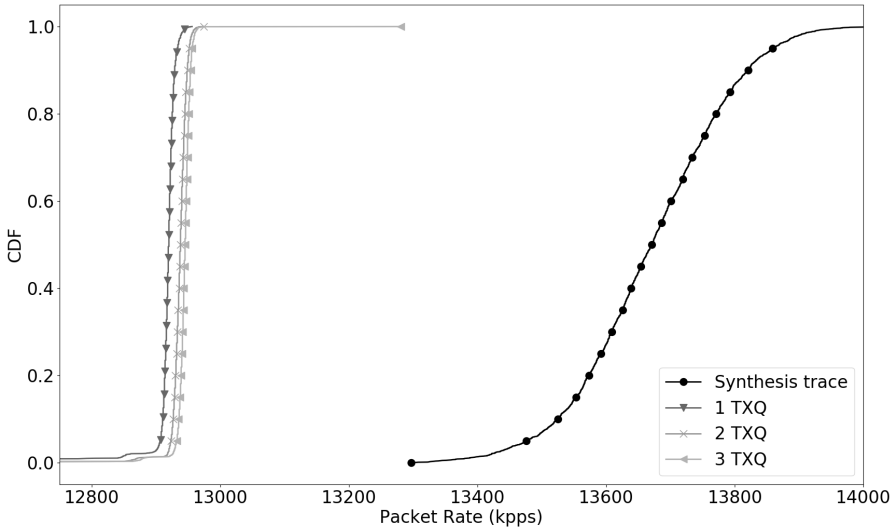


Figure 12: The CDF of packet rate at 9.19 Gbps (Poisson traffic) with single worker (Hardware-based rate limiting with single computing core). Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (1 TXQ) is 921,024. Pearson correlation coefficients of all three traces are below 0.01.

avoid overrunning individual workers. The capacity of the system is calculated as the product of port bandwidth and the number of ports when HRL is disabled. When HRL is enabled, the required capacity is roughly double of the capacity with HRL disabled.

6.3.1. Time Synchronization

In a multi-worker configuration, high-quality time synchronization was one of the critical factors in achieving high accuracy traffic profile generation. Time skew among the workers prevented them from starting packet generation at the exact point-of-time as desired. Also, as shown in our experiment a time skew within 10 milliseconds could result in a 70% MSE increment compared to one with perfect time synchronization. The quality of network time synchronization depended on the software implementation and most importantly, the support of the hardware timestamp. In our work, we adopted the clock-disciplined linuxptp[21] implementation which features a proportional-integral controller servo for frequency adjustment of PTP hardware clock.

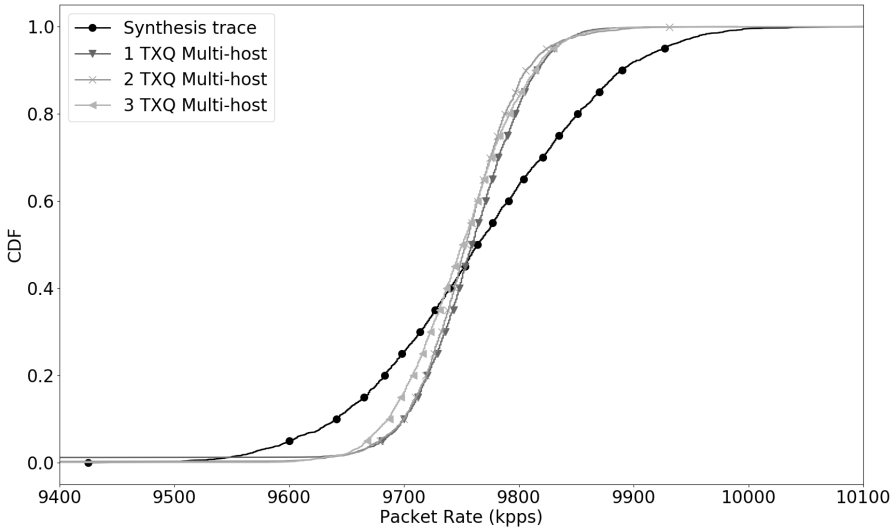


Figure 13: The CDF of packet rate at 6.72 Gbps (Poisson traffic) with 5 workers (Hardware-based rate limiting). Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (1 TXQ) is 38,422. Pearson correlation coefficients are 0.73, 0.58 and 0.78 for 1 TXQ, 2 TXQ and 3 TXQ.

We configured one worker as the grandmaster while other workers operated in slave mode. The workers in slave mode synchronized their PHC to that of the grandmaster. A daemon on each worker then synchronized the hardware clock to the system clock. We noticed a time error deviation of 200 nanoseconds. Errors in the hardware clock to system clock synchronization came to within 100 nanoseconds deviation was also observed for most of the hosts.

6.3.2. Frequency Component Selection

A given traffic distribution consists of various frequency components. For example, a CBR traffic consists of only one dominant frequency component while a given Poisson traffic profile is made up of different frequency components. The more the number of frequency components selected during the packet generation process, the higher the accuracy of the traffic generated. However, with more frequency components selected, the worker node takes more time and computation resources to generate traffic in time. Figure 15 shows the accuracy of the generated traffic and the computation time mea-

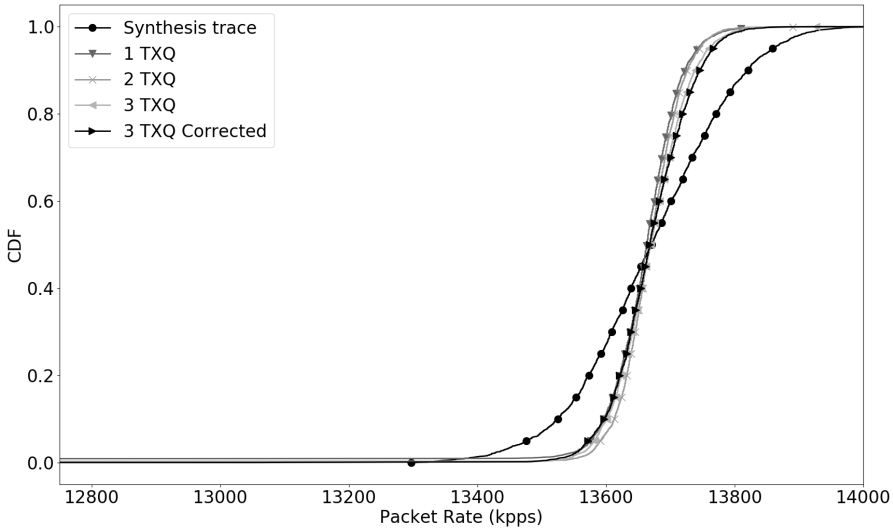


Figure 14: The CDF of packet rate at 9.19 Gbps (Poisson traffic) with 5 workers of re-synchronized trace (Hardware-based rate-limiting). Compared to that of the target traffic profile (synthesis trace), the MSE of the generated traffic distribution (3 TXQ Corrected) is 17,832. Pearson correlation coefficients are 0.81, 0.64 and 0.78 for 1 TXQ. 2 TXQ and 3 TXQ. The Pearson correlation coefficient of the perfect time synchronization (marked by "Corrected") trace is 0.91.

surement with the number of frequency components selected. The NMSE of the generated traffic is inversely proportional to the number of frequency components used during the process. With more frequency components used, the calculation time increases proportionally, indicating that more physical resources or worker nodes is required.

6.4. Summary

For a single host equipped with limited resources (e.g., one computing core), it is difficult to generate high bit rate traffic accurately. By utilizing the mechanisms of hardware rate limiting and multiple transmission queues, MSE can be improved to the high bit rates. The rationale behind the proposed methodology is straightforward. A host of limited resources only needs to take care of generating a selected set of traffic components affordable. Therefore, compared to that of the single-host configuration, a system of

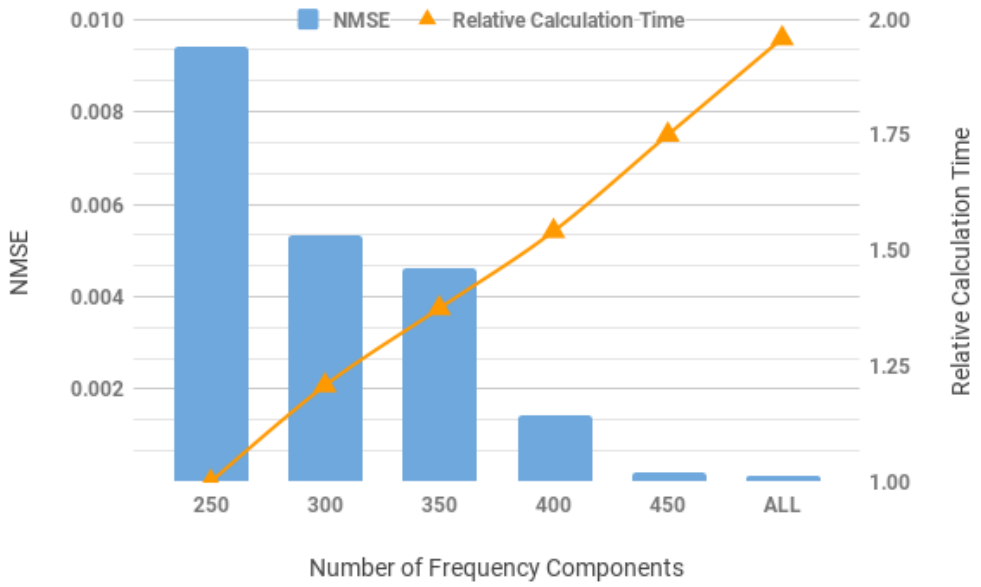


Figure 15: The NMSE and computation time for one worker node generates a specific traffic profile at the rate of 1.31 Gbps with different number of frequency components selected. The number of the frequency component used in the traffic reconstruction process is proportional to the accuracy of the resulting traffic at the cost of higher calculation time.

multiple hosts can generate any given distribution of high-speed traffic profile with lower MSE.

Based on a scenario of perfect time synchronization by manually realignment of packet streams from each worker, the normalized mean square error (NMSE) of single worker setup in each traffic volume is selected as the base and compared with the NMSE of various workers. With an increasing number of workers, we observe a downtrend on the NMSE ratio with various enforced loadings. As shown in Figure 16, the NMSE of a 5-worker configuration at the rate of 1.31 Gbps and 6.72 Gbps is decreased by 16% and 55%, respectively. Shown in Figure 17, the NMSE of the 5-worker configuration at the rate of 9.19 Gbps is decreased by 36%. Notice that, since the generation of traffic at 9.19 Gbps is far beyond the ability of a single worker to utilize a single computing core, the NMSE ratio of a single-worker at 9.19 Gbps is normalized to that of the two-worker configuration.

An accuracy value, obtained from the average NMSE of 1.31 Gbps traf-

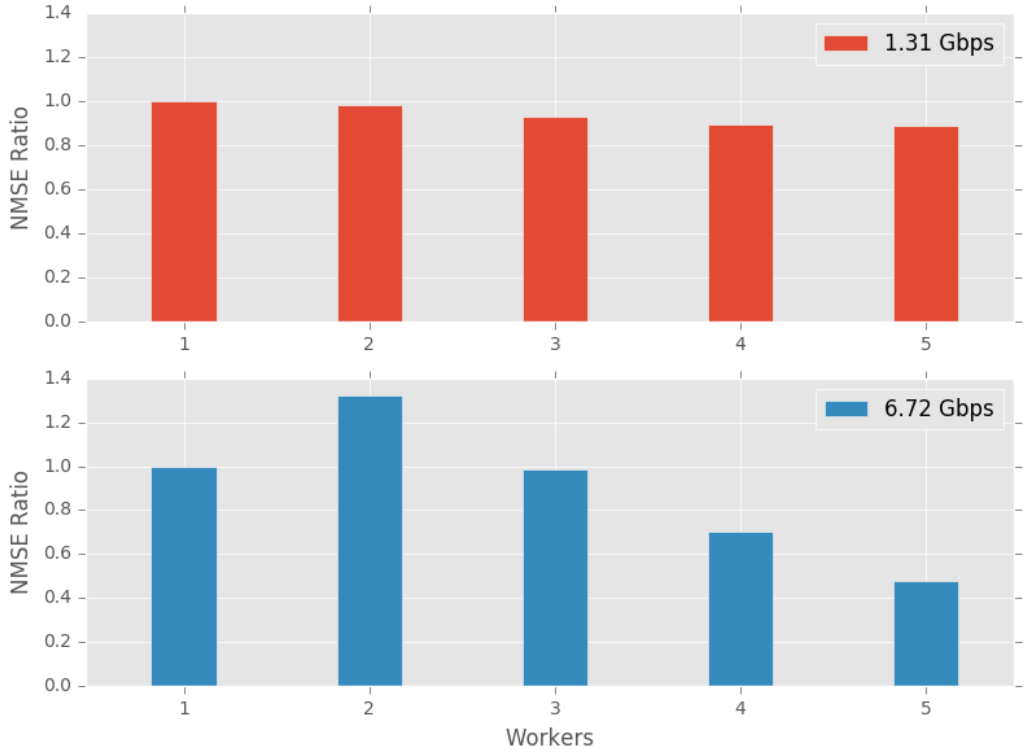


Figure 16: Comparisons of NMSE with various number of workers with target traffic rate of 1.31 Gbps and 6.72 Gbps, respectively. The more the worker presented in the system the lower the NMSE achieved. The NMSE is normalized to that of the single-worker setup.

fic with hardware-rate limiting, is selected as the baseline accuracy threshold. Based on the tested results of all traffic profiles and configurations, the maximum throughput with an NMSE value lower than the selected baseline accuracy threshold is summarized in Table 7. The maximum aggregated throughput is based on the statistics counter of the Ethernet switch in multi-worker configuration.

As the traffic volume increases, the error rate of the single-worker mode grows. However, with multi-worker configurations, the error rate does not increase significantly with an increased traffic rate. In order to explore the performance under multi-worker configurations, tests are further conducted based on the aggregated traffic beyond 10 Gbps. An OpenFlow Switch (Edgecore AS5712-54X) was used to steer traffic generated from multiple worker nodes into aggregated traffic at 40 Gbps. As the resulting 40 Gbps

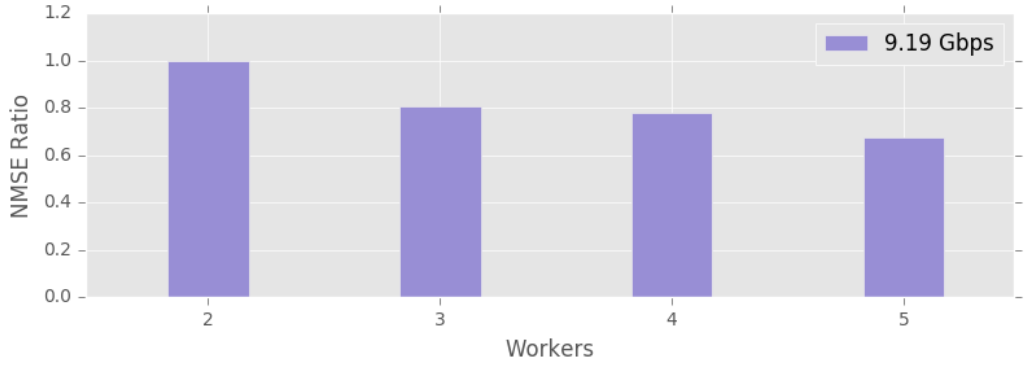


Figure 17: Comparison of NMSE with various number of workers with target traffic rate of 9.19 Gbps. The more the worker presented in the system the lower the NMSE achieved. The NMSE is normalized to that of the two-worker setup for better visual perception.

traffic will overrun our 10 Gbps packet capture card, we replaced the capture card with a Mellanox ConnectX-3 40 Gbps network adapter at the receiving end and configured it with three receiving queues to maximize receiving capability. The packet loss rate with the maximum-sized packet of 1500 bytes at 30 Gbps is 12.24%. A higher packet loss rate of 89.62% with the 64-byte minimum-sized packets at 40 Gbps was also observed. Therefore, as a result of the performance bottleneck of a network adapter at the receiving end, it is difficult to verify the maximum scalability and accuracy at a higher traffic rate.

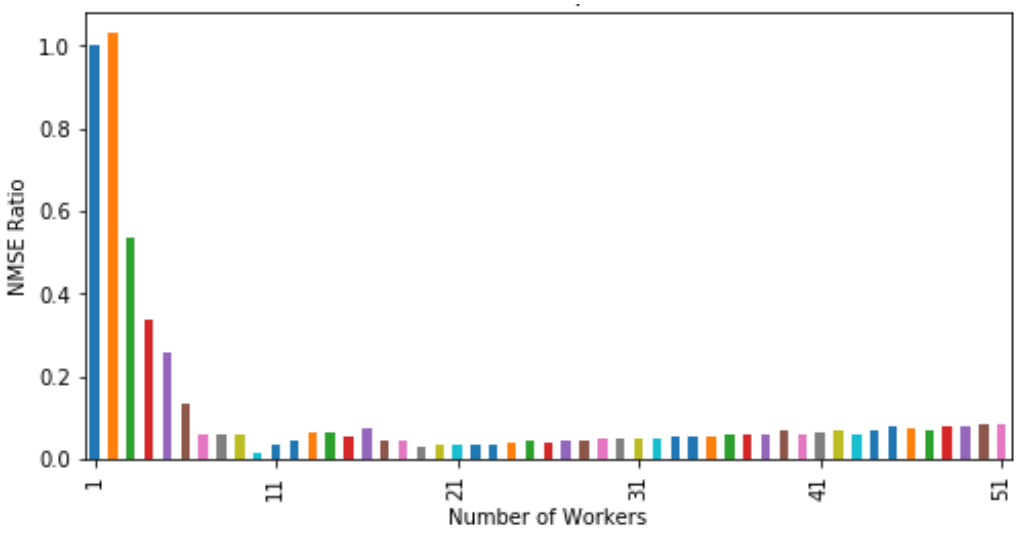
To further evaluate the accuracy and scalability limit of the system, we simulated the behavior of multi-worker packet generation process with the random time synchronization error and worker capability taken into account. The simulation results are presented in Figure 18. For the accuracy limit evaluation, a Poisson traffic profile with an average loading of 6.72 Gbps is enforced, and a various number of workers ranging from 1 to 51 are used in each iteration. As shown in Figure 18a, a drastic downtrend of NMSE is observed as the number of workers grows. The trend discontinued at ten workers, where minimum NMSE (maximum accuracy) is achieved and began to climb up gently afterward. Similarly, to verify the scalability of the system, we increased the average loading to 90 Gbps. As shown in Figure 18b, the NMSE continues to drop as more workers engage in the generation process. On top of that, with more workers available, we decreased the t_{sample} to

one-tenth of the default value under high workload. The higher sampling frequency contributed to an average of 30% decrease on the NMSE. Notice that the results with less than ten workers are omitted as the scale of traffic overwhelmed the capacity of the system and led to a significantly higher error rate.

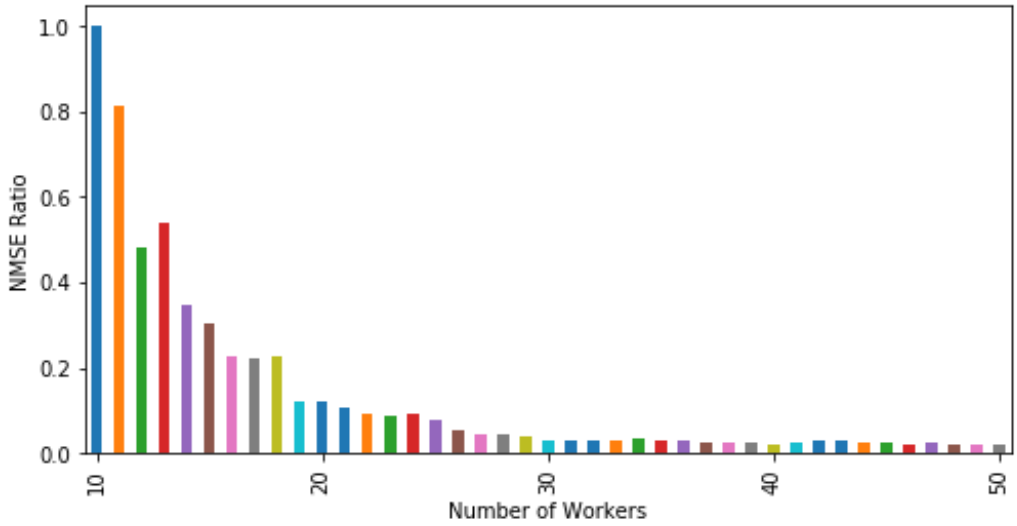
The experiment results and simulation results show that the proposed system can effectively distribute the traffic workload to the workers to generate more accurate traffic profile. Most importantly, the system can scale up to a heavy workload with larger clusters of worker nodes and availability of the resource.

Table 7: The maximum throughput with the NMSE value lower than the baseline accuracy threshold (with hardware-rate limiting at 1.31 Gbps traffic). Due to the limited capability of the NIC with high packet loss rate at the receiving side, the maximum measurable throughput is bounded and the NMSE of multi-worker configuration is not available*.

Configurations		**Maximum Throughput**	**Maximum Measurable Throughput**	**NMSE**
Single-worker	Single Queue	0.90 Gbps	0.90 Gbps	8.31E-04
	Multi-queue	6.72 Gbps	6.72 Gbps	9.28E-04
Single-worker (HRL)	Single Queue	6.80 Gbps	6.80 Gbps	9.72E-04
	Multi-queue	9.19 Gbps	9.19 Gbps	7.78E-04
3 Workers		27.51 Gbps (aggregated)	9.19 Gbps (receiving end)	N/A*
5 Workers		49.95 Gbps (aggregated)	9.19 Gbps (receiving end)	N/A*



(a) Simulation with 6.72 Gbps Poisson Traffic



(b) Simulation with 90 Gbps Poisson Traffic

Figure 18: Comparison of NMSE with simulated workers. Fig. 18a shows the accuracy limit of the system as the downtrend of NMSE discontinued at ten workers and started to climb up slowly. The NMSE in the figures is normalized to a single worker and ten workers respectively.

7. Conclusions and Future Work

In this work, we propose a software-based packet generator that is capable of supporting distributed packet generation of user-defined traffic profile on commercial-off-the-shelf (COTS) technology network interface cards (NICs) with high accuracy and scalability. The system is developed on a novel Fourier-based profile decomposition and formulation methodology. This is an important feature that is missing in the ecosystem, because it creates the possibility to generate traffic distribution with multiple-gigabit accurately. By increasing the number of worker nodes, the accuracy and throughput of the system can thus be maintained. In this study we have tested traffic generation of 40 Gbps with minimum-sized Ethernet frame on multi-host setup is conducted. However, because of the limitation of the packet capturing capacity of the NIC, an analysis of accuracy is not available at this time.

Time synchronization among the workers and the controller constitutes a crucial factor for the accuracy of the distributed system. The experiment results show that a system with time offset of several milliseconds among the workers can result in 70% increment in error rate when compared to one that with perfect time synchronization. To mitigate the time offset of the system, clock-disciplined precision time protocol with NIC card featuring hardware-timestamping is adopted in the proposed system.

In addition, the experiment results show that different NIC hardware architecture does have an impact on the accuracy of the system. For instance, an Intel XL710 NIC tends to pack packets into bursts, which makes it hard to insert the desired gap between packets. The Intel 82599, on the other hand, does not manifest such behavior. We, therefore, plan to further enhance the performance of a system by exploring the hardware architecture of commodity network interface cards. Furthermore, instead of using an STFT-based profile decomposition of the fixed-time window, the system can be further extended by using the method of wavelet transform-based profile decomposition. We look forward to achieving the goal of packet generation and accuracy analysis at higher throughput using multi-worker configurations with adaptive time and frequency resolution in the near future.

8. Acknowledgment

This research was funded in part by the Ministry of Science and Technology (MoST), Taiwan, and in part by Estinet Technologies Inc. and Chunghua Telecom.

9. References

[1] Developer Guide | Protocol Buffers. URL: https://developers.google.com/protocol-buffers/docs/overview; (Accessed on 2017-12-24).

[2] Getting started with pktgen, pktgen 2.7.7 documentation. URL: http://pktgen.readthedocs.io/en/latest/getting_started.html.

[3] IEEE 802.3 50 Gb/s, 100 Gb/s, and 200 Gb/s Ethernet Task Force. http://www.ieee802.org/3/cd/; . (Accessed on 2017-11-30).

[4] IEEE P802.3bs 400 Gb/s Ethernet Task Force. . URL: http://www.ieee802.org/3/bs/; (Accessed on 2017-11-30).

[5] Ixia Makes Networks Stronger. URL: https://www.ixiacom.com/; (Accessed on 2018-01-03).

[6] LuaJIT. URL: http://luajit.org/luajit.html; (Accessed on 2017-12-24).

[7] Network, Devices & Services Testing - Spirent. URL: https://www.spirent.com/; (Accessed on 2018-01-03).

[8] PF_ring ZC (zero copy). URL: http://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/; (Accessed on 2017-05-04).

[9] Programmer's Guide Data Plane Development Kit 18.02.0-rc0 documentation. URL: http://dpdk.org/doc/guides/prog_guide/; (Accessed on 2017-12-24).

[10] Xena Networks - High-performance Gigabit Ethernet test solutions. URL: https://xenanetworks.com/; (Accessed on 2018-01-03).

[11] Intel 82574 gigabit ethernet controller family datasheet. https://www.intel.com.tw/content/www/tw/zh/embedded/products/networking/82574l-gbe-controller-datasheet.html; 2015. (Accessed on 2017-09-20).

[12] Intel 82580 EB/DB GbE Controller Datasheet. https://www.intel.com/content/www/us/en/embedded/products/networking/82580-eb-db-gbe-controller-datasheet.html; 2015. (Accessed on 2017-09-20).

[13] Angrisani, L., Botta, A., Miele, G., Pescape, A., Vadursi, M.. Experiment-driven modeling of open-source internet traffic generators. IEEE Transactions on Instrumentation and Measurement 2014;63(11):2529–2538. doi:10.1109/TIM.2014.2348633.

[14] Angrisani, L., Narduzzi, C.. Testing communication and computer networks: an overview. IEEE Instrumentation Measurement Magazine 2008;11(5):12–24. doi:10.1109/MIM.2008.4630738.

[15] Antichi, G., Shahbaz, M., Geng, Y., Zilberman, N., Covington, A., Bruyere, M., Mckeown, N., Feamster, N., Felderman, B., Blott, M., Moore, A.W., Owezarski, P.. Osnt: open source network tester. IEEE Network 2014;28(5):6–12. doi:10.1109/MNET.2014.6915433.

[16] Avallone, S., Guadagno, S., Emma, D., Pescape, A., Ventre, G.. D-ITG distributed internet traffic generator. In: First International Conference on the Quantitative Evaluation of Systems, 2004. QEST 2004. Proceedings. 2004. p. 316–317. doi:10.1109/QEST.2004.1348045.

[17] Barford, L., Burch, J.. Fourier analysis from networked measurements using time synchronization. IEEE Transactions on Instrumentation and Measurement 2007;56(5):1601–1604. doi:10.1109/TIM.2007.903599.

[18] Bonelli, N., Giordano, S., Procissi, G.. Enabling packet fan-out in the libpcap library for parallel traffic processing. In: 2017 Network Traffic Measurement and Analysis Conference (TMA). 2017. p. 1–9. doi:10.23919/TMA.2017.8002904.

[19] Botta, A., Dainotti, A., Pescape, A.. Do you trust your software-based traffic generator? IEEE Communications Magazine 2010;48(9):158–165. doi:10.1109/MCOM.2010.5560600.

[20] Bradner, S., McQuaid, J.. RFC 2544; 1999. URL: https://www.ietf.org/rfc/rfc2544.txt.

[21] Cochran, R., et al. The linux ptp project. http://linuxptp.sourceforge.net/; 2015. (Accessed on 2019-04-05).

[22] Emmerich, P., Gallenmller, S., Raumer, D., Wohlfart, F., Carle, G.. MoonGen: a scriptable high-speed packet generator. In: Proceedings of the 2015 ACM Conference on Internet Measurement Conference. ACM; 2015. p. 275–287.

[23] Fusco, F., Deri, L.. High speed network traffic analysis with commodity multi-core systems. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement. New York, NY, USA: ACM; IMC '10; 2010. p. 218–224. doi:10.1145/1879141.1879169.

[24] Hintjens, P.. ZeroMQ: The Guide, URL: http://zguide.zeromq.org/page:all.

[25] Jiang, D., Xu, Z., Zhang, P., Zhu, T.. A transform domain-based anomaly detection approach to network-wide traffic. Journal of Network and Computer Applications 2014;40:292 – 306. doi:10.1016/j.jnca.2013.09.014.

[26] Lockwood, J.W., McKeown, N., Watson, G., Gibb, G., Hartke, P., Naous, J., Raghuraman, R., Luo, J.. Netfpga–an open platform for gigabit-rate network switching and routing. In: 2007 IEEE International Conference on Microelectronic Systems Education (MSE'07). IEEE; 2007. p. 160–161.

[27] Nawab, S.H., Quatieri, T.F.. Short-time fourier transform. In: Advanced topics in signal processing. Prentice-Hall, Inc.; 1987. p. 289–337.

[28] Olsson, R.. Pktgen the linux packet generator. In: Proceedings of the Linux Symposium, Ottawa, Canada. volume 2; 2005. p. 11–24.

[29] Patil, A.G., Surve, A.R., Gupta, A.K., Sharma, A., Anmulwar, S.. Survey of synthetic traffic generators. In: 2016 International Conference on Inventive Computation Technologies (ICICT). volume 1; 2016. p. 1–3. doi:10.1109/INVENTIVE.2016.7823282.

[30] Postel, J., Anderson, C.. White Pages Meeting Report. RFC 1588; 1994. URL: https://www.ietf.org/rfc/rfc1588.txt.

[31] Rizzo, L.. Netmap: a novel framework for fast packet i/o. In: 21st USENIX Security Symposium (USENIX Security 12). 2012. p. 101–112.

[32] Rizzo, L., Deri, L., Cardigliano, A.. 10 gbit/s line rate packet processing using commodity hardware: Survey and new proposals. 2012.

[33] Salah, K., Hamawi, M.. Performance of ip-forwarding of linux hosts with multiple network interfaces. Journal of Network and Computer Applications 2013;36(1):452 – 465. doi:10.1016/j.jnca.2012.04.013.

[34] Schneider, F., Wallerich, J., Feldmann, A.. Packet capture in 10-gigabit ethernet environments using contemporary commodity hardware. In: Proceedings of the 8th International Conference on Passive and Active Network Measurement. Berlin, Heidelberg: Springer-Verlag; PAM'07; 2007. p. 207–217.

[35] Wu, W., DeMar, P.. Wirecap: A novel packet capture engine for commodity nics in high-speed networks. In: Proceedings of the 2014 Conference on Internet Measurement Conference. New York, NY, USA: ACM; IMC '14; 2014. p. 395–406. doi:10.1145/2663716.2663736.

[36] Zhang, J., Tang, J., Zhang, X., Ouyang, W., Wang, D.. A survey of network traffic generation. In: Third International Conference on Cyberspace Technology (CCT 2015). 2015. p. 1–6. doi:10.1049/cp.2015.0862.

Ching-hao Chang received B.S. degree in electrical engineering from National Chung Cheng University, Chiayi, Taiwan, in 2015 and the M.S. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2017. He is currently a research assistant at Network Benchmarking Lab, Taiwan. His research interest includes design and implementation of high-speed networking, information security and blockchain technology.

Ying-Dar Lin received the Ph.D. degree in computer science from the University of California at Los Angeles (UCLA), Los Angeles, CA, USA, in 1993. He is a Distinguished Professor of computer science at National Chiao Tung University (NCTU), Taiwan. He was a visiting scholar at Cisco Systems in San Jose, California, during 2007–2008. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic and has been an approved test lab of the Open Networking Foundation (ONF) since July 2014. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. His research interests include network security and wireless communications. His work on multihop cellular was the first along this line, and has been cited over 750 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced. He is also an IEEE Fellow (class of 2013), IEEE Distinguished Lecturer (2014–2017), and an ONF Research Associate. He currently serves on the editorial boards of several IEEE journals and magazines. He published a textbook, Computer Networks: An Open Source Approach (www.mhhe.com/lin), with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011).

Yu-Kuen Lai received the M.S. and Ph.D. degrees in electrical and computer engineering from North Carolina State University (NCSU) at Raleigh, NC, USA in 1997 and 2006, respectively. He is an Associate Professor with the Electrical Engineering Department, Chung-Yuan Christian University (CYCU), Chung-Li, Taiwan. From 1997 to 2002, he was a Senior ASIC Design Engineer with Delta Networks, Inc., and Applied Micro Circuit Corporation (AMCC) at Research Triangle Park, NC, USA. His research interests include network processor architecture, streaming data processing, network traffic analysis, FPGA systems design, and computer network security. Dr. Lai served as the Electronic

Communication Officer for the IEEE Taipei Section in 2015. He was the recipient of the CYCU Distinguished Teaching Award and CYCU Outstanding Teaching Award in 2011 and 2017, respectively.

Yuan-Cheng Lai received the Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1997. In August 1998, he joined the faculty of the Department of Computer Science and Information Science, National Cheng Kung University, Tainan, Taiwan. In August 2001, he joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, where he has been a Professor since February 2008. His research interests include performance analysis, protocol design, wireless networks, and web-based applications.