Poisson Shot-Noise Process Based Flow-Level Traffic Matrix Generation for Data Center Networks

Yoonseon Han*, Jae-Hyoung Yoo†, James Won-Ki Hong*†

*Division of IT Convergence Engineering, POSTECH
{seon054, jwkhong}@postech.ac.kr

†Department of Computer Science and Engineering, POSTECH
{styoo}@postech.ac.kr

***Abstract*—The number of data centers has been increased for various reasons such as cloud computing, big-data analysis, multimedia service, etc. With public interests on data center, many novel technologies for data center networks have been proposed and deployed to support data center operations more efficiently and effectively. However, the construction of data center network incurs significant costs. Moreover, various technologies interplay each other to achieve multiple objectives, and it makes difficult to validate and/or verify characteristics of data center network. In addition, it difficult to perform experiments with a number of hosts and switches. Therefore, it is necessary to observe the characteristics of target data center network before building it. A common approach to evaluate data center is to run simulations that should be similar with real-world data center environment. However, generating traffic with the characteristics of data center networks is not matured yet. People still employ a traffic generator based on the characteristics of Internet traffic. We design a traffic generator that shows more accurate characteristics of data center network traffic. Various traffic characteristics exploited explored by several studies are considered. The proposed method generates flow-level network traffic matrix based on Poisson Shot-Noise model. We implemented the traffic generator using Python programming language to create traffic matrix. To evaluate the proposed method, we compare the results with real data center network traffic. Our results show that the generated traffic owns similar characteristics with the real network traffic in terms of flow size, duration, and the mean and variance of total traffic rate.**

Keywords*—*Traffic Generation Method, Data Center Networks, Flow-Level Traffic Matrix, Poisson Shot-Noise Model

I. Introduction

Data center refers a facility used to house computer servers and hosts. Modern data centers might contain tens of thousands of hosts to support the needs of cloud computing, multimedia contents, and big data analysis. Data center brings benefits for customers in terms of CAPAX/OPEX reduction of the customers' services. Many data center network topologies and architectures are proposed to address the diverse requirements such as cost reduction, energy saving, scalability, overcome of bandwidth/capacity limitations and etc. In the current data center network, various technologies interplay to achieve multiple objectives, and they make the structure of data center network more larger and complex. In this situation, performing experiments, to measure performance or to confirm proper operations, on real-world data center becomes more difficult. The construction of tested incurs significant costs in terms of time and money. Therefore we are required to perform simulations to measure networking in data center instead of running actual tasks on real testbed or operating data center.

The challenge is to reflect the characteristics of real-world data center traffic for the simulations. For accurate test results, it is important to construct the simulation environment as same as the target data center network in terms of topology and link capacity. However, it is impossible to emulate all behaviors of running applications and network traffic generated by them. In this situation, the popular way for testing experiments is to use artificially generated network traffic based on mathematical models. However, most of previous studies on the traffic generation have focused on generating network traffic with the characteristics of Internet traffic, not data center network traffic. The characteristics of data center network are different from those of internet traffic because the traffic occurs inside the data center among a number of hosts. For example, distributed computing tasks show two traffic characteristics, i.e. distributing a task into smaller ones and aggregating the results. Therefore, we need a new mathematical traffic model and generation method that reflect the characteristics of data center networks.

In this paper, we propose a network traffic generation method with the consideration of various characteristics of data center network traffic using mathematical tools. We utilized the characteristics of data center networks explored by several studies [1]–[4]. These studies showed flow level traffic characteristics based on working data center network traffic analysis. To model the characteristics, we introduced several mathematical models such as Poisson shot-noise process, Poison arrival process, Pareto, and Weibull distribution. Then, we express flow-level traffic characteristics such as flow arrival, size, and duration. Moreover, with a powerful notation "Shot", a mathematical function for describing total traffic transmission rate. We can address the behavior of network traffic on a specific link in terms of the mean transmission rate and variation (burstiness of the traffic).

Our method uses a traffic matrix that contains traffic volume information among hosts of the given data center networks at a certain duration of time unit. After generating flow-level traffic summary, which is similar with origin-destination (OD) flow matrix, we generate network packets using an existing packet-level traffic generator such as iPerf [16] by creating control commands according to the generated traffic matrix.

The proposed method consists of six steps: (1) Recognizes network topology, (2) Decides new flow arrivals, (3) Generates flows with 5-tuples, (4) Assigns flow duration and size, (5) Injects flow transmission rate, and (6) Generates traffic matrix. We implemented the proposed method using Python programming language. To evaluate the proposed method, we compared the traffic generation results with the characteristics of real-world data center network traffic.

The paper is organized as follows. Section II introduces diverse data center networks topologies, studies exploring the characteristics of data center network traffic, and existing technologies to generate network traffics. Section V presents our traffic generation method to address data center network traffic characteristics. Thereafter, Section VI describes how the proposed method is implemented, and then shows the evaluation results. Finally, Section VII concludes this paper with our future work.

II. Related work

A data center is a facility used to house computer servers or hosts along with a data center network that interconnects them using dedicated links and switches. Modern data centers might contain tens of thousands of hosts with significant bandwidth requirements [5]. A multi-rooted hierarchical tree topology which consists of three layers (edge, aggregate, and core) has been widely adopted for many data center networks. Unfortunately, the conventional topology has several limitations for building and managing large-scale data centers such as limited capacity, high oversubscription ratio, high capital expenditures (CAPEX), and operating expenditures (OPEX). To overcome these limitations, various DCN topologies have been suggested. Fat-Tree [5], VL2 [6], and PortLand [7] are based on a Clos network topology, which provides extensive path diversity using smaller commodity switches. DCell [8] and BCube [9] take a server centric approach. These new approaches bring breakthroughs to solve the problems of the conventional topology. Usually, in data centers, the hosts running a set of related or grouped applications are arranged at the same rack or near location to reduce communication cost. Therefore, this is obvious that the traffic generation method must consider the network topology to generate network traffic for data center correctly.

Mathematical tools to express the properties of network network traffic has beed studies during long time. The most of approaches analyzed Internet backbone traffic, and proposed various mathematical models to describe the properties in the form of probability and/or stochastic process theory. Thanks to the studies, we know that Internet traffic has long range dependency and self-similarity [10], [11]. Many studies analyze and model Internet traffic at the flow level. The number of active flows can be estimated using M/G/∞ queuing model in a non-congested link [12]. In [13], [14] , a method to infer the total flow transmission rate is proposed from the moments of traffic transmission rate function using Poisson Shot-Noise process. Our proposed traffic generation method also applied Poisson Shot-Noise Process to model the average transmission rate and variance (burtiness).

Network traffic characteristics of data center networks are slightly known through a few studies despite of the significant interest. The studies performed by Benson et al. [1], [2] analyzed traffic data in the form of both simple network management protocol (SNMP) logs and packet traces. The traffic data was obtained from several DCNs such as universities, private enterprises, and commercial clouds. Kandula et al. [3] collected network related events from 1,500 servers in a DCN for over two months by using a lightweight monitoring program installed in the servers, rather than switches. Heller et al. [4] found that the total traffic volume in a data center network over time varies daily, weekly, monthly, and yearly. The characteristics found by these studies are described in Section III.

Many traffic generation methods and tools have been proposed, and some of them are available on the market. [15] summarized network traffic generators that are widely used in networking researches by classifying them into several categories. Maximum throughput generators, such as iPerf [16], are frequently used to test end-to-end network performances. Model-based traffic generators utilize different stochastic models for creating packet-level traces to reflect several statistical characteristics [17]. High-level and auto-configurable generators are based on a higher-level model of network traffic to create statistically similar traffic with live measurements. HARPOON is a traffic generator that produces network traffic with various flow-level characteristics [18]. SWING is another high-level traffic generator that can generate traffic based on the characteristics of real traces [19]. However, these traffic generators have limitations in that they are focused on the creation of network traffic similar to that measured on the Internet. They do not address the characteristics of data center network traffic.

III. The Characteristics of Data Center Network Traffic

In this section, we introduce the characteristics of data center network traffic observed by several studies [1]–[4]. We consider macro and flow level traffic characteristics such as network usage change, intra-rack traffic ratio, flow size and duration.

Characteristic 1 : The total network traffic volume varies depending on the time.

The total traffic volume in a data center network varies daily, weekly, monthly, and yearly [4]. Typically, data center networks show different usage ratios by the time. For example, data center networks volume shows low usage ratio during the mid-night than the daytime, and weekends might have lower usage ratio than weekdays. These characteristics are easily measured using SNMP. Fig. 1 shows the trend of ingress and egress traffic according to the time. This traffic traces were collected using SNMP from a commercial data center network located in Korea for 6 days, and were aggregated according to the reported time.

Characteristic 2 : Most of data center traffic stays in the same rack.

From the studies of Benson et al. [1], [2] and Kandula el al. [3], most of traffic flows have a pair of hosts (source and destination) located in the same rack. Both studies show that a majority of network traffic around 50 to 80% stay in the

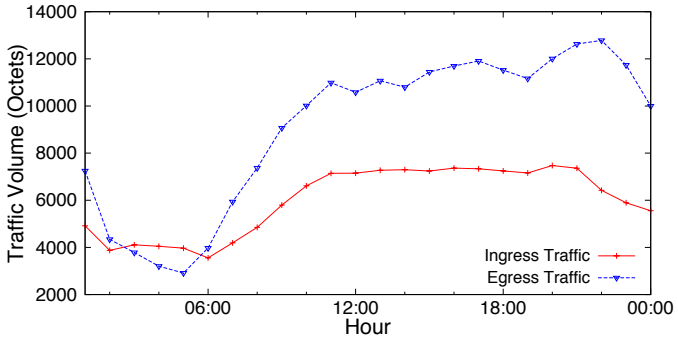

Fig. 1. The trend of ingress and egress data on a data center

same rack. Depending on the type of data center, the ratio of intra versus extra rack traffic could be changed, but this claim hold for most data centers for both enterprise and commercial purposes. The reason of high intra rack traffic ratio is that the related or grouped applications are arranged among the nearly located hosts to reduce communication cost.

Characteristic 3 : Most of data center traffic is occupied by small portion of flows.

Benson et al. [1], [2] and Kandula et al. [3] described the characteristics of flow duration and size for data center network traffic. The terminology, "Flow", is usually defined as a group of packets having same five-tuple, (source IP address, destination IP address, source port number, destination port number, and protocol). This paper follows the same definition. Benson et al. [1], [2] and Kandula et al. concluded that 80% of flows last less than 11 seconds in duration, and 80% of flows are smaller than 10 Kb in size. Moreover, most of the traffic volume is occupied by the top 10% of traffic flows. Fig. 2 and 3 show the distribution of flow duration and size from the data collected from the university data center that is the same data set with analyzed in [1]. Note that Fig. 3 is represented in log scale.

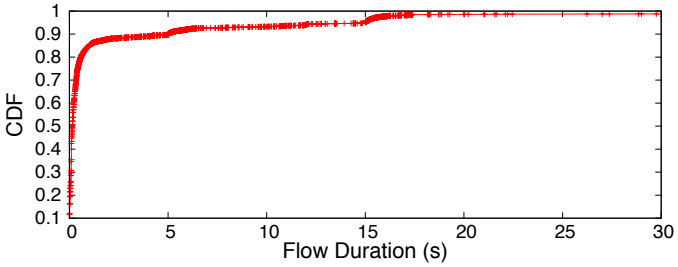

Fig. 2. The distribution of flow duration

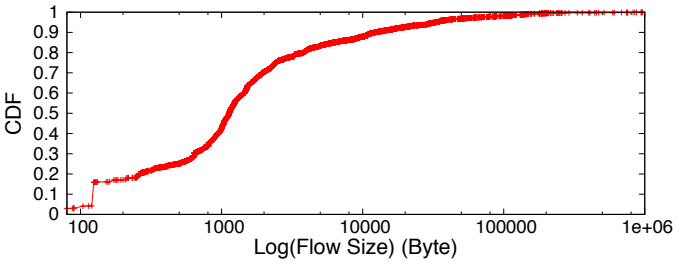

Fig. 3. The distribution of flow size

Unfortunately, there are no accurate mathematical model to represent the relationship between flow size and duration. We know that flow size and duration are in a proportional relationship, but not in a linear relationship. Fig. 4 shows the cumulative distribution of flow size over flow duration.

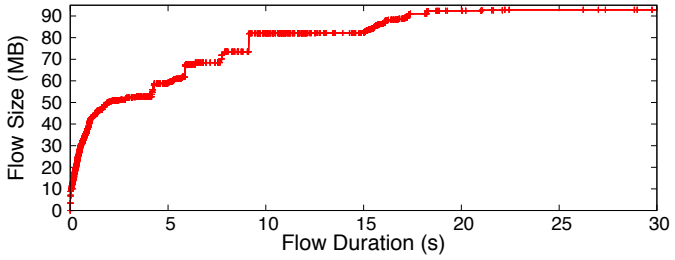

Fig. 4. The cumulative distribution of flow size over duration.

IV. The Network Traffic Model

In this section, we describe the Poisson Shot-Noise Process. In [13], [14], this model is proposed to describe the behavior of data flows arriving at Internet backbone. Although the model is proposed for Internet backbone traffic, the model is still valid if the two assumptions hold. The notations used for Poisson Shot-Noise Process is summarized on TABLE I.

TABLE I. Notations used in Poisson Shot-Noise Process Model.

Symbol	Description
T_n	The arrival time of nth flow. ($n \in \mathbb{Z}$)
S_n	The size of nth flow. (Byte)
D_n	The duration of nth flow. (s)
$X_n(t - T_n)$	The transmission rate of nth flow at the given time t. (Byte)

Assumption 1 : Flow arrivals follow a homogeneous Poisson process of finite rate λ [13], [14].

The flow arrival rate of data center network changes according to the time. For instance, there will be higher flow arrival rate at the peak in the daytime when the data center usage rate is high (see Fig.1). To address this change, we need to model flow arrival rate with more general processes such as non-homogeneous Poisson process or Markov arrival process. However, the change of flow arrival rate is quite slow compared to flow generation interval (usually 1s). In a short time range, the change rate is small enough to ignore. Therefore, we can assume the flow arrivals follow a homogeneous Poisson process with a fixed arrival rate.

If we assume a homogeneous Poisson process for new flow arrivals, the number of new flow arrivals during $T_{start} - T_{end}$, N, can be express as (1). Where T_{start} is the start time, T_{end} is end time of flow generation, and $\lambda(t)$ is a function to return average flow arrival rate at the given time t.

$$\mathrm{N} = \int_{T_{start}}^{T_{end}} \lambda(t)\,\mathrm{d}x \tag{1}$$

Assumption 2 : Flow rate functions are independent of each other and identically distributed [13], [14].

A flow rate function, also called as "shot", returns transmission rate of a flow at the given time. The flow rate function

of n th flow is expressed as $X(t - T_n)$. The assumption on the independence of shots are based on that a data center link is shared by various and numerous flows and under utilized [13], [14]. The assumption on the identical distribution of shots is to keep the simplicity of the proposed method. The later assumption could be relaxed by introducing multiple classes of shots. A shot depend on flow size S_n, flow duration D_n, and its shape. The relationship is presented in (2). Using the concept of a shot, we can express the total rate of the traffic as (3). This model is a Poisson shot noise process. By applying the Laplace Stieltjes Transform (LST) to $R(t)$, we can compute the all moments of $R(t)$ such as its first order moment (mean) and second order moment (variance) [13], [14].

$$\int_{T_n}^{t} X(t - T_n) = S_n \tag{2}$$

$$R(t) = \sum_{n \in \mathbb{Z}} X(t - T_n) \tag{3}$$

The shape of a shot is closely related with the characteristics of the traffic such as average total rate and its variance. For the simplicity, we assumed a simple shot function that can be represented in a power function as (4). A power function is simple, but powerful model to represent the shape of shots. Fig.5 shows various shapes of shots by adjusting the parameters (a, b).

$$X(t - T_n) = a(t - T_n)^b \tag{4}$$

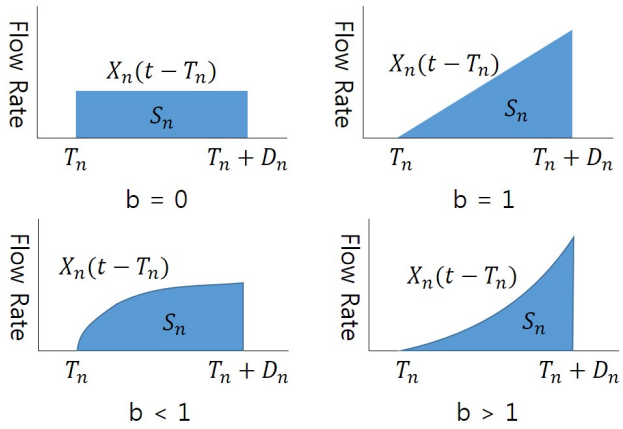

Fig. 5. Various shapes of the shots defined by a power function, $a(t - T_n)^b$.

V. Proposed Traffic Generation Method

The objective of our method is to generate network traffic that has the similar characteristics of real data center network traffic at the flow-level. The traffic characteristics considered are flow arrival rate, intra-rack traffic ratio, flow size and flow duration. Moreover, the proposed method is designed to address total flow transmission rate and its burtiness at a specific link. To achieve the objective, we utilized the analysis results of data center network and traffic model mentioned in the Section III and IV. The output of the method is a traffic matrix that contains a snapshot of the concurrent network traffic at a given data center network represented in a list of triplets (a source host, a destination host, and traffic volume). The generated traffic matrix is used to send commands to actual traffic generators such as iPerf [16] at a given time frame. There could be multiple traffic generators used to generate actual network traffic at the packet level. In this case, a single traffic matrix generator coordinates the multiple traffic generators. Fig.6 shows the overall system structure diagram.

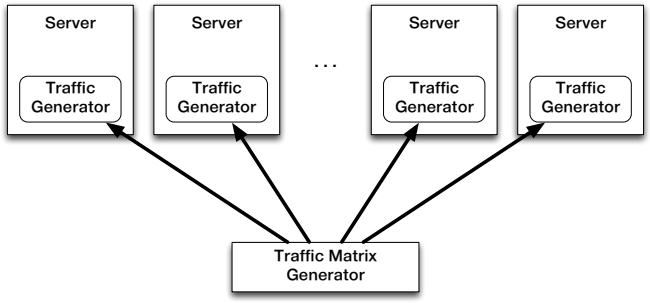

Fig. 6. Overall system structure

The procedure to generate a traffic matrix is composed of six-steps: (1) Recognizes network topology, (2) Decides new flow arrivals, (3) Generates flows with 5-tuples, (4) Assigns flow duration and size, (5) Injects flow transmission rate, and (6) Generates traffic matrix. By injecting proper characteristics through each step, the proposed method generates a traffic matrix that contains a set of flows similar with the real network traffic. To generate network traffic, the overall process could be repeated after a certain period of time. The overall steps and required inputs are depicted in Fig.7. The details of each generation step is described in later sections.

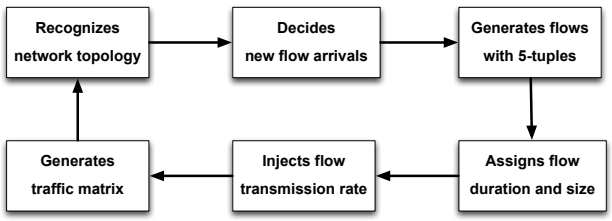

Fig. 7. Overall process to generate traffic matrix

A. Recognizes network topology

The first step in the generation of the network traffic matrix is recognizing the structure of a target data center network. The purpose of this step is to abstract physical network elements and their relationships into a form that can be handled by the subsequent steps of traffic matrix generation. To represent the network topology, we use a graph, G(V, E), that contains vertices and edges. The vertices represent network elements such as hosts, switches, and routers. The edges represent links between the various network elements in V. In addition to a graph, our method receives meta-data as input. The meta-data provide information about vertices and edges such as the type of network elements, the element IP address, and the link capacity.

A recognized target network topology is used to extract relationships such as hosts located in the same rack, a subnet, or a pod for fat-tree topology. Typical data center networks consist of multiple servers, switches, routers, and additional network elements. In the canonical tiered CISCO architecture for data center networks, several servers connect to the same switch, and they form a rack.

The output of this step is two functions, Hosts() and HList(s, d). The Hosts() function returns a list that contains all the hosts in the given network. HList(s, d) returns a list of hosts in a given hop count d from a given host s. Using the HList(s, d) function, we can easily obtain the list of hosts that have the same location properties. For example, the hosts in the same rack as $Host_A$ will be returned by HList($Host_A$, 1). Fig.8 visualizes the role of Hosts() and HList(s, d).

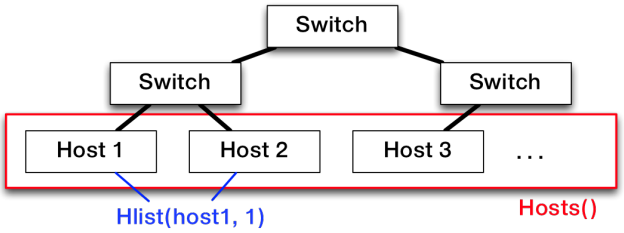

Fig. 8. The visualization of Hosts() and HList(s, d) functions.

B. Decides new flow arrivals

In this step, the number of new flow arrivals will be decided. Typically, data center networks show different usage ratios by time as mentioned in Section III. To apply this characteristic, we assumed that the flow arrivals follow a homogeneous Poisson process. $\lambda(t)$ is a function that returns average flow arrival rate. We can decide the number of new flow arrivals using a random number generator of Poisson distribution with $\lambda(t)$. N can be represented mathematically as (5) where $f_p(k; \lambda(t))$ is a Poisson distribution with arrival rate $\lambda(t)$.

$$N \sim f_p(k; \lambda(t)) = Pr(X = k) \tag{5}$$

C. Generates flows with 5-tuples

As the output of the previous step, the number of flows that should be generated at a given time t is decided. In this step, flow instances that contain quintuples are generated, such as source IP, destination IP, source port, destination port, and protocol. A set of newly generated flow instances is denoted as F_{new}, and the set of flows that have been existed at a previous time is denoted as F_{t-1}. The set of flows at a given time t is represented as (6).

$$F_t = F_{t-1} \cup F_{new} \tag{6}$$

Other important inputs for this step are R_{int} and R_{tcp}. R_{int} is a ratio of flows in which the source and the destination of a flow are located in the same rack. R_{tcp} is a ratio of flows that uses TCP protocol. These inputs are used to decide the source hosts, destination hosts, and protocols of newly generated flows.

The first step in deciding 5-tuples for each flow instance in F_{new} is to choose the source IP address and destination IP address. A source host s is selected from all the hosts in a particular network using a random number generator that follows a uniform distribution. To select the destination host, a two-step selection procedure is used. The first step determines whether a destination host is located in the same rack using the Bernoulli trial that follows $B(\infty, R_{int})$. If a destination host is located in the same rack, an actual destination host d is randomly selected from hosts returned by HList(s, 1) defined at the previous step. Otherwise, a destination host d is randomly selected from {Hosts() - HList(s, 1)}.

The port numbers of both source and destination hosts are randomly selected from a range of integers (e.g. from 1 to 65535). A protocol p is selected from the TCP or UDP using the Bernoulli trial that satisfies $B(\infty, R_{tcp})$. To prevent a duplication of existing flow instances, the new flow must be unique within the flow set F_t. Flow duplication can be checked using (7), where f_i and f_j are flow instances. If a duplication has occurred, all quintuples are regenerated.

$$f_i \nexists \forall f_j \in (F_{t-1} \cup F_{new}) \tag{7}$$

This procedure can be expanded to address various data center network topologies. For example, flows in Fat-tree topology can be categorized into three groups: intra-switch flows, intra-pod flows, and inter-pod flows [5]. By adjusting the hop count d, a parameter of the HList(s, d) function, we can generate flows by inserting one more Bernoulli trial to decide the traffic ratio for intra-pod. Similarly, this step can be easily modified for other topologies.

D. Assigns flow duration and size

Through the previous steps, a set of traffic flow instances with 5-tuples is generated. However, the actual behavioral characteristics, such as flow duration and flow size, are not decided yet. In section III, we described the characteristics of flow duration and size for data center network traffic. In this section, we describe our mathematical model to address those characteristics.

The distribution of flow duration fits well with Pareto distribution, which was originally used to describe the allocation of wealth among individuals. A well-known “80-20 rule”, in which the larger portion of the wealth is owned by a smaller percentage of people, describes a social phenomenon. The flow duration characteristics are similar to the wealth case. 80% of flows last less than 11s, but 0.1% of flows last longer than 200s. By adjusting a shape parameter a_p and a scale parameter M_p for a Pareto distribution, we can generate a distribution similar to the distribution of flows collected from a real data center network. Using the two parameters, a_p and M_p, our method generates random numbers for each flow as its duration. If set the exception of flow duration, we can calculate a shape parameter a_p as (9) with an arbitrary scale parameter 0s. Note that the domain of Pareto distribution is [1, ∞], so we move the flow duration distribution by adding 1s during random number generation.

$$D_n \sim \overline{F}(x) = Pr(X > x) = \begin{cases} (\frac{x}{M_p})^{a_p} & \text{for } x \geq M_p \\ 1 & \text{for } x \leq M_p \end{cases} \tag{8}$$

$$\mathbb{E}[D_n] = \frac{a_p}{a_p - 1} M \tag{9}$$

The inter arrival time between n-1th flow and nth flow can be decided using exponential distribution. We assumed that the flow arrivals follow a homogeneous Poisson process. Therefore, it is a correct mathematical model to address the inter arrival time. The exponential distribution with exception $1/N$ is represented on (10). After successfully generating the flow duration and inter-arrival time, we can define two functions, InitTime(f_n) and Duration(f_n) where f_n refers to nth flow instance. InitTime(f_n) returns the arrival time of a flow f_n, and Duration(f_n) returns the flow duration of a flow f_n.

$$T_n - T_{n-1} = X \sim f(x; N) = Ne^{-Nx} \tag{10}$$

Unfortunately, flow duration and size are not an independent relationship. Fig. 4 shows that flow duration and flow size are in a proportional relationship, but not in a linear relationship. Designing an accurate joint distribution of flow duration and size is very difficult. Rather than finding the joint distribution, we use two distributions for each flow duration and size. To reflect their correlation, we introduce a new random variable Y_n which represent mean transmission rate of a flow. Using Y_n, we can express the relationship between flow size and flow duration as (11). To find the distribution of Y_n, we analyzed mean transmission rate pattern from the traffic trace used in Section III (See Fig.9). We have concluded that Y_n can be modeled with a Gaussian distribution. Note that the random variable Y_n is independent with flow duration D_n.

$$S_n = Y_n \cdot D_n \tag{11}$$

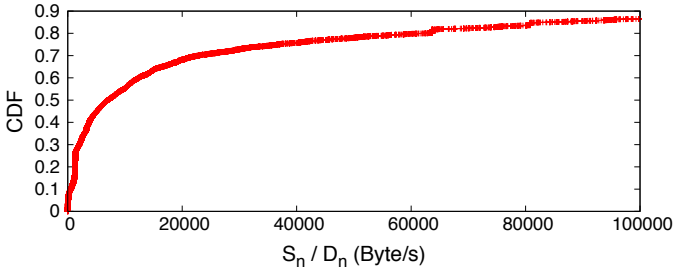

Fig. 9. The CDF of S_n/D_n

To use Poisson shot-noise process, $\mathbb{E}[S_n]$ and $\mathbb{E}[\frac{S_n^2}{D_n}]$ are required as inputs. Therefore, we need to design the distribution of Y_n to satisfy them. We can derive the mean and variance of Y_n as (12) and (13) from (11). As summary, Y_n is a random variable which follows a Gaussian distribution as same as (14). By generating Y_n, we can calculate S_n with already generated D_n. If the generated number is less than 80, we set S_n as 80 Bytes which is the minimum packet size of TCP.

$$\mathbb{E}[Y_n] = \frac{\mathbb{E}[S_n]}{\mathbb{E}[D_n]} \tag{12}$$

$$Var[Y_n] = \frac{\mathbb{E}[\frac{S_n^2}{D_n}]}{\mathbb{E}[D_n]} \tag{13}$$

$$Y_n \sim f_g(x, \mathbb{E}[Y_n], \sqrt{Var[Y_n]}) = \frac{1}{\sqrt{2\pi \cdot Var[Y_n]}} e^{-\frac{(x-\mathbb{E}[Y_n])^2}{2Var[Y_n]}} \tag{14}$$

E. Injects flow transmission rate

Several types of applications run in a data center. Each application shows different communication behaviors that affect the duration, size, and behavioral pattern of flows. For example, most of the flows show the ON-OFF pattern [1], in which a large number of packets are transmitted during the ON period. During the OFF period, there are small numbers of packets. To characterize this behavioral pattern, we have introduced the concept of "Shot" in section IV. Using Poisson shot-noise process, we can easily estimates the mean and variance at the specific link using (15) and (16) [13], [14]. It means that deciding the shape of shot affects to the generated traffic pattern severely.

$$\mathbb{E}[R(t)] = N \cdot \mathbb{E}[S_n] \tag{15}$$

$$Var[R(t)] = N \cdot \mathbb{E}[\int_0^{D_n} X_n^2(u)du] \tag{16}$$

The shapes of shots are differ according to the running applications. For the simplicity, we have assumed a singular type of shape of shot represented in a power function, $a(t - T_n)^b$. We can calculate the parameter a and b using (17) and (18) [13], [14]. From this property, we can freely adjust the average and variance (burtiness) of the generated traffic.

$$a = \frac{(b+1)S_n}{D_n^{b+1}} \tag{17}$$

$$b = k - 1 + \sqrt{k^2 - k}, \quad k = \frac{Var[R(t)]}{N \cdot \mathbb{E}[\frac{S_n^2}{D_n}]} \tag{18}$$

F. Generates traffic matrix

A traffic matrix contains traffic volume information among hosts at a certain amount of time. Generally, the traffic matrix is represented in the form of triplets (source host, destination hosts, and demand). A traffic matrix is similar with origin-destination (OD) flow matrix. Through the previous steps, we have obtained all the knowledge about traffic flows that should exist in a given network. By extracting the required information from F_t, we can generate the traffic matrix. Using (19), we can calculate traffic volume between a source host i and a destination host j during $T2 - T1$.

$$T_{i,j} = \sum_{n \in N} \int_{T1}^{T2} X_n(t - T_n)dt \tag{19}$$

Using the traffic matrix, we can generate a command for packet level traffic generators such as iPerf [16]. In our method, we periodically send commands to generate packet-level network traffic with the same amount of traffic volume specified in the traffic matrix. By adjusting the period of the proposed method, the granularity of traffic generation can be changed. For example, a short repeating period generates a finer grained traffic matrix than a long period. However, a short time period requires more frequent generation of the traffic matrix which requires more computing resources. Therefore, deciding a proper repeating period is necessary by the purpose.

VI. Implementation and Evaluation

The proposed method is implemented with Python programming language. To generate random numbers used in each step, Random, NumPy, and SciPy libraries for scientific computing is used. For the evaluation, we employed Mininet [20] network emulation tool for constructing a virtual DCN topology. With Mininet, we can create a realistic virtual network, which executes real Linux kernel, switch, and application code, on a single machine. To generate packets among hosts, we used iPerf [16], which is a network maximum throughput generators that can create TCP or UDP data streams. Unfortunately, iPerf have critical performance issue to generate all packets for a large number of hosts in a Mininet simulation environment. Due to the problem, we just generated traffic matrix rather than generating all packets.

For comparison, a real network traffic trace collected from a university data center network is used. The trace is the same with the trace used in [1]. The data center consists of 500 hosts, and the trace is collected on a link of an aggregate switch. The trace contains 38355 flows in 41,446,637 Kb during 200s. From this trace, we extracted several parameters used for the proposed method. Among 38355 flows, we have analyzed 8561 flows excluding flows terminated without FYN. The set of analyzed flows have 72,905 Kb. The extracted parameters for the traffic generation are summarized in the TABLE II.

TABLE II. Parameters for traffic generation

Name	Model	Parameters
New flow arrivals, N	Poisson	$\lambda(t) = 192$
Internal rack flow ratio	Bernoulli	$R_{int} = 0.8$
TCP flow ratio	Bernoulli	$R_{tcp} = 0.85$
Flow duration, D_n	Pareto	$a_p = 1.504$, $M_p = 1.0001$
Flow transmission rate, Y_n	Gaussian	$\mathbb{E}[Y_n] = 4303.69$, $Var[Y_n] = 69936.37$
Flow size, S_n	$Y_n \cdot D_n$	$\mathbb{E}[S_n] = 8517.97$, $\mathbb{E}[\frac{S_n^2}{D_n}] = 4891096388$
Average of total traffic rate, $\mathbb{E}[R(t)]$	Poisson shot-noise	2893130
Variance of total traffic rate, Var[R(t)]	Poisson shot-noise	$5.61442 * 10^{12}$

To compare with the result of our method, we assumed 500 servers deployed in canonical three-tired data center topology. The results of macro characteristics are obvious such as inter arrival time and intra rack flow ratio. To show that the results reflect the micro characteristics well, we analyzed the generation result in terms of flow duration and size. To compare the generated traffic with collected traffic, we generated 8565 flows. The comparison results are depicted on Fig.10, 11 and 12 in terms of flow size, duration, and size over duration.

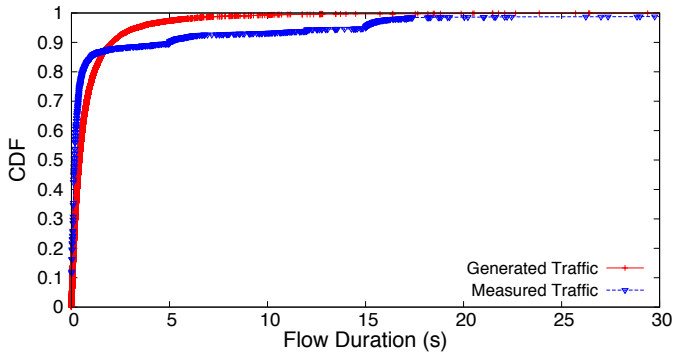

Fig. 10. The comparison of flow duration

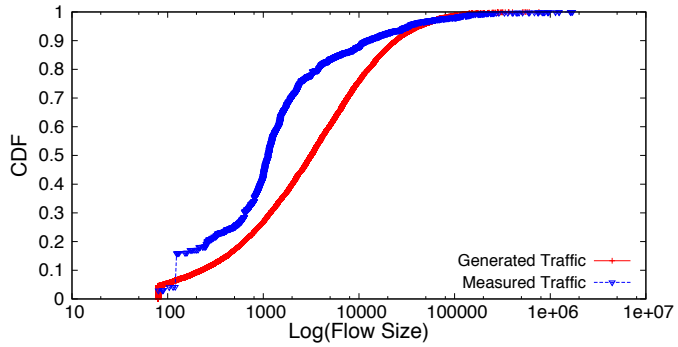

Fig. 11. The comparison of flow size

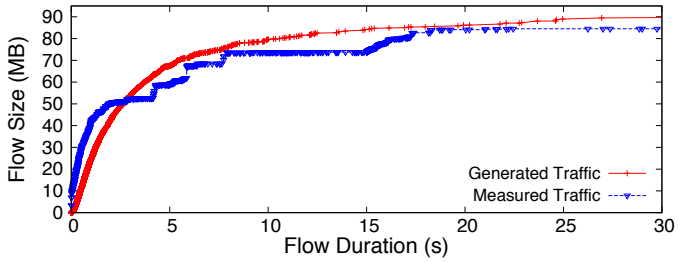

Fig. 12. The comparison of flow size according to flow duration

To inject shape of shots, we calculated a and b using (17) and (18). Finally, we can get the shape of the shot functions as $a(t - T_n)^{5.9786}$ (b=5.9786). Note that a changes depending on D_n and S_n. To evaluate total traffic rate, we've generated 50000 flows during 261s. The generated traffic during first 50s is excluded to remove unstable traffic of the bootstrapping period. Fig.13 visualizes the total traffic rate of the measured and generated traffic.

VII. Conclusion and Future Work

In this paper, we proposed a traffic matrix generation method with the consideration of the characteristics of data center network traffic at the flow level. The proposed method generates a traffic matrix rather than directly generating network packet. The benefits of our method are that easily reflects various network characteristics such as new flow arrival rate, flow ratio in the same rack, flow duration, and flow size. Moreover, the means and variance of total traffic rate are addressed easily with the flow level characteristics. Our method consists of six-steps: Recognizes network topology, Decides

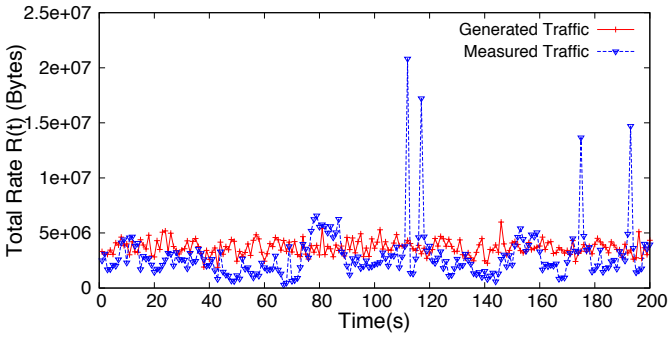

Fig. 13. The comparison of total traffic rate, R(t)

new flow arrivals, Generates flows with 5-tuples, Assigns flow duration and size, Injects flow transmission rate, and Generates traffic matrix. The fundamental idea of our method is to utilize random number generators that have the similar statistical characteristics with the real network traffic. The method is implemented with Python programming language by using several mathematical libraries, and it is implemented on a simulation environment. The generated traffic was compared to the real network traffic trace collected from a university data center, and the results show that it have the similar characteristics in terms of flow size, duration, and total traffic rate. The most powerful aspect of our method is that it can be easily implemented as software. Therefore, it could be used for the test and simulation to find the characteristics of a target data center.

For future work, the first task is to develop a packet generation mechanism. In this version of implementation, our method cannot generate network packets, and uses external packet-level traffic generators such as iPerf. The packet generation algorithm should reflect the characteristics of packets which is not reflected by flow-level such as inter-packet arrival time and packet size distribution. The second task is deep analysis of the characteristics of data center network traffic to find accurate parameters, and to extend the method to address non-TCP traffic, flow-dependency, etc.

REFERENCES

[1] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2010, pp. 267–280.

[2] T. Benson, A. Anand, A. Akella, and M. Zhang, "Understanding data center traffic characteristics," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 92–99, Jan. 2010.

[3] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference*. New York, NY, USA: ACM, 2009, pp. 202–208.

[4] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, "Elastictree: Saving energy in data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*. Berkeley, CA, USA: USENIX Association, 2010, pp. 17–17.

[5] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. New York, NY, USA: ACM, 2008, pp. 63–74.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 51–62, Aug. 2009.

[7] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "Portland: A scalable fault-tolerant layer 2 data center network fabric," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 4, pp. 39–50, Aug. 2009.

[8] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: A scalable and fault-tolerant network structure for data centers," in *Proceedings of the ACM SIGCOMM 2008 Conference on Data Communication*. New York, NY, USA: ACM, 2008, pp. 75–86.

[9] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube: A high performance, server-centric network architecture for modular data centers," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. New York, NY, USA: ACM, 2009, pp. 63–74.

[10] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: Evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, Dec. 1997.

[11] A. Feldmann, "Characteristics of tcp connection arrivals." Wiley, 1998.

[12] S. B. Fred, T. Bonald, A. Proutiere, G. Régnié, and J. W. Roberts, "Statistical bandwidth sharing: A study of congestion at flow level," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 111–122.

[13] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. Owezarski, "A flow-based model for internet backbone traffic," in *Proceedings of the 2Nd ACM SIGCOMM Workshop on Internet Measurment*, ser. IMW '02. New York, NY, USA: ACM, 2002, pp. 35–47.

[14] C. Barakat, P. Thiran, G. Iannaccone, C. Diot, and P. owezarski, "Modeling internet backbone traffic at the flow level," *Signal Processing, IEEE Transactions on*, vol. 51, no. 8, pp. 2111–2124, Aug 2003.

[15] S. Molnar, P. Megyesi, and G. Szabo, "How to validate traffic generators?" in *Communications Workshops (ICC), 2013 IEEE International Conference on*, 2013, pp. 1340–1344.

[16] M. Gates, A. Tirumala, J. Dugan, and K. Gibbs, *Iperf version 2.0.0*, Part of Iperf's source code distribution, NLANR applications support, University of Illinois at Urbana-Champaign, Urbana, IL, USA, May 2004. [Online]. Available: http://iperf.sf.net

[17] "MGEN - The Multi-Generator Toolset," http://manimac.itd.nrl.navy.mil/MGEN/, Feb. 2004.

[18] J. Sommers, H. Kim, and P. Barford, "Harpoon: A flow-level traffic generator for router and network tests," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 392–392, Jun. 2004.

[19] K. V. Vishwanath and A. Vahdat, "Swing: Realistic and responsive network traffic generation," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 712–725, Jun. 2009.

[20] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. New York, NY, USA: ACM, 2010, pp. 19:1–19:6.