Locality Matters! Traffic Demand Modeling in Datacenter Networks

Zhiwen Liu
Tsinghua University
liu-zw20@mails.tsinghua.edu.cn

Mowei Wang
Tsinghua University
wang.mowei@outlook.com

Yong Cui
Tsinghua University
cuiyong@tsinghua.edu.cn

ABSTRACT

Understanding and modeling traffic demand characteristics in datacenter networks is of great importance for datacenter network optimization. However, prior traffic models are oversimplified and insufficient in capturing the complex locality properties of traffic demand. We analyze real-world traffic traces and discover strong dependency between the spatial attributes (*source*, *destination*) and non-spatial attributes (*interarrival time*, *flow size*) of traffic demand. We propose *Lomas* to model the joint distribution of multi-dimensional traffic demand attributes and generate synthetic traces. *Lomas* is a novel extension of hierarchical *Bayes* model that can represent the relationships among these attributes as a dependency graph. We validate *Lomas* by showing its ability to recreate the flow-level traffic demand patterns of real-world traffic traces. Our approach can be easily adapted to different datacenters with heterogeneous traffic demand patterns, making it a convenient tool for practitioners to utilize.

CCS CONCEPTS

• **Networks** → **Network performance modeling**;

KEYWORDS

Traffic demand modeling; Locality; Hierarchical Bayes model

ACM Reference Format:

Zhiwen Liu, Mowei Wang, and Yong Cui. 2022. Locality Matters! Traffic Demand Modeling in Datacenter Networks. In *6th Asia-Pacific Workshop on Networking (APNet'22), July 1–2, 2022, Fuzhou, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3542637.3542639

APNet 2022, July 1–2, 2022, Fuzhou, China

ACM ISBN 978-1-4503-9748-3/22/07…$15.00
https://doi.org/10.1145/3542637.3542639

1 INTRODUCTION

Understanding the network traffic demand patterns is critical for datacenter network optimization. For example, the skewness of traffic matrics should be considered when designing datacenter topologies to ensure high bandwidth utilization [1, 8, 18, 20, 21, 25, 40], and flow-level characteristics (flow-size distributions, interarrival-time distributions, etc.) are important for designing congestion control [4, 22, 26, 28, 32, 33, 42], flow scheduling [2, 5, 7, 13] and load balancing [3, 23].

There is a battery of prior works focused on analyzing traffic patterns, dating back to the 1990s. Some researchers developed a number of models to capture the statistical temporal traffic patterns (burstiness on different time scales) based on *Poisson* or *'self-similar'* assumptions [14, 16, 27, 34, 39]. Others focused on statistical spatial traffic patterns (skewness of traffic matrix) and studied how to measure and estimate the traffic matrix [31, 41]. However, most of these models revolve around the statistical characteristics of entire traffic matrices, which are deficient in accuracy and granularity. Recently, an enlightening work was proposed to model the temporal and non-temporal traffic characteristics simultaneously, the problem is that it is on the packet level and it excludes well-concerned information like packet size [6].

In datacenter optimization scenarios, systematically understanding the multi-dimensional patterns of **flow-level traffic demand** is more important, given that every attribute of traffic demand can affect datacenter performance, e.g., *Flow-Completion Time* (*FCT*) has a strong dependence on *flow size* [15]. Flow-level traffic demand refers to the need to transfer data from one end-host to another with arbitrary flow-size and interarrival-time attributes. By analyzing traffic demand, researchers can learn traffic patterns exclude the bias introduced by network protocols and algorithms (e.g., load balancing) used during data collection.

Lacking an ideal model to help understand the flow-level traffic demand patterns, researchers concede to the *all-to-all* communication pattern with equal interarrival-time and flow-size distribution as a *common practice* [1–4, 8, 18, 23, 25, 28, 33], which is an easy-to-use solution inspired by [1, 4, 18]. The assumption behind this approach is that spatial attributes (*source*, *destination*) and non-spatial attributes (*interarrival time*, *flow size*) are **independent**, which is contrary to the

widely believed locality properties [6, 9, 37]. Besides, even though the *all-to-all* pattern may be a good representation for MapReduce/Hadoop applications [2], it's unsuitable to be used as a universal-adapted model considering the heterogeneous mix of services across datacenters. Besides, an *all-to-all* pattern could hardly be the worst case for scenarios like load balancing, where unbalanced demand patterns are of greater concern. To address this problem, researchers propose many other communicating patterns to simulate real-world scenarios, such as stride for HPC applications and staggered-prob for intra, inter-rack differentiation [2]. However, all these common practices are over simplified.

In order to systematically understand the requirements for traffic demand modeling, we carry out a comprehensive analysis of real-world traces [9]. We find strong **dependency** between spatial attributes (*source*, *destination*) and non-spatial attributes (*interarrival time*, *flow size*) of flow-level traffic demand, resulting in complex multivariate distribution and various **locality** properties. Traffic demand within different *src-dst* pairs is quantitatively imbalanced and behaviorally heterogeneous, e.g., traffic demand among some end-hosts is composed of dense elephant flows, while among others are consist of sparse mice flows. These locality properties should be considered in earnest when designing a traffic demand model. Our findings show that independently modeling the marginal distribution of **multi-dimensional** traffic demand is unsuitable, and they also reveal that the traffic demand modeling problem is complicated and challenging (§ 2).

In this paper, we propose an approach (named *Lomas*) to model and recreate the complex traffic demand patterns. *Lomas* is a generative model capable of learning the **joint distribution** of multi-dimensional traffic demand attributes (*source*, *destination*, *interarrival time*, *flow size*). Instead of assuming that all variables are dependent on all other variables, which tend to overfit observed data and be computationally infeasible, our approach represents the complex probabilistic relationships among these attributes as a commom-sensical and expressive dependency graph [35]. We extend the hierarchical *Bayes* model utilizing a latent functional space to capture the distribution of non-spatial attributes (*interarrival time*, *flow size*) for each *src-dst* pair, which simplifies the traffic demand modeling problem by relating traffic demand patterns to the latent functionalities of both sending-node and receiving-node (§ 3).

We evaluate *Lomas* using two real-world traffic traces [9]. The evaluation results prove *Lomas* can capture the complex locality properties, and generate synthetic traces with closer distribution distance to the ground truth compared with *common practice* (§ 4). *Lomas* can be adapted to different datacenters to discover the flow-level traffic demand patterns as well as the skewness of traffic matrics, which is of great importance in designing network protocols, algorithms, and datacenter topologies. *Lomas* is also useful to generate synthetic benchmarks considering the limited amount of publicly available traces (§ 5). We hope that our approach can serve as a *"handy tool"* to the research community for better design and evaluation before deployment.

2 MOTIVATION

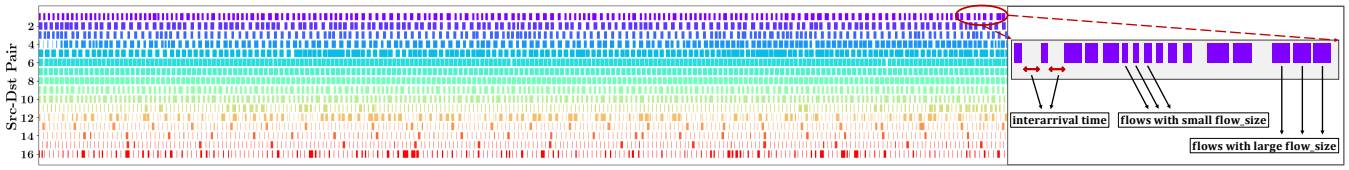
In this section, we first describe the fine-grained locality properties existing in real-world traces and the deficiency of *common practice*. Then, we analyze the requirements for a traffic-demand model to capture these locality properties.

2.1 Locality Properties in Real-world Datacenter Traffic Traces

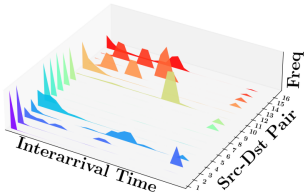
We derive the flow-level traffic demand from two publicly available real-world packet traces (named EDU1 and EDU2) [9]. The term *flow* in the context of this paper means the observed 4-tuple (*src*, *dst*, *interarrival time*, *flow size*) with a chosen long inactivity timeout to determine when a flow starts or ends [4], and the 4-tuple represents the sending node, the receiving node, flow interarrival time, and flow size respectively. Because of space constraints, we take EDU1 as an example to illustrate our analysis results, and EDU2 has similar results.

Figure 1a visualizes a fraction of the trace within 16 representative *src-dst* pairs. In Figure 1a, each row represents one *src-dst* pair with a corresponding color, and each rectangle represents a flow-arrival event (a flow) within the given source and destination. The width of each rectangle represents the *flow size*, and the blank between two adjacent rectangles represents the *interarrival time*. We observe significant difference in *interarrival-time* (Figure 1b) and *flow-size* (Figure 1c) distributions across *src-dst* pairs. Besides, we sum up the entries over the entire trace file, resulting in the total demand distribution within these 16 *src-dst* pairs (Figure 1d), which reveals the skewness of total demand [6], i.e., some *src-dst* pairs transfer more data than others. Finally, we investigate the degree (indegree plus outdegree) of each node in EDU1 (we plot a subset in Figure 1e for space constraints), and the result shows that the degree distribution is also highly skewed. We observe that the majority of nodes have low degree but a few have relatively high degree, indicating the degree distribution is right-skewed in this case.

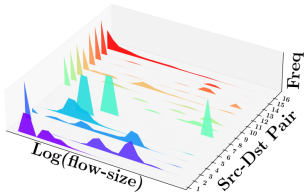
Conventionally, flows are generated assuming that the source and the destination are chosen randomly. The *interarrival time* is independently sampled from a *Poisson* process with parameter λ equal to the empirical full-sample mean, and *flow size* is independently sampled from the empirical cumulative distribution function (CDF) [1–4, 8, 18, 23, 25, 28, 33]. These simple assumptions are contrary to the fact that



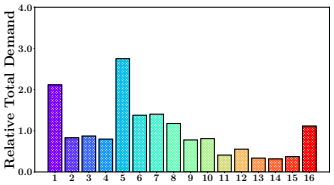
(a) Visualization: real-world trace from EDU1



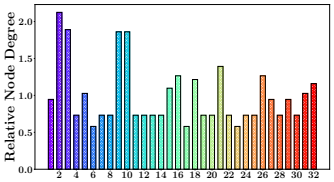
(b) Interarrival distribution



(c) Size distribution

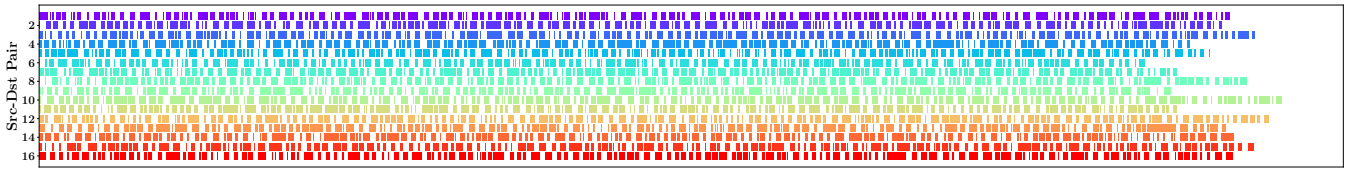


(d) Total demand per src-dst

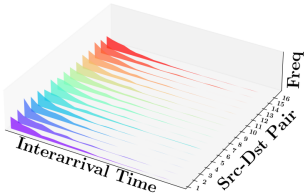


(e) Degree of each node

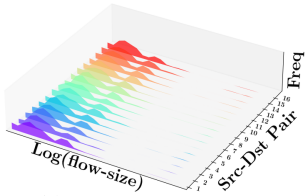
Figure 1: Real-world traffic demand: (a) Per src-dst traffic demand in each row. Each rectangle represents a flow. Width represents flow size. Blank represents interarrival time. Interarrival-time distributions within the 16 src-dst pairs are in (b), and flow-size distributions are in (c). (d) shows total demand and (e) shows degree of each node.



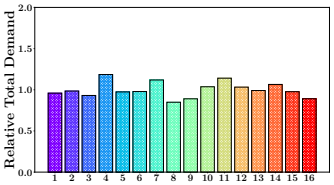
(a) Visualization: synthetic trace by common practice



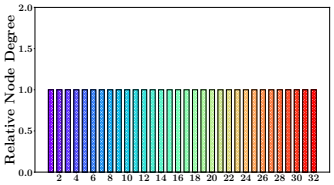
(b) Interarrival distribution



(c) Size distribution



(d) Total demand per src-dst



(e) Degree of each node

Figure 2: Traffic demand generated by common practice

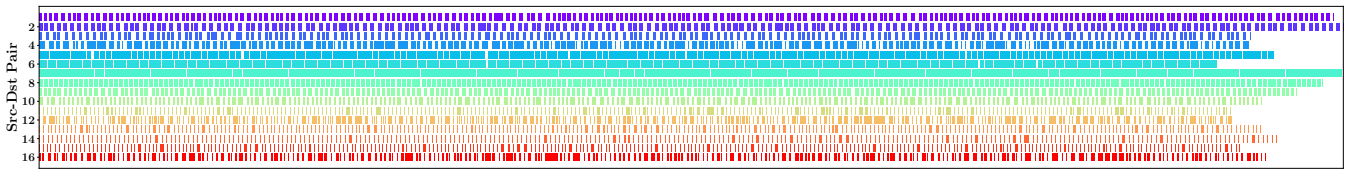
the multi-dimensional attributes of traffic demand have heterogenous probability distributions across different *src-dst* pairs. As a result, the traffic demand generated by *common practice* fails to capture the locality properties in real-world traffic traces. Comparing Figure 2a to Figure 1a we can find obvious and intuitive difference in *interarrival-time* and *flow-size* patterns. And comparing Figures 2b, 2c, 2d and 2e with their counterparts in Figures 1 respectively we can see the huge difference between the *common practice* and the ground truth, which proves that the *common practice* is unsuitable to model the complexities of real-world traffic demand in datacenter networks.

2.2 Requirements and Methodology for Traffic Demand Modeling

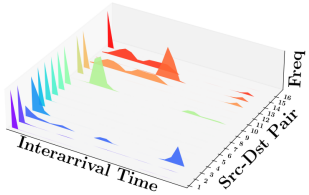
Considering the complex and heterogeneous traffic demand patterns in datacenter networks, we conclude that an ideal traffic-demand model should have the ability to capture the following locality properties:

(1). Heterogeneous arrival patterns within differenct *src-dst* pairs: traffic demand within different source nodes and destination nodes presents different *interarrival-time* and *flow-size* distributions.
(2). Imbalanced total demand across different *src-dst* pairs: some end-hosts tend to transfer more data to certain peers than others.
(3). Asymmetric node degree distribution in datacenter networks: some of the nodes have low degree while the other nodes have relatively high degree.

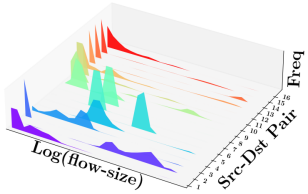
These locality properties motivate us to consider the intrinsic characteristics of source nodes and destination nodes in datacenters. The difference of traffic demand patterns across *src-dst* pairs can be related to the latent functionalities of both sending-nodes and receiving-nodes. For example, inside the social-network datacenters, the size of network flows sending from cache nodes are significantly larger than Hadoop nodes; and per-second flow rates of cache follower nodes are also much larger than Hadoop nodes [37].



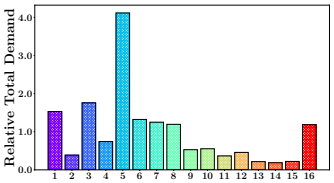
(a) Visualization: synthetic trace by *Lomas*



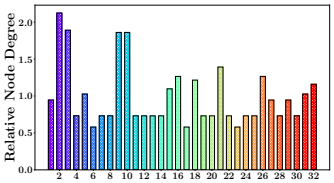
(b) Interarrival distribution



(c) Size distribution



(d) Total demand per src-dst



(e) Degree of each node

Figure 3: Traffic demand generated by *Lomas*

Based on our observations, we utilize a hierarchical *Bayes* method to learn the latent functionalities distribution of each *src-dst* pair (a combination of sending-node and receiving-node) as well as the distribution over the traffic demand patterns of each latent functionality. In our model, a *src-dst* pair is represented by a distribution of functionalities, and a functionality is featured by a distribution of flow types. As a result, our model can inference the *interarrival-time* and *flow-size* distribution for each *src-dst* pair based on the latent functionalities, thus capturing the heterogeneous locality properties. And introducing the latent space brings in the following advantages:

(1). Dimensionality reduction. We can use latent functionalities as low-dimensional representations of a huge number of *src-dst* pairs (proportional to the square of the number of nodes), which makes it possible to tackle the scale of modern datacenters.

(2). Transferability. The latent functionality distributions in datacenters are tied to the datacenter services. Therefore, a traffic demand model with latent functionality layer can be transferable to other datacenters provided no significant variation in datacenter services.

Besides, we are not the first to use hierarchical *Bayes* method to model the multivariable dependencies. Similar methodologies have been applied to natural language processing, e.g., Latent Dirichlet Allocation (LDA) is a three-level *Bayes* model mainly used for document classification [11]. However, to the best of our knowledge, such approaches have not been used in the context of traffic demand modeling in datacenter networks.

3 DESIGN

In this section, we first formulate the traffic demand modeling problem. Then, we go into details of our model (named *Lomas*) design and training. Finally, we illustrate the generative process using the trained model to get synthetic traces.

3.1 Problem Formulation

We organize the flow-level traffic demand as flow sequences (denoted by $\mathcal{A}$) according to flow arrival time. Each flow is a 4-tuple (*src*, *dst*, *interarrival time*, *flow size*). Both *src* and *dst* are discrete variables, while *interarrival time* and *flow size* are continuous variables. Before we utilize *Bayes* method to build our model, we have to address several challenges: *i*) how to infer the latent functionalities of each *src-dst* pair using observed flow sequences; *ii*) how to deal with discrete and continuous variables simultaneously in one model. All the notations used in this paper are summarized in Table 1.

Table 1: Notations used in the paper

Symbol	Explanation
M	number of nodes in a datacenter
R	number of *src-dst* pairs (equal to M^2)
$\mathcal{A}$	flow sequences in the entire trace
$\mathcal{A}_r$	*sending flows* within *src-dst* pair r
N_r	number of *sending flows* in $\mathcal{A}_r$
$a_{r,n}$	*n-th sending flow* in $\mathcal{A}_r$
$z_{r,n}$	functionality index for $a_{r,n}$
V	number of non-repeating *sending flow* types in $\mathcal{A}$
K	hyperparameter for the number of functionalities
α	hyperparameter for functionality distribution θ
η	hyperparameter for flow-pattern distribution β
θ	a $R \times K$ matrix for the functionality distribution
β	a $K \times V$ matrix for the flow-type distribution

In order to infer the latent functionality distributions of corresponding *src-dst* pairs, we divide the entire trace $\mathcal{A}$ into subgroups according to the *src* and *dst* attributes of every flow. We assume the number of nodes in a datacenter is M, and the number of *src-dst* pairs is M^2. For each *src-dst* pair (denoted by r, $r = 1, 2, \ldots, R$, and $R = M^2$), we observe a collection of *sending flows* (denoted by $\mathcal{A}_r$, a subset of $\mathcal{A}$) which are sent from src_r to dst_r. Considering the *sending flows* in $\mathcal{A}_r$ share the same *src* and *dst* attributes, we can use a 2-tuple (*interarrival time*, *flow size*) to represent each *sending*

flow in $\mathcal{A}_r$. By doing so, we relate functionality distribution of each *src-dst* pair to its *sending flows*, and therefor the *sending flows* can be used for latent functionalities inference.

In order to deal with continuous variables, we discretize *interarrival time* and *flow size* attributes of every *sending flow* by creating sets of contiguous quantile-based bins for computational simplification, considering that minor fluctuations in these two variables do not really make much difference in traffic demand patterns. We index all the non-repeating discretized *sending flows* in $\mathcal{A}$ with $\{1, \ldots, V\}$ [1], and we take each *sending flow* as a distinct type of flow (i.e., a certain size of flow that need to be sent at some interarrival time). Based on the finite flow types resulted by discretization, we can represent *sending flows* using one-hot vectors. Thus, the *n-th sending flow* in $\mathcal{A}_r$ with flow-type v is represented by a *V-vector* $a_{r,n}$ such that $a^v_{r,n} = 1$ and $a^u_{r,n} = 0$ for $u \neq v$.

3.2 Functionality-Based Modeling

Using the functionality-based hierarchical *Bayes* analysis framework, the goal of *Lomas* is to find the probability distribution of *src-dst* pairs in the latent functional space (θ) that can best explain the observed traffic demand patterns. Assuming the dimensionality of the latent space is K (K is a hyperparameter), the modeling procedure can be defined as follows, *i*) modeling the heterogeneous mix of K types of functionalities for each *src-dst* pair; *ii*) revealing the distribution of *sending flow* types for each functionality.

Assuming the dimensionality of latent space (K) is known and fixed, the functional probability distributions for a datacenter with R *src-dst* pairs are parameterized by a $R \times K$ matrix θ, and θ_r (*r-th* row of θ) is a *K-dimensional vector* with $\sum_{k=1}^{K} \theta_r^k = 1, (r = 1, 2, \ldots, R)$. If there is no *sending flows* within *src-dst* pair r, we set θ_r to $\vec{0}$. For each *sending flow* $a_{r,n}$ of $\mathcal{A}_r$, our model assumes it belongs to a latent functionality $z_{r,n}$ where $z_{r,n}$ follows a *Multinominal* distribution with parameter θ_r. And $z_{r,n}$ is a K-dimension one-hot vector with $z^k_{r,n} = 1$ and $z^j_{r,n} = 0$ for $j \neq k$, which means the functionality index of $z_{r,n}$ is k. Besides, the probability distributions of *sending flow* types are parameterized by a $K \times V$ matrix β where $\beta_{k,v} \propto \sum_{r=1}^{R} \sum_{n=1}^{N_r} p(a^v_{r,n} = 1, z^k_{r,n} = 1)$. Both θ and β are probability matrices to be estimated.

And we use two hyperparameters α and η to specify our prior beliefs about functionality sparsity for each *src-dst* pair r and flow-type sparsity for each functionality k. Both α and η are associated with *Dirichlet* distributions [38] and their values will influence the way the *Dirichlets* generate *Multinominal* distributions of θ and β. A higher α value will lead to distributions that center around averages for θ, indicating *src-dst* pairs are made up of more functionalities. While lower α values will lead to distributions that are more dispersed. Parameter η works in an analogous way for β. Conditional on the α and η parameters, the likelihood for N_r *sending flows* ($\mathcal{A}_r$) of a *src-dst* pair r is:

$$p(\mathcal{A}_r, z, \theta, \beta | \alpha, \eta) =$$
$$\iint p(\beta|\eta)p(\theta|\alpha)(\prod_{n=1}^{N_r} \sum_{z_{r,n}} p(z_{r,n}|\theta)p(a_{r,n}|z_{r,n}, \beta))d\theta d\beta$$

For model training, we utilize the approximate inference technique named *Gibbs Sampling* [19] to estimate parameter θ and β. *Gibbs Sampling* is a special case of the *Markov Chain Monte Carlo* method [12, 17]. And there are other approximate techniques ready for use, like variational inference [24]. We choose *Gibbs Sampling* for its flexibility and simpleness.

3.3 Generating Synthetic Traces

The generative process using our trained model is shown in Algorithm 1. *Lomas* allows researchers to synthesize traces in different settings (e.g., time windows, number of nodes), which is particularly useful given the limited and nonflexible historical traces. For example, when a larger M is chosen, our model will pad the original $R \times K$ matrix θ into a larger $R' \times K$ matrix θ'. Besides, our model can be used to simulate different levels of traffic demand (e.g., demand peaks in e-commerce datacenters during Shopping Festival) by adjusting the *interarrival time* probability distribution function.

Algorithm 1: Generative process of *Lomas*

input : time window t; number of nodes M.
1 **for** each sending-node i **do**
2 **for** each receiving-node j **do**
3 *Initialization src-dst:* $r \leftarrow M \times (i-1) + j$;
4 *Initialization time:* $cur\_t \leftarrow 0$;
5 *Initialization flow id:* $n \leftarrow 1$;
6 **while** $cur\_t < t$ **do**
7 Draw functionality index $z_{r,n} \sim Multinominal(\theta_r)$;
8 Draw *sending flow* type $\hat{a}_{r,n} \sim Multinominal(\beta_{z_{r,n}})$;
9 Convert discretized *interarrival time* and *flow size* bins to continuous values;
10 *Update:* $cur\_t \leftarrow cur\_t$ + *interarrival time*;
11 *Update:* $n \leftarrow n + 1$;
output : a synthetic trace with the time window from 0 to t.

In generating process, we iteratively generate R subgroups of *sending flow* $\widehat{\mathcal{A}}_r$ to get the synthetic trace $\widehat{\mathcal{A}}$. For the *n-th sending flow* ($\hat{a}_{r,n}$) in $\widehat{\mathcal{A}}_r$, we first sample its functionality index $z_{r,n}$ from the *Multinominal* distribution with parameter θ_r, and then the *sending flow* $\hat{a}_{r,n}$ can be chosen conditioned on $z_{r,n}$ and β (line 7-8). After that, we convert the discretized *interarrival time* bins and *flow size* bins to continuous values by assuming these variables are distributed evenly within each bin (line 9), and this method was also used in [10]. Finally, we accumulate the *interarrival time* within each $\widehat{\mathcal{A}}_r$

[1] V = (number of *interarrival-time* bins) × (number of *flow-size* bins).

and move on to the next *src-dst* pair when the accumulated time overran the given time window (line 10).

4 EVALUATION

In this section, we first describe the evaluation setups and our model (*Lomas*) implementation. Then, we present the preliminary evaluation results with different settings.

4.1 Setups

Datasets. We evaluate *Lomas* using two real-world traces (EDU1, EDU2) [9]. We extract over 0.2 million flows from EDU1 and over 1 million flows from EDU2 (§ 2).

Metric. We choose *Kolmogorov–Smirnov* (*KS*) statistic [30] as a preliminary metric to quantify the distance between the empirical distribution function of the ground truth $\mathcal{A}$ and the synthetic trace $\widehat{\mathcal{A}}$. If distributions are identical, the *KS*-distance is 0, and its maximum possible value is 1. For *interarrival time* and *flow size* attributes, we calculate their average *KS*-distance across all *src-dst* pairs respectively.

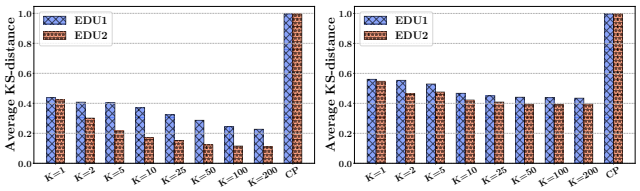
Implementation. Our model is implemented in *Python* using *Gensim* [36]. We use 8 different K values (number of latent functionalities) to test the performance of *Lomas*. And we set both α and η to 'auto', which is a feature of *Gensim* that enable automatically parameter learning.

Baseline. We compare *Lomas* with the *common practice* (abbreviated to *'CP'*), where the parameter λ for *interarrival time* equals to the empirical full-sample average values, and the *flow size* follows the empirical full-sample CDF.

4.2 Results

Figure 3 visualizes the generated flow-level traffic demand using *Lomas*. Each row in Figure 3a represent the same *src-dst* pair as Figure 1a, and we can see clearly from Figure 3a that our synthetic trace has similar *interarrival-time* distribution and *flow-size* distribution (Figure 3b and 3c) to the ground truth. And statisticaly, the total demand within each *src-dst* pair and the degree of each node (Figure 3d and 3e) are also similar to the ground truth.

The evaluation results using *KS*-distance in Figure 4 show two main observations: *i*) The distributions of generated traces by *Lomas* are closer to the ground truth (with lower *KS* values) compared with *common practice* (*CP*), and the *KS*-distance decreases with higher K. We notice that the average *KS*-distance for *flow size* is higher than *interarrival time*, and the reason is that our discretization method underfits the irregular *flow size* distributions and therefore causes high discretization error. We will optimize it in future work. *ii*) A relatively small K (e.g., $K = 25$) is sufficient to model the traffic demand generated by thousands of nodes, which proves the applicability of *Lomas* to large scale datacenters.



(a) KS-distance for *interarrival* **(b) KS-distance for *flow size***

Figure 4: Evaluation Results

5 APPLICATION AND FUTURE WORK

Lomas is a generative model capable of learning the complex locality properties of flow-level traffic demand in datacenter networks. Given the limited amount of publicly available datasets, *Lomas* can be particularly useful to generate synthetic benchmarks and recreat representative traffic demand patterns. First, our approach can help understand the flow-level locality properties and facilitate the design of datacenter network protocols and algorithms. Second, *Lomas* can discover the skewness of traffic matrices, which is helpful for optimizing the datacenter topologies to ensure high bandwidth utilization. Finally, our trained models can generate synthetic traces retaining the traffic demand patterns, and these synthetic benchmarks can be used by researchers to evaluate their designs in different settings (e.g., for longer traces or higher level of traffic demand)..

Besides, there are several future directions we wish to explore. First, we plan to reevaluate the existing datacenter optimization proposals [2–4, 8, 18, 29, 33] using our synthetic benchmarks to understand the robustness and applicability of these popular designs. Second, we will explore the possibilities to introduce new techniques to enhance our model performance, e.g., a recurrent neural networks (*RNN*) module to capture inter-flow correlations. Finally, we want to apply our approach to model more real-world traffic traces (e.g., traces from Facebook [37]) and release our trained and desensitized models to facilitate future research.

6 CONCLUSION

We present *Lomas*, an approach for traffic demand modeling in datacenter networks. Preliminary evaluation results have shown that *Lomas* holds promise for capturing the flow-level locality properties in traffic demand as well as synthesizing traces. Besides, *Lomas* can be easily adapted to different datacenters with heterogeneous traffic demand patterns.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This work was supported by the National Key R&D Program of China under Grant 2018YFB1800303, NSFC Project of China under Grant 62132009 and Grant 61872211, and by Tsinghua University-China Mobile Communications Group Co.,Ltd. Joint Institute. *(Corresponding author: Yong Cui.)*

REFERENCES

[1] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. 2008. A scalable, commodity data center network architecture. In *SIGCOMM*. ACM, 63–74.

[2] Mohammad Al-Fares, Sivasankar Radhakrishnan, Barath Raghavan, et al. 2010. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI*. USENIX Association, 281–296.

[3] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, et al. 2014. CONGA: distributed congestion-aware load balancing for datacenters. In *SIGCOMM*. ACM, 503–514.

[4] Mohammad Alizadeh, Albert G. Greenberg, David A. Maltz, et al. 2010. Data center TCP (DCTCP). In *SIGCOMM*. ACM, 63–74.

[5] Mohammad Alizadeh, Shuang Yang, Milad Sharif, et al. 2013. pFabric: minimal near-optimal datacenter transport. In *SIGCOMM*. ACM, 435–446.

[6] Chen Avin, Manya Ghobadi, Chen Griner, et al. 2020. On the Complexity of Traffic Traces and Implications. In *SIGMETRICS*. ACM, 47–48.

[7] Wei Bai, Li Chen, Kai Chen, et al. 2014. PIAS: Practical Information-Agnostic Flow Scheduling for Data Center Networks. In *HotNets*. ACM, 25:1–25:7.

[8] Hitesh Ballani, Paolo Costa, Raphael Behrendt, et al. 2020. Sirius: A Flat Datacenter Network with Nanosecond Optical Switching. In *SIGCOMM*. ACM, 782–797.

[9] Theophilus Benson, Aditya Akella, and David A. Maltz. 2010. Network traffic characteristics of data centers in the wild. In *IMC*. ACM, 267–280.

[10] Shane Bergsma, Timothy Zeyl, Arik Senderovich, et al. 2021. Generating Complex, Realistic Cloud Workloads using Recurrent Neural Networks. In *SOSP*. ACM, 376–391.

[11] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *JMLR* 3, Jan (2003), 993–1022.

[12] Chain Monte Carlo. 2004. Markov chain monte carlo and gibbs sampling. *Lecture notes for EEB* 581 (2004), 540.

[13] Li Chen, Kai Chen, Wei Bai, et al. 2016. Scheduling Mix-flows in Commodity Datacenters with Karuna. In *SIGCOMM*. ACM, 174–187.

[14] Mark Crovella and Azer Bestavros. 1997. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Trans. Netw.* 5, 6 (1997), 835–846.

[15] Nandita Dukkipati and Nick McKeown. 2006. Why flow-completion time is the right metric for congestion control. *SIGCOMM Comput. Commun. Rev.* 36, 1 (2006), 59–62.

[16] Mark W. Garrett and Walter Willinger. 1994. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *SIGCOMM*. ACM, 269–280.

[17] Charles J Geyer. 1992. Practical markov chain monte carlo. *JSTOR* 7, 4 (1992), 473–483.

[18] Albert G. Greenberg, James R. Hamilton, Navendu Jain, et al. 2009. VL2: a scalable and flexible data center network. In *SIGCOMM*. ACM, 51–62.

[19] Thomas L Griffiths and Mark Steyvers. 2004. Finding scientific topics. *Proceedings of the National academy of Sciences* 101, suppl 1 (2004), 5228–5235.

[20] Chuanxiong Guo, Guohan Lu, Dan Li, et al. 2009. BCube: a high performance, server-centric network architecture for modular data centers. In *SIGCOMM*. ACM, 63–74.

[21] Chuanxiong Guo, Haitao Wu, Kun Tan, et al. 2008. Dcell: a scalable and fault-tolerant network structure for data centers. In *SIGCOMM*. ACM, 75–86.

[22] Mark Handley, Costin Raiciu, Alexandru Agache, et al. 2017. Re-architecting datacenter networks and stacks for low latency and high performance. In *SIGCOMM*. ACM, 29–42.

[23] Keqiang He, Eric Rozner, Kanak Agarwal, et al. 2015. Presto: Edge-based Load Balancing for Fast Datacenter Networks. In *SIGCOMM*. ACM, 465–478.

[24] Matthew D. Hoffman, David M. Blei, Chong Wang, et al. 2013. Stochastic variational inference. *JMLR* 14, 1 (2013), 1303–1347.

[25] Simon Kassing, Asaf Valadarsky, Gal Shahaf, et al. 2017. Beyond fat-trees without antennae, mirrors, and disco-balls. In *SIGCOMM*. ACM, 281–294.

[26] Gautam Kumar, Nandita Dukkipati, Keon Jang, et al. 2020. Swift: Delay is Simple and Effective for Congestion Control in the Datacenter. In *SIGCOMM*. ACM, 514–528.

[27] Will E. Leland, Murad S. Taqqu, Walter Willinger, et al. 1993. On the Self-Similar Nature of Ethernet Traffic. In *SIGCOMM*. ACM, 183–193.

[28] Yuliang Li, Rui Miao, Hongqiang Harry Liu, et al. 2019. HPCC: high precision congestion control. In *SIGCOMM*. ACM, 44–58.

[29] Hongzi Mao, Malte Schwarzkopf, Shaileshh Bojja Venkatakrishnan, et al. 2019. Learning scheduling algorithms for data processing clusters. In *SIGCOMM*. ACM, 270–288.

[30] Frank J Massey Jr. 1951. The Kolmogorov-Smirnov test for goodness of fit. *JASA* 46, 253 (1951), 68–78.

[31] Alberto Medina, Nina Taft, Kavé Salamatian, et al. 2002. Traffic matrix estimation: existing techniques and new directions. In *SIGCOMM*. ACM, 161–174.

[32] Radhika Mittal, Vinh The Lam, Nandita Dukkipati, et al. 2015. TIMELY: RTT-based Congestion Control for the Datacenter. In *SIGCOMM*. ACM, 537–550.

[33] Behnam Montazeri, Yilong Li, Mohammad Alizadeh, et al. 2018. Homa: a receiver-driven low-latency transport protocol using network priorities. In *SIGCOMM*. ACM, 221–235.

[34] Vern Paxson and Sally Floyd. 1994. Wide-Area Traffic: The Failure of Poisson Modeling. In *SIGCOMM*. ACM, 257–268.

[35] Judea Pearl and Thomas Verma. 1987. The Logic of Representing Dependencies by Directed Graphs. In *AAAI*. Morgan Kaufmann, 374–379.

[36] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, 45–50.

[37] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, et al. 2015. Inside the Social Network's (Datacenter) Network. In *SIGCOMM*. ACM, 123–137.

[38] Jayaram Sethuraman. 1994. A constructive definition of Dirichlet priors. *Statistica sinica* (1994), 639–650.

[39] Murad S. Taqqu, Walter Willinger, and Robert Sherman. 1997. Proof of a fundamental result in self-similar traffic modeling. *SIGCOMM Comput. Commun. Rev.* 27, 2 (1997), 5–23.

[40] Yiting Xia, Xiaoye Steven Sun, Simbarashe Dzinamarira, et al. 2017. A Tale of Two Topologies: Exploring Convertible Data Center Network Architectures with Flat-tree. In *SIGCOMM*. ACM, 295–308.

[41] Yin Zhang, Matthew Roughan, Nick G. Duffield, et al. 2003. Fast accurate computation of large-scale IP traffic matrices from link loads. In *SIGMETRICS*. ACM, 206–217.

[42] Yibo Zhu, Haggai Eran, Daniel Firestone, et al. 2015. Congestion Control for Large-Scale RDMA Deployments. In *SIGCOMM*. ACM, 523–536.