Differentiated Transmission based on Traffic Classification with Deep Learning in DataCenter

Keke Zhu*, Gengbiao Shen*, Yong Jiang*, Jianhui Lv*, Qing Li†, Mingwei Xu*
*Tsinghua University, †Southern University of Science and Technology
zkk17@mails.tsinghua.edu.cn, sgb16@mails.tsinghua.edu.cn, lvjianhui2012@163.com, jiangy@sz.tsinghua.edu.cn,
liq8@sustech.edu.cn, xumw@tsinghua.edu.cn

***Abstract*—Production datacenters generally collect diverse applications with differentiated requirements, e.g., low latency and high throughput. Therefore, datacenter transmission schemes must strive to meet these requirements. However, despite significant efforts, prior solutions are either ineffective to satisfy different requirements or costly to apply. Besides, no scheme can accommodate to all diverse data center scenarios or dynamic traffic patterns. In this paper, we propose SmartTrans, a deep learning based latency-aware differentiated transmission service, including three main components. First, SmartTrans utilizes deep learning methods for traffic classification and flow size rank prediction. Second, according to the classified results of flows, SmartTrans adopts multilevel priority queues to execute differentiated scheduling. Third, SmartTrans enlarges the switch buffer to increase the capacity of datacenter networks (DCN), which effectively fights against the traffic burst and improves the throughput of latency-insensitive flows. We evaluate SmartTrans with several real workloads. Experiment results show that SmartTrans achieves both the low average flow completion time (FCT) for latency-sensitive flows and the high throughput for latency-insensitive flows.**

***Index Terms*—flow scheduling, data center, deep learning, latency, throughput**

I. Introduction

There are a large number of applications in contemporary datacenters[1]. Traffic in the datacenter is usually a mix of diverse flows with different requirements which consist of the latency-sensitive traffic, e.g., online gaming and web search, and the throughput-oriented traffic such as Hadoop, MapReduce or virtual machine migration[2]. To guarantee high-quality access to the variety of applications and services with the appropriate performance, it is necessary to take advantage of all limited resources in datacenters. In modern datacenters, for satisfying the low latency requirement of some applications, network devices in datacenters are designed with the shallow buffer to reduce the maximum queuing delay that packets may suffer from. However, as other applications require the high bandwidth, DCN need a large capacity to absorb the traffic burst and avoid the packet drop.

To optimize the performance of datacenter, many schemes have been proposed. Some of them use policies that approximate or implement well-known disciplines to minimize the average FCT, e.g., PDQ[3], PASE[4], and PIAS[5]. PDQ, and PASE need know the prior knowledge of flow size to approximate Shortest Job First (SJF) discipline. PIAS improves pFabric[6] by a dynamic priority assignment without the prior experience, which approximates Least Attained Service (LAS) according to the flow packets that have been already sent so far. Although some of them substantially improve overall average FCT. However, they do not consider the application types and just assign priority according to flow size, which may result in that a larger latency-sensitives flow is queued behind many small latency-insensitive flows. Besides, flow control schemes such as DCTCP[7], HULL[8] and D2TCP[9] try to reduce FCT without relying on flow information by keeping low queue occupancy using explicit congestion notification (ECN). While these schemes generally improve latency, they are fundamentally constrained because they can never precisely estimate the flow rate to use so as to schedule flows to minimize FCT while ensuring that the network is fully utilized.

Based on the analysis of current solutions, we claim that a good datacenter transmission control scheme should meet the following design principles:

- The low latency for the user-interactive flows and the high throughput for the throughput-hungry flows.
- The universality for diverse network scenarios.

We argue that differentiated transmission can satisfy all the requirements mentioned above. In this paper, we present SmartTrans, a deep learning based latency-aware differentiated transmission control service in datacenters, which is a potential universal solution to simplify the transmission in the dynamic and diverse network scenarios.

Here are three key designs of SmartTrans. First, SmartTrans utilizes deep learning method at end hosts to predict flow information, because the blind scheduling without knowing flow information is likely to result in sub-optimal results. Second, based on the obtained flow information, SmartTrans use multilevel priority queues to execute differentiated scheduling. Finally, SmartTrans enlarges switch buffer space to improve the capacity of DCN.

The rest of this paper is organized as follows. In the next section, we present the details of SmartTrans. We evaluate SmartTrans and compare it with other state-of-the-art schemes in Section III. Finally, we briefly conclude SmartTrans in Section IV.

Corresponding author: Qing Li (liq8@sustech.edu.cn).

Annex to ISBN 978-3-903176-28-7 © 2020 IFIP

II. Design of SmartTrans

We show the details of our design in this section. There are two main parts of SmartTrans: a deep learning-based method to predict flow information and multilevel priority queues to execute differentiated scheduling.

A. Flow Classification and Flow Size Prediction at End-host

For differentiated scheduling, datacenters need to know the information of flows. Different from the previous works which assume flow information as a prior, SmartTrans use deep learning method to predict flow information. It considers flow type firstly to ensure low latency of key applications and flow size secondly for optimizing average FCT.

Notice that SmartTrans predicts which rank or interval the flow size belongs to instead of the accurate flow size. There are two reasons: 1). It is difficult to predict the accurate flow size, we have used a regression model to predict the size and find that the error is huge. Increasing the hidden size or layer number can help a bit, but the cost is expensive 2). There are limited number of queues in the modern switch, it is not necessary to know the accurate flow size. We just need to know which queue the flow belongs to according to its size. So we convert the regression problem to a classification problem.

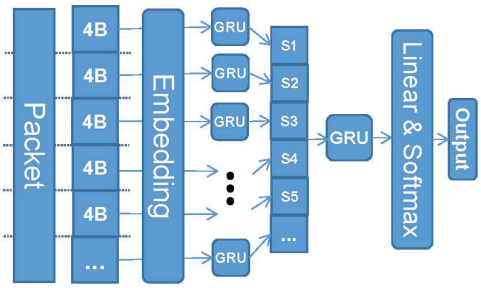


Fig. 1. Traffic classification and flow size prediction architecture.

The architectures of the traffic classification and flow size rank prediction neural network (TCFSNN) used in SmartTrans are shown in Fig. 1. Because of the sequential relationship between packets, SmartTrans chooses the recurrent neural network (RNN) model which commonly used in natural language processing. We can regard flow as a language with only 256 vocabularies as flow is made up of bytes and each byte can be translated as a number in 0∼255. In SmartTrans, each flow is treated as a communication conversation and each byte is a word between applications. Several words form a sentence which describes some meaningful information, such as four bytes of IP address. When SmartTrans finds a flow, it copies the first three IP packets of this flow and then lets the flow go. Copied packets are fed to TCFSNN to get the prediction of flow type and flow size rank. Note that TCFSNN uses copied packets to make prediction, it does not block the normal process of flows.

Here we detail the design of TCFSNN. There are one embedding layer, two bidirectional GRU layer and one full connect layer in TCFSNN. It converts the copied packets to digits and combines every four digits into a group to extract information such as IP address efficiently. Since the ordinary RNN suffers from the gradient vanishing, we use an improved RNN optimization: the Gated Recurrent Unit (GRU)[10], which adopts a gating mechanism to retain long-term memory during propagation. There are two gates in GRU, where z and r denote the update and the reset gates. The output of GRU h_t at time t is a linear interpolation between the previous output h_{t-1} and the candidate activation $\hat{h_t}$, which can be expressed as

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{1}$$

where an update gate z_t decides the values that the unit updates at time t, and controls the keeping of the old memory and the adding of the new memory. z_t updates according to

$$z_t = \sigma(W_z x_t + U_z h_{t-1}) \tag{2}$$

Where W and U represent the trainable parameters of this gate. The calculation of the candidate $\hat{h_t}$ is similar to that of the traditional recurrent unit

$$\hat{h_t} = tanh(W_h x_t + U_h(r_t \odot h_{t-1})) \tag{3}$$

Where reset gate r_t determines how the previous memory state affects the present candidate state. r_t computed similarly to the update gate

$$r_t = \sigma(W_r x_t + U_r h_{t-1}) \tag{4}$$

TCFSNN uses the focal loss that is employed to solve the problem of the imbalance multi-classification problem while training neural networks, because the number of the small flow is much more than the large flow in datacenters. The main idea of the focal loss is to focus more on hard-to-distinguish samples and reduce the impact of simple samples. The focal loss for K-classification can be simply calculated as

$$FL(y_l) = -\alpha(1 - softmax(y_l))^{\gamma} log(softmax(y_l)) \tag{5}$$

where l is the true label, α is the weight assigned to the class, and γ is the focusing parameter.

By leveraging the TCFSNN, SmartTrans can get flow information instead of assuming it as a prior. Although the prediction is not absolutely right, it also benefits the scheduling and indeed improves the performance of SmartTrans as the experiments in Section III shows.

B. Multilevel Priority for the Differentiated Transmission

SmartTrans leverages the multilevel priority both at end-hosts and switches for the differentiated transmission to ensure the user experience while optimizing average FCT of all flows. The multilevel priority means that it uses multiple factors instead of one to determine the priority of flows, since one-dimension scheduling may lead to a sub-optimal result.

In SmartTrans, two factors are taken into account:

- **Flow Type**: Different applications have distinct requirements, thus datacenters had better offer service by flow type. Compared to the equal treatment of all flows, it is a better approach that lets latency-insensitive flows give way to latency-sensitive flows. However, same minimum RTO (minRTO) for all flows will lead to timeout and trigger retransmission for flows that make way. Thus, SmartTrans allows the network operator to set different minRTO values according to application types. In

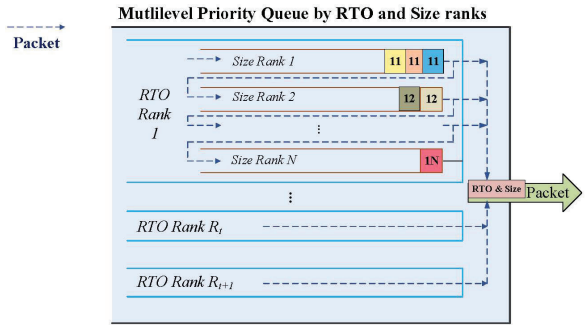


Fig. 2. Multilevel Priority Queues. Applications in different RTO rank have different delay-tolerance and minRTO. Each RTO rank has one or more size rank to optimize FCT. Each square in size rank represents a flow which is tagged with RTO rank and size rank.

SmartTrans, each minRTO value represents an RTO rank and applications which have the same minRTO belong to same RTO rank. RTO rank 1 has the minimum minRTO and flows in RTO rank 1 have the least delay tolerance and have a higher priority. SmartTrans regards the flow type as the most significant factor to assign priority.

- **Flow Size**: Flow size is an important factor affecting FCT. When an elephant flow is queued in front of mice flows, mice flows' FCT will increase sharply and even lead to timeout. Therefore, SmartTrans takes flow size as the second factor for determining priority to optimize the average FCT. As Fig. 2 shows, a RTO rank includes one or more size ranks in the switch. Each size rank is a priority queue and one or more priority queues form a RTO rank.

In SmartTrans, hosts keep multilevel priority queues as well as network switches. TCFSNN gives the host the type and size rank information of every flow, then the host tags this information on DSCP filed of every packet in flow and places packets to the corresponding priority queues according to the type and size as Fig. 2 shows. Switches in DCN have the same config of multilevel priority as host. When switch receives a packet, it gets the flow information from DSCP field of the packet and places the packet to the multilevel priority queue according to the information. Considering that TCFSNN needs time to calculate, packets are given the highest priority before TCFSNN outputs the result.

We introduce that how SmartTrans determine the thresholds between different size rank when a RTO rank has more than one size rank. SmartTrans leverages existing optimization software to derive the thresholds numerically.

1) Problem Formulation: We assume that there are K priority queues and each one is denoted as Q_i, $1 \leq \mathrm{i} \leq K$, where Q_1 has the highest priority. We denote the percentage of flows in Q_i as θ_i and the cumulative density function of flow size distribution as $F(x)$. Let $\alpha_i = F^{-1}(\sum_{l=1}^{i} \theta_l)$ denote the size rank threshold for Q_i and Q_{i+1}, $1 \leq \mathrm{i} \leq K-1$, where $F^{-1}(x)$ is the inverse function of $F(x)$. Denote the $E[S_i] = \int_{C_{i-1}}^{C_i} F^{-1}(x)dx$ as the average size of a given flows in the queue Q_i, where $C_i = \sum_{l=1}^{i} \theta_l$ is the cumulative sum of θ. Thus, given the flow arrival rate as λ, then the flow arrival rate to Q_i is $\lambda_i = \lambda E[S_i]$.

The service rate for a queue depends on whether the queues with higher priorities are all empty. Thus, the highest priority queue Q_1 has the capacity $\mu_1 = \mu$ where μ is the service rate of the link. The fraction of time that Q_1 is idle is $(1-\rho_1)$ where $\rho_i = \lambda_i/\mu_i$ is simply the utilization of Q_i. Thus, the service rate of Q_2 is $\mu_2 = (1-\rho_1)\mu$ since its service rate is μ given that Q_1 is empty. Thus, we have $\mu_i = \prod_{j=0}^{i-1}(1-\rho_j)\mu$ with $\rho_0 = 0$. Thus, $T_i = 1/(\mu_i - \lambda_i)$ which is the average delay of Q_i assuming $M/M/1$ queues.

For a flow f_i in Q_i, it experiences the delays in different priority queues up to the i-th queue. So the average FCT for $T(f_i)$ is upper-bounded by: $\sum_{q=1}^{i} T_q$. Thus we have an optimization problem of choosing an optimal set of θ_i to minimize the average FCT of flows on this bottleneck link. The problem can be formulated as:

$$\begin{aligned} \min_{\{\theta_i\}} \quad & \sum_{l=1}^{K} \left(\theta_l \sum_{q=1}^{l} T_q\right) \\ \text{s.t.} \quad & \sum_{l=1}^{l=K} \theta_l = 1 \\ & \theta_i \geq 0, i = 1, ..., K \end{aligned} \tag{6}$$

2) Solution Method: Since there are generally often less than 8 priority queues in modern switches, the problem can be solved numerically and quickly with existing optimization software. We use a global optimization toolbox available in SciPy. For 4, 8-queues cases, the problem is on average solved in 12s, 30s respectively on a modern PC with i5-6400 CPU. In summary, given flow arrival rate λ and flow size distribution $F(x)$, we can compute the thresholds relatively quickly.

III. Evaluation

In this section, we design experiments to evaluate SmartTrans's performance and compare it with other schemes.

A. Traffic Classification and Flow Size Prediction

We design experiments to evaluate the accuracy and efficiency of the traffic classification and the flow size prediction. However, it is hard to find a DCN dataset that could provide both enough application types and flow numbers. Hence, four different datasets are used in this experiment, one for flow classification and three for flow size prediction. In the flow classification experiment, we refine data from a real-world public dataset, UNB CIC VPN-nonVPN dataset [11]. There are about 20M flows in our training set. Several common flow types are selected such as FTP, P2P, Browsing, Email, Spotify and Youtube in flow classification experiments.

Flow size prediction is performed with three public datasets of real traffic from two universities[12] and an academic building dataset at Dartmouth College. Each dataset consists of packet trace for more than 3 million flows. We train 10 epochs in our experiments and apply the 5-fold cross-validation for the dataset. F1-score is used to estimate our methods, where F1-score is:

$$F1 - score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{7}$$

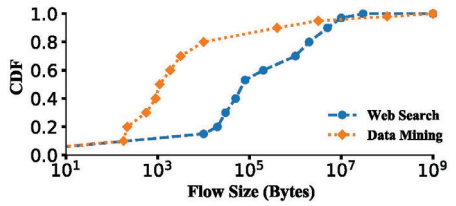


Fig. 3. Traffic distributions used for evaluation.

TABLE I
TRAFFIC CLASSIFICATION RESULT

Metrics / Protocol	Precision(%)	Recall(%)	F1-Score(%)
Browsing	99.57	99.88	99.72
Email	99.87	100	99.94
FTP	99.35	100	99.67
P2P	100.00	99.77	99.88
Spotify	99.09	100	99.54
Youtube	100	98.34	99.16

Where $Precision$ refers to the fraction of relevant instances among the total retrieved instances. $Recall$ refers to the fraction of relevant instances retrieved over the total relevant instances.

Table I shows the Precision, Recall and F1 score of the traffic classification experiment. We find that SmartTrans achieves nearly perfect classification of all kinds of flows. The impressive performance of TCFSNN is owing to the ability of deep learning methods to deal with complex sequences. Then we evaluate the performance of SmartTrans for predicting flow size rank in different scenarios. Lets K denotes the number of priority queues. We vary the thresholds from 10KB to 10MB when K = 2. The predicted result shows that it achieves nearly 98% accuracy in the test case, because the network only needs to predict whether the flow is elephant or mice flow. Next, we show the predicted result of K = 4 in Table II. As K increases, predict accuracy shows a downward trend, because it is getting closer to predicting the actual flow size, rather than the interval where the flow size is. However, it is also acceptable and could get considerable gain when K is not large. Considering that modern datacenter switches typically have eight queues, the accuracy is acceptable when $K \leq 8$. We also find that it always has an excellent performance on predicting the elephant flow, because the number of the elephant flow is less than the mice, which makes it easier to find consistent features.

B. Large Scale Simulation Setup

In this part, we test SmartTrans under datacenter topology by NS3 simulation. Our NS-3 based implementation of SmartTrans[13] has been open sourced to GitHub. We run the experiments on a DELL EMC PowerEdge R840 server.

TABLE II
FLOW SIZE PREDICTION RESULT WITH K=4

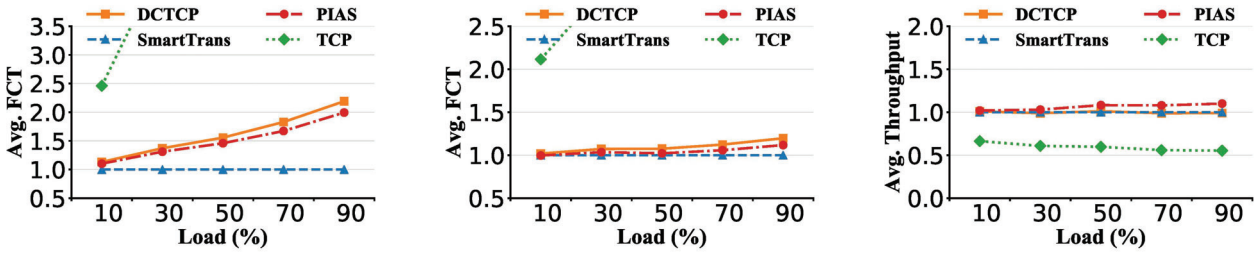
Metrics / FlowSize	Precision(%)	Recall(%)	F1-Score(%)
(0, 10KB]	95.99	87.31	91.45
(10KB, 5MB]	90.36	97.02	93.57
(5MB, 10MB]	99.26	100	99.63
(10MB, ∞)	100	100	100

Experiment Setting: We evaluate the performance of SmartTrans with the 8x8 leaf-spine topology. This topology has 8 leaf switches, 8 spine switches and 128 hosts. Each leaf switch is connected to 16 hosts and 8 spine switches using 10Gbps links, thus forming a 2:1 oversubscription network. The base end-to-end round-trip time across the spine (4 hops) is 85.2μs. SmartTrans aims to be a universal scheme to simplify traffic scheduling in diverse scenarios. So in this part, ECMP, the widely employed load balancing scheme, is used for load balancing. In our experiment, we simulate two widely-used realistic DCN traffic workloads: a web-search workload [7] and a data-mining workload [14]. Their cumulative distribution function of flow sizes are shown in Fig. 3. For DCTCP, we set the ECN marking threshold as 65 packets for 10Gbps links, which follows the parameter setting as [7] recommends. We assign every port a buffer whose size is 800 packets. There are usually eight priority queues in the datacenter switch. Unless specified, we use the first four queues as RTO Rank 1 which has a higher priority for latency-sensitive flows. The remaining four priority queues form RTO Rank 2. minRTO of flows in RTO Rank 1 and RTO Rank 2 are set as 5ms and 10ms, respectively.

1) Average Latency-Sensitive FCT: Fig. 4(a) and Fig. 5(a) show the average FCT of latency-sensitive flows under different workloads and load levels. From the figures, we see that SmartTrans delivers the best performance. By contrast, the performance of PIAS is varied. SmartTrans achieves about 100% and 30% lower FCT than PIAS for the web search and data mining workloads. It is expected, because SmartTrans employs the multilevel priority queues and places latency sensitives flows in high priority queues. In both two workloads, there are not many large flows on the same link concurrently. As a result, PIAS does not suffer from starvation as significantly. Hence, PIAS is up to 9.87% and 5.27% lower average FCT compared to DCTCP respectively.

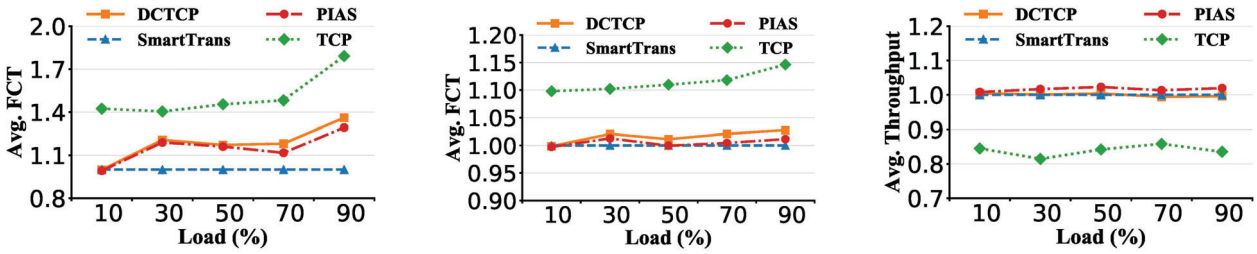
2) Overall Average FCT: We also compare the overall average FCT of these schemes at different loads, the results under every workload are shown in Fig. 4(b) and Fig.5(b). As the results show, when the load is low, PIAS sometimes performs slightly better than SmartTrans: it is within 0-1.2%. However, SmartTrans gradually shows an advantage than other schemes as the load increase and especially better at heavy load. There are 11.81% and 1.16% lower overall average FCT for SmartTrans than PIAS at 90% load in the web search and data mining respectively. Note that although the multilevel priority employed by SmartTrans damages the average FCT of latency-insensitive flows, the overall average FCT of SmartTrans is still comparable to DCTCP at low load and much better than PIAS and DCTCP under high load.

3) Average Throughput of Latency-Insensitive Flows: Generally, the optimization of latency-sensitive flows is at odds with the throughput of latency-insensitive flows. To evaluate the effect of using multilevel priority and cache on latency-insensitive flows, we show the average throughput of



(a) Avg. latency-sensitive FCT (b) Overall avg. FCT (c) Latency-insensitive flow throughput

Fig. 4. **Simulation results of the web-search workload in the symmetric topology (normalized to SmartTrans).**



(a) Avg. latency-sensitive FCT (b) Overall avg. FCT (c) Latency-insensitive flow throughput

Fig. 5. **Simulation results of the data-mining workload in the symmetric topology (normalized to SmartTrans).**

latency-insensitive flows in Fig. 4(c) and Fig. 5(c). The results show that average throughput of latency-insensitive flows in SmartTrans is comparable to PIAS at low load level and is 9.82% and 1.97% lower than PIAS at 90% load in web search and data mining respectively. It is expected because SmartTrans always places latency-insensitive flows behind of latency-sensitive flows and there are only four queues for latency-insensitive flows compared to eight queues of PIAS. But the loss is small and acceptable, we get a great gain for latency-sensitive flows and a better overall average FCT. The throughput of latency-insensitive flows is even slightly better than DCTCP in web search and data mining workloads. This is because we enlarge switch buffer to enhance the DCN capacity, which could effectively fight against the burst and reduces the packet dropping.

IV. Conclusion and Future Work

In this paper, we propose SmartTrans, a deep learning based latency-aware differentiated transmission control service for datacenter. SmartTrans uses deep learning methods to classify the flow type and predict the flow size while enlarging switch buffer to offer high throughput for bandwidth-hungry flows. Meanwhile, it uses the multilevel priority to improve the overall average FCT. Comprehensive experiments show that SmartTrans improves the average latency-sensitive FCT significantly and overall average FCT slightly while offering high throughput in all scenarios.

V. Acknowledgement

This work is supported by National Natural Science Foundation of China under grantNo. 61972189, Guangdong Province Key Area R&D Program under grant No. 2018B010113001, the project "PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)" and the Shenzhen Key Lab of Software Defined Networking under grant No. ZDSYS20140509172959989.

References

[1] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, "The nature of data center traffic: Measurements & analysis," in *ACM IMC*, 2009.

[2] S. Bashir, "Handling elephant flows in a multi-tenant data center network," Ph.D. dissertation, 09 2015.

[3] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *ACM SIGCOMM*, 2012.

[4] A. Munir, G. Baig, S. M. Irteza, I. A. Qazi, A. X. Liu, and F. R. Dogar, "Friends, not foes: Synthesizing existing transport strategies for data center networks," in *ACM SIGCOMM*, 2014.

[5] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and W. Sun, "PIAS: Practical information-agnostic flow scheduling for data center networks," in *ACM HotNets*, 2014.

[6] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pFabric: Minimal near-optimal datacenter transport," in *ACM SIGCOMM*, 2013.

[7] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data center TCP (DCTCP)," in *ACM SIGCOMM*, 2010.

[8] M. Alizadeh, A. Kabbani, T. Edsall, B. Prabhakar, A. Vahdat, and M. Yasuda, "Less is more: Trading a little bandwidth for ultra-low latency in the data center," in *USENIX NSDI*, 2012.

[9] B. Vamanan, J. Hasan, and T. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," in *ACM SIGCOMM*, 2012.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *EMNLP*, 2014.

[11] Gerard, Drapper, A. Gil, Habibi, M. Lashkari, A. Mamun, A., and Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *IN Proceedings of the 2nd International Conference on Information Systems Security and Privacy*, 2016.

[12] T. Benson, A. Akella, and D. Maltz, "Network traffic characteristics of data centers in the wild," in *ACM IMC*, 2010.

[13] "Smarttrans." [Online]. Available: https://github.com/Akeron-Zhu/CacheTrans

[14] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, "VL2: A scalable and flexible data center network," in *ACM SIGCOMM*, 2009.