TrafficShaper: Shaping Inter-Datacenter Traffic to Reduce the Transmission Cost

Wenxin Li, Xiaobo Zhou, Keqiu Li, *Senior Member, IEEE*, Heng Qi, and Deke Guo, *Senior Member, IEEE, Member, ACM*

***Abstract*— The emerging deployment of geographically distributed data centers (DCs) incurs a significant amount of data transfers over the Internet. Such transfers are typically charged by Internet service providers with the widely adopted *qth percentile charging model*. In such a charging model, the time slots with top $(100-q)$ percent of data transmission do not affect the total transmission cost and can be viewed as *"free."* This brings the opportunity to optimize the scheduling of inter-DC transfers to minimize the entire transmission cost. However, a very little work has been done to exploit those *"free"* time slots for scheduling inter-DC transfers. The crux is that existing work either lacks a mechanism to accumulate traffic to *"free"* time slots, or inevitably relies on prior knowledge of future traffic arrival patterns. In this paper, we present *TrafficShaper*, a new scheduler that shapes the inter-DC traffic to exploit the *"free"* time slots involved in the *qth percentile charging model*, so as to reduce or even minimize the transmission cost. When shaping traffic, *TrafficShaper* advocates a simple principle: *more traffic peaks* should be scheduled in *"free"* time slots, while *less traffic differentiation* should be maintained among the remaining time slots. To this end, *TrafficShaper* designs a pricing-aware control framework, which makes online decisions for inter-DC transfers without requiring a prior knowledge of traffic arrivals. To verify the performance of *TrafficShaper*, we conduct rigorous theoretical analysis based on *Lyapunov optimization* techniques, large-scale trace-driven simulations, and small-scale testbed implementation. Results from rigorous mathematical analyses demonstrate that *TrafficShaper* can make the transmission cost arbitrarily close to the optimum value. Extensive trace-driven simulation results show that *TrafficShaper* can reduce the transmission cost by up to 40.23%, compared with the state-of-the-art solutions. The testbed experiments further verify that *TrafficShaper* can realistically reduce the transmission cost by up to 19.38%.**

***Index Terms*— Inter-datacenter network, Lyapunov optimization, percentile pricing model.**

I. Introduction

LARGE-SCALE organizations, such as Google, Microsoft, and Amazon, have made huge investments in building geo-distributed data centers (DCs) to deliver their online services [1], [2]. A key feature of these services is that they continuously produce large volumes of data transmission among different DCs [3]. A recent survey [4] highlighted that 70% of the IT firms have huge data transmission among DCs, ranging from 1Gbps to 10Gbps, nearly half having 5Gbps or more — i.e., from 330 TB to 3.3 PB a month. Such huge data transmission incurs substantial cost for the service provider. In fact, the annual transmission cost is of up to hundreds of millions of dollars, which approximately equals to the power cost of DCs [5]. From the perspective of service provider, the fundamental objective is to reduce, or even minimize the transmission cost incurred by the inter-DC transfers.

Service providers typically purchase bandwidth from Internet Service Providers (ISPs) for their inter-DC transfers, while ISPs charge service providers based on the widely adopted *q-th percentile charging model* [6]–[9]. Such charging model can be described as follows: In a charging period of N time slots, the ISP samples the bandwidth usage that a service provider consumed in every time slot and sorts them in ascending order (each time slot is typically 5 minutes). Then, the q-th percentile of all samples is taken as the billed bandwidth. For example, if 95-th percentile charging is in use and the charging period is 30 days, then the billed bandwidth exactly equals to the bandwidth usage of the 8208-th sorted time slot ($95\% \times 30 \times 24 \times 60/5 = 8208$). Clearly, in such *$q$-th percentile charging model*, the time slots with top $(100-q)$ percent of data transmission actually do not affect the total transmission cost, and can be viewed as *"free"*. This provides an opportunity to reduce service provider's transmission cost by carefully scheduling their inter-DC transfers.

Further, the diverse time-sensitivities exhibited by different inter-DC transfers also motivate the design of new scheduling methods. For examples, *interactive transfers* are most sensitive to delay, *larger transfers* require to be done within several hours, while *background transfers* are without strict time requirements [1]–[3], [10]. Hence, we believe that, the transmission cost of a service provider can be effectively

Manuscript received April 11, 2017; revised October 10, 2017 and February 22, 2018; accepted March 11, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor A. Wierman. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1000205, in part by the State Key Program of National Natural Science of China under Grant 61432002, in part by the Joint Funds of the National Natural Science Foundation of China under Grant U1701263, in part by the NSFC under Grant 61672379, Grant 61772112, Grant 61702365, Grant 61425002, and Grant 61751203, in part by the Dalian High-level Talent Innovation Program under Grant 2015R049, in part by the Natural Science Foundation of Tianjin under Grant 17JCQNJC00700, in part by the National Natural Science Foundation of China under Grant 61772544, and in part by the National Basic Research Program (973 Program) under Grant 2014CB347800. *(Corresponding authors: Xiaobo Zhou; Heng Qi.)*

W. Li and H. Qi are with the School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China (e-mail: liwenxin@mail.dlut.edu.cn; hengqi@dlut.edu.cn).

X. Zhou and K. Li are with the Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin 300350, China (e-mail: xiaobo.zhou@tju.edu.cn; keqiu@tju.edu.cn).

D. Guo is with the College of System Engineering, National University of Defense Technology, Changsha 410073, China, and also with the Tianjin Key Laboratory of Advanced Networking, School of Computer Science and Technology, Tianjin University, Tianjin 300350, China (e-mail: guodeke@gmail.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TNET.2018.2817206

reduced, if inter-DC traffic can be accumulated to those *"free"* time slots as much as possible, while satisfying the deadline requirement of each traffic. In such a case, the optimal solution would be to schedule all *"free"* time slots as *traffic peaks*, and at the same time maintain *no traffic differentiation* among the remaining time slots.

Intuitively, it may be a step towards the right direction to design an offline inter-DC transfer scheduling method to obtain the optimal solution. However, such offline optimization inevitably relies on *prior* knowledge of traffic arrival patterns, which are actually unavailable in practice. To the best of our knowledge, no existing methods are in place to exploit the *"free"* time slots in the *q-th percentile charging model* to minimize the transmission cost as well as to guarantee deadlines for inter-DC transfers. *First*, state-of-the-art methods on inter-DC traffic either lack a mechanism to accumulate traffic to *"free"* time slots [10], [11], or cannot guarantee the deadlines of inter-DC transfers [12]. *Second*, although some Internet traffic scheduling methods investigated the impact of the percentile charging model, they either require prior knowledge of future traffic demand [7], or assume uniform deadline requirements for all traffic [13].

In this paper, we propose *TrafficShaper*, a new scheduler that aims to minimize the transmission cost of inter-DC transfers by fully exploiting the advantages of *"free"* time slots in *q-th percentile charging model* and the diverse deadline requirements among inter-DC transfers. *TrafficShaper* advocates to shape the inter-DC traffic to construct more traffic peaks during those *"free"* time slots, and maintain less traffic differentiation among the remaining time slots. To this end, *TrafficShaper* designs a pricing-aware online control framework to practically make scheduling decisions for inter-DC transfers, without prior knowledge of the traffic arrival patterns. Specifically, a stochastic optimization problem is formulated to guide the design of such control framework. This problem takes into account different percentile values across DCs, practical constraints of heterogeneous link capabilities, and different time-sensitivities of inter-DC transfers. Nevertheless, it is impractical to obtain an optimal solution for this problem, due to the unknown information of future traffic arrivals. This motivates *TrafficShaper* to transform it into a relaxed problem, which is then solved by designing an online control algorithm based on *Lyapunov Optimization* techniques [14], [15]. Such relaxation does not have much of an impact on the optimality of this online algorithm, in terms of both transmission cost and system stability. More precisely, results from rigorous theoretical analysis prove that *TrafficShaper* can arbitrarily approach the optimal solution within an $O(1/V)$ gap, where V is a control parameter representing how much we emphasize the transmission cost minimization compared to the deadline guarantee. To make *TrafficShaper* more practical, we further present a distributed manner to elaborate how it can be implemented in real-world inter-datacenter networks. Finally, we evaluate *TrafficShaper* using large-scale trace-driven simulations as well as small-scale testbed implementation. Simulation results show that compared to the Equal Split (ES) [13] and Shortest-Deadline-First (SDF) [16], [17] methods, *TrafficShaper* reduces the transmission cost by up to 26.19% and 40.23%, respectively, while accommodates considerable amount of inter-DC transfers with deadlines guaranteed. The testbed results further verify that *TrafficShaper* can practically reduce the transmission cost by up to 19.38%.

In summary, the main contributions of this paper include:

- We address the challenging problem of minimizing transmission cost under the *q-th percentile charging model*, when scheduling a large amount of inter-DC transfers with diverse time requirements and without any prior knowledge of future traffic arrival patterns.
- We present *TrafficShaper*, a new scheduler that exploits the *"free"* time slots in *q-th percentile charging model* to minimize the transmission cost of inter-DC transfers. Specifically, *TrafficShaper* employs a pricing-aware online control framework to schedule inter-DC transfers, and can provably approach a transmission cost that is arbitrarily close to optimum.
- We proceed to design a distributed method for implementing the *TrafficShaper* scheduler. Specifically, in the sender side, this distributed method determines how much bandwidth to be allocated to each flow; and in the receiver side, it decides the receiving rate for each flow; finally, the minimal value between sending rate and receiving rate is chosen as the rate for each flow.
- We conduct extensive trace-driven simulations as well as a small-scale testbed implementation to evaluate the performance of *TrafficShaper*. Both simulation and implementation results have shown that *TrafficShaper* is cost-effective and can provide deadline guarantees for inter-DC transfers, when compared to the state-of-the-art methods.

The rest of this paper is organized as follows. Section II introduces the background and motivation of *TrafficShaper*. Section III presents the system model and problem formulation. In Section IV, we describe the key component in *TrafficShaper* — pricing aware online control framework. In Section V, we evaluate and analyze the performance of *TrafficShaper*. We discuss current limitations of *TrafficShaper* and relevant future research in Section VI. Section VII summarizes the related work and Section VIII concludes this paper.

II. *TrafficShaper*: Background and Motivation

In this section, we first present the inter-DC network model. Then, we use an illustrative example to motivate the design of *TrafficShaper*. Finally, we show some practical factors faced by *percentile charging model*.

A. Inter-DC Network Model

In this paper, we consider an inter-DC network model where all DCs, geographically distributed across the world, connect to the ISPs with logical uplinks and downlinks, as shown in Fig. 1. To complete an inter-DC transfer, each DC only needs to send data to ISPs, and then the ISPs forward the data to the destination DC. Many recent studies [18], [19] have revealed that the ISPs' networks are non-blocking, and the network bottlenecks only appear between DCs and ISPs.

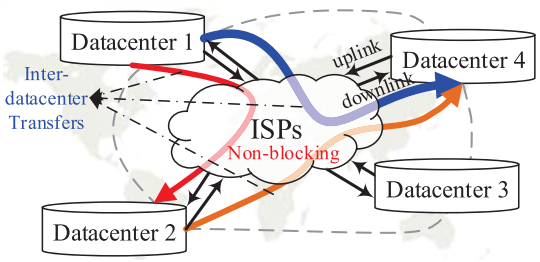


Fig. 1. An illustrative example of inter-DC network model.

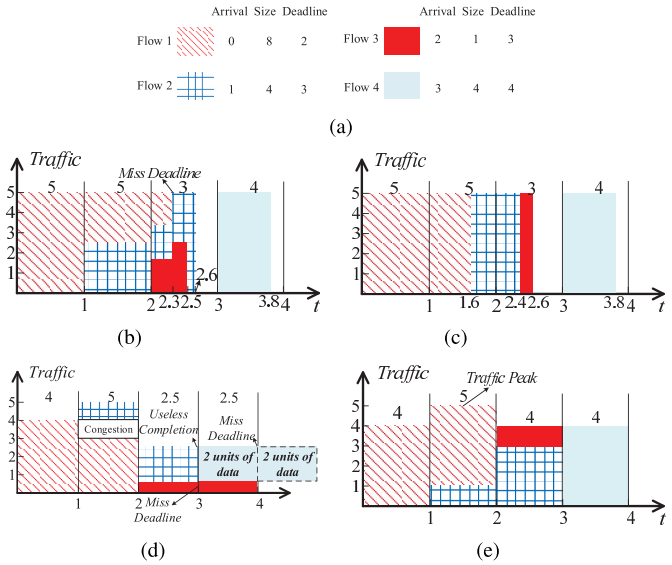


Fig. 2. An illustrative example with one link and four flows whose information is shown in Fig. 2(a). Assuming the 75-*th percentile charging model* is in use, then the billed bandwidth usage incurred by (b) FS is 5; (c) SDF is 5; (d) ES is 4; (e) optimal schedule is 4. Both FS and ES miss some flows' deadlines. Though SDF and optimal schedule accommodates all flows with deadlines guaranteed, SDF incurs a higher transmission cost than the optimal schedule.

Such an inter-DC network model is widely used in practice, and can ease the presentation and analysis due to omitting the routing details inside the ISP networks. Some routing proposals [11], [12] are orthogonal to our methods in this paper, and can further reduce the transmission cost of inter-DC transfers if combined with our work.

B. A Motivating Example

Instead of directly scheduling inter-DC transfers regardless of the *percentile charging model*, an intelligent scheduling should be pricing-aware. As such, the *"free"* time slots in the *percentile charging model* can be efficiently utilized for data transmission, and accordingly the transmission cost can be significantly reduced.

For a better intuition of our problem, we consider an example in Fig. 2, where a link is capable of serving up to 5 units of data in one time slot, while carrying four flows in a period of 4 time slots. As shown in Fig. 2(a), the data sizes for these four flows are $8, 4, 1, 4$, respectively, while the deadlines are the end of the 2nd, 3rd, 3rd, 4th time slot, respectively. Assuming flows arrive at the link at the beginning of each time slot and the 75-th percentile charging model is in use (second largest in this case), the billed bandwidth usages for four scheduling methods are illustrated in Fig. 2(b)-Fig. 2(e). The default fair sharing ensures fairness among concurrent flows in a link, which misses one flow's deadline and incurs high billed bandwidth usage [20]. Shortest-Deadline-First follows the shortest- or smallest-first policy [16], [17]. It is effective in guaranteeing deadlines for the four flows, but results in high billed bandwidth usage. The recently proposed equal splitting method mainly considers that all flows can be delayed by a uniform time D, e.g., one time slot, and splits the arriving flows into $D+1$ sets of equal size [13]. Then each set is served in one time slot. It can reduce the billed bandwidth usage with large values of D, but may lead to high deadline miss rate as well as significant congestion in some time slots, i.e., it causes flow 1 and flow 2 congested at the second time slot and misses the deadlines of both flow 3 and flow 4. Finally, the optimal schedule would make one time slot become the traffic peak, and maintains no traffic differentiation among the other three time slots. The time slot of traffic peak is identified as a *"free"* time slot, and it does not affect the total cost based on the *percentile charging model*. In this case, all flows can be accommodated with guaranteed deadlines, and the transmission cost is reduced to 4.

C. Practical Factors Faced by the Percentile Charging Model

The basic *q-th percentile charging model* encounters three practical factors when it comes to more general cases. *First*, service providers today rely on multiple ISPs to connect their geographically distributed DCs [21]. The per unit bandwidth cost as well as the percentiles to be complied to can vary significantly across different DCs, due to the regional pricing and peering competitions among ISPs [22]. *Second*, ISPs usually measures both the inbound and outbound traffic, calculates the percentile billed bandwidth for each direction, and uses the maximum of these two values [23]. So, both uplink and downlink bandwidth usage should be jointly considered when calculating the transmission cost. *Third*, service provider may contractually commit to an amount of bandwidth regardless of whether the committed bandwidth can be used up [24]. Further, additional cost occurs when percentile billed bandwidth exceeds the committed bandwidth.

III. *TrafficShaper*: System Model and Problem Formulation

In this section, we first describe the system model and then present the problem formulation.

A. System Model

We consider an inter-DC network where a service provider runs its service over a set of DCs, $\mathcal{M} = \{d_1, d_2, \cdots, d_M\}$. For each DC d_j, let U_j^E and U_j^I denote the bandwidth capacities of its uplink l_j^E and downlink l_j^I, respectively. We consider a system that operates in a discrete-time mode, where the time can be divided into T ($T \in \mathbb{N}^+$) time slots. Each time slot t ($= 0, 1, \cdots, T-1$) has a same duration, e.g., 5 minutes. Let S denote the total number of available sessions.

At time t, session s transmits exactly one flow at a rate of $r_s(t)$, with the remaining data size and the time till deadline being denoted as $\lambda_s(t)$ and $\tau_s(t)$, respectively. Actually, such deadline information can be passed to the transport layer [25]. In such case, let $e_s(t) = \lambda_s(t)/\tau_s(t)$ denote the expected rate for session s at t. Note here the session s will be closed once the corresponding flow is finished or past its deadline. Let $x_j^E(t)$ denote the aggregate bandwidth usage on the uplink l_j^E of d_j at t. It is calculated as $x_j^E(t) = \sum_{s \in S(l_j^E)} r_s(t)$, where $S(l_j^E)$ is the set of flows traversing the uplink l_j^E. Similarly, we have $x_j^I(t) = \sum_{s \in S(l_j^I)} r_s(t)$ for the downlink l_j^I of d_j at t. For each d_j, let q_j denote the percentile that it complies to, let c_j and B_j denote the per unit bandwidth cost and the committed bandwidth, respectively. Consider that each charging period consists of N time slots, emerging $K = T/N$ charging periods. At the end of a charging period k $(= 0, 1, \cdots, K-1)$, the percentile billed bandwidth for uplink l_j^E can now be calculated as follows:

$$b_j^E(k) = P_{q_j}(x_j^E(kN), \cdots, x_j^E(kN + N - 1)), \tag{1}$$

where $P_{q_j}(\cdot)$ is a function defined as the $\lceil \frac{100-q_j}{100} N \rceil$-th largest number in the bandwidth usage sequence of a charging period. Similarly, the percentile billed bandwidth for the downlink l_j^I at charging period k is calculated as:

$$b_j^I(k) = P_{q_j}(x_j^I(kN), \cdots, x_j^I(kN + N - 1)). \tag{2}$$

Finally, the actual bandwidth that the service provider needs to pay for its DC d_j during charging period k is defined as

$$b_j(k) = \max\{b_j^E(k), b_j^I(k), B_j\}. \tag{3}$$

Given this definition, we actually need to schedule traffic to *"free"* time slots as much as possible, and simultaneously utilize the committed bandwidth at other time slots with best efforts, so as to follow the simple principle of *"more peak, less differentiation"*.

It is worth noting that even though we mainly focus on the q-th percentile charging method, our model can roughly support other pricing methods. For example, when $q = 100$, our model supports the peak bandwidth pricing that is based on the maximum bandwidth usage over all time slots in a charging period. Furthermore, when $q = 50$, our model can approximately support the average bandwidth pricing model that charges customers based on the average bandwidth usage across all time slots in a charging period.

B. Problem Formulation

We now study the inter-DC transfer scheduling problem that minimizes the transmission cost, and at the same time guarantees deadlines for inter-DC transfers. The problem is stated and formulated as follows.

1) Transmission Cost: For service providers that operate multiple DCs across the world, one of the most important operation costs is the transmission cost for a large amount of inter-DC traffic. Specifically, in charging period k, the transmission cost is the summation of bandwidth cost incurred by each DC, i.e., $\sum_{j=1}^{M} c_j b_j(k)$. In this paper, we define the objective as the *long term average of transmission cost*, which is calculated as the average cost over all charging periods

$$\overline{C} = \lim_{T \to \infty} \frac{1}{\frac{T}{N}} \sum_{k=0}^{\frac{T}{N}-1} \sum_{j=1}^{M} c_j b_j(k). \tag{4}$$

2) Deadline Constraint: Deadline is important for the performance of inter-DC transfers. To complete an inter-DC flow within its deadline, we require the transmission rate to be larger than or equal to the expected rate, $r_s(t) - e_s(t) \geq 0, \forall s, \forall t$. In our formulation, we relax this constraint with its long term time-average:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} (e_s(t) - r_s(t)) \leq 0, \quad \forall s. \tag{5}$$

By incorporating this constraint, we are essentially guarantee that, for every flow that requires $e_s(t)$, the allocated rate $r_s(t)$ is on average larger than $e_s(t)$. It should be noted that this constraint is a relaxation because that realistic flows will not last forever. This relaxation is attractive because of its simplicity in formulating the deadline constraint, yet is effective to guarantee deadlines in practice [26].

3) Link Capacity Constraint: When scheduling the inter-DC transfers, both the uplink and downlink bandwidth capacities of each DC should be satisfied, i.e., $x_j^E(t) \leq U_j^E$ and $x_j^I(t) \leq U_j^I$, $\forall j, \forall t$. Specifically, we impose the long-term averaged aggregate rates with the following inequalities satisfied:

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} x_j^E(t) \leq U_j^E, \quad \forall j, \tag{6}$$

$$\lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} x_j^I(t) \leq U_j^I, \quad \forall j. \tag{7}$$

These constraints mean that the long-term averaged aggregate rates on each DC must not exceed the link capacities. In these constraints, the temporary overloading is allowed due to the buffering in switches or routes [27]. It should be noted that the buffer size (typically 30KB [26]) may not be very large, but the traffic to be buffered in a stable network should be relatively little. Otherwise, the network could be overloaded and no mechanism can avoid packet loss. In such a case, the over-capacity traffic has to wait at the source nodes until there is free bandwidth in the network. In our analysis, we mainly focus on the scenario where the long-term averaged traffic load generated can be handled by the capacity of the network, such that we can have a chance to design a proper flow scheduling scheme to stabilize the network.

4) Decision Variable: For all $s \in \mathcal{S}$ at each time t, the allocated rate $r_s(t)$ should be a non-negative value,

$$r_s(t) \geq 0, \quad \forall s, \forall t. \tag{8}$$

Given the above objective function and constraints, we now can formulate the following stochastic optimization **P1**:

$$\min \ \overline{C} = \lim_{T \to \infty} \frac{1}{\frac{T}{N}} \sum_{k=0}^{\frac{T}{N}-1} \sum_{j=1}^{M} c_j b_j(k)$$

subject to: Eqs. (5)(6)(7)(8).

Problem **P1** is a long-term optimization problem, where the current control decisions are coupled with the future decisions. For example, current decisions on inter-DC flow scheduling may delay excessive flows and hence block the transfer of future flows. To solve such long-term optimization, one may design an offline optimal scheduling algorithm, or leverage the dynamic programming techniques [28]. However, it encounters two challenges when it comes to realistic networks: *(1)* The traffic arrival pattern is usually unknown in advance, yet is difficult to be accurately predicted; *(2)* The percentile function $P_{q_j}(\cdot)$ relies on the bandwidth usage over a large number of time slots in a charging period. These challenges make it infeasible to identify and use the *"free"* time slots, and thereby impractical to reduce or minimize the transmission cost.

IV. *TrafficShaper*: Pricing-Aware Online Control Framework

In response to the challenges of problem **P1**, we take advantage of *Lyapunov optimization* techniques [14], [15] to design an online control framework. In particular, this framework is proved to approach a time-averaged transmission cost that is arbitrarily close to optimum, while still guaranteeing deadlines and system stability.

A. Decomposition Using Lyapunov Optimization

We first provide a brief primer on Lyapunov optimization technique, which has recently received renewed interest in solving resource allocation problems in both wireless and wired networks [15], [29]. Lyapunov optimization typically solves problems in the form of stochastic programming, and the key idea is to decompose a long-term stochastic optimization problem into several sub-problems that can be sequentially solved in each time slot. To this end, it first transforms the long-term constraints into well-studied queue stability problems by introducing a set of actual or virtual queues. It then defines a non-negative function, called a Lyapunov function, to measure the aggregate congestion of all queues in the network. Based on the Lyapunov function, it further defines a Lyapunov drift to indicate the expected change in the Lyapunov function from one time slot to the next. Finally, it adds the Lyapunov drift to the objective of the stochastic programming and minimizes a supremum bound on such drift-plus-objective expression, such that the long-term stochastic optimization problem can be efficiently decomposed into per time slot sub-problems.

While recognizing the significance of Lyapunov optimization, our problem **P1** cannot be directly solved with Lyapunov optimization technique, as the percentile function $P_{q_j}(\cdot)$ imposes tightly coupling among variables of $x_j^E(t)$ or $x_j^I(t)$ over all time slots in a charging period. Thus, we are inspired to first transform problem **P1** to a relaxed optimization problem **P2** that is decomposable. We then decompose the problem **P2** using the *Lyapunov optimization* techniques [14]. Such decomposition follows two steps: *1)* transforming the long-term constraints (Eqs. (5)(6)(7)) into queue stability problems; *2*) constructing the *drift-plus-cost* to divide the relaxed problem **P2** into several sub-problems that can be solved in each time slot. The *drift-plus-cost* can actually characterize the cost-deadline tradeoff.

1) A Relaxed Optimization Problem: To construct a relaxed problem, we consider that all system information including $x_j^E(t)$ and $x_j^I(t)$ cannot be observed till the end of time slot t. In other words, at the end of each time slot t, we only know these information for this time slot t as well as each of the previous $t-1$ time slots. Therefore, we set $x_j^E(t') = 0$ and $x_j^I(t') = 0$ for $t' > t$, and define $\beta_j^E(t) = P_{q_j}(x_j^E(\lfloor t/N \rfloor N), \cdots, x_j^E(t), 0, \cdots, 0)$ to indicate the bandwidth that contributes to the percentile billed bandwidth for uplink l_j^E at time slot t in $\lfloor t/N \rfloor$-th charging period. Specifically, when $t = \lfloor t/N \rfloor N + N - 1$, $\beta_j^E(t)$ is exactly the percentile billed bandwidth in $\lfloor t/N \rfloor$-th charging period. Similarly, we define $\beta_j^I(t) = P_{q_j}(x_j^I(\lfloor t/N \rfloor N), \cdots, x_j^I(t), 0, \cdots, 0)$ for downlink l_j^I. Define $\beta_j(t) = \max\{\beta_j^E(t), \beta_j^I(t), B_j\}$, and then we have the following relaxed problem **P2**:

$$\min_{r_s(t)} \widetilde{C} = \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{j=1}^{M} c_j \beta_j(t) \text{ s.t. Eqs. (5)(6)(7)(8).} \quad (9)$$

Even though problem **P2** and **P1** have different objectives, **P2** is equivalent to the original problem **P1** in terms of the optimal solution, which is shown in the following theorem.

*Theorem 1: The optimal solution of **P2** is the same as that of **P1**, and the objectives of the two problems satisfy* $\sum_{j=1}^{M} c_j B_j \le \widetilde{C} \le \overline{C} \le N\widetilde{C}$.

Proof: Please see the Appendix A in the online supplementary material. ■

2) Queue-Based Constraints: We construct two groups of actual queues and one group of virtual queues. *Firstly*, to accommodate the constraint in Eq. (5), we construct a group of virtual queues $Q_s(t)$ for each $s \in S$. Initially, we define $Q_s(0) = 0, \forall s \in S$, and then update the queues in each time slot as follows:

$$Q_s(t+1) = \max\{Q_s(t) + e_s(t) - r_s(t), 0\}, \quad \forall s. \quad (10)$$

These virtual queues take $e_s(t)$ as input and $r_s(t)$ as output. They stores the difference in the expected transmission rate and actual transmission rate. The queue lengths are essentially historical deviation from expected rates of the inter-DC flows.

Secondly, we construct a group of actual queues $Q_j^E(t)$ for the uplink of each DC. When inter-DC flows arrive, they are actually stored in queue $Q_j^E(t)$ to await be scheduled. The queueing dynamics are then

$$Q_j^E(t+1) = \max\{Q_j^E(t) - U_j^E + x_j^E(t), 0\}, \quad \forall j. \quad (11)$$

For each DC d_j, $Q_j^E(0) = 0$. $x_j^E(t)$ can be viewed as arrivals of queue $Q_j^E(t)$, while the capacity U_j^E can be viewed as the service rate of such a queue. Additionally, for each $d_j \in \mathcal{M}$, we call $Q_j^E(t)$ the backlog at time t, as it represents an amount of output bandwidth that needs to be allocated.

Thirdly, we construct another group of actual queues $Q_j^I(t)$ for the downlink of each DC, with the backlog being empty at time slot 0, i.e., $Q_j^I(0) = 0$.

$$Q_j^I(t+1) = \max\{Q_j^I(t) - U_j^I + x_j^I(t), 0\}, \quad \forall j. \quad (12)$$

These queues can actually accommodate the constraints in Eqs. (5) (6) (7) if they are stable, as proved in the following.

Theorem 2: Equalities (10) (11) (12) *are essentially equivalent to constraints* (5) (6) (7) *if the following stability conditions are satisfied:* $\lim_{T\to\infty} Q_s(T)/T = 0$, $\lim_{T\to\infty} Q_j^E(T)/T = 0$, $\lim_{T\to\infty} Q_j^I(T)/T = 0$.

Proof: Please see the Appendix B in the online supplementary material. ∎

3) Characterizing the Cost-Deadline Tradeoff: Let $\boldsymbol{Q}(t)$ denote the concatenated vector of all virtual and actual queues, $\boldsymbol{Q}(t) = [Q_s(t), Q_j^E(t), Q_j^I(t)]$. Then, we define the Lyapunov function as follows:

$$L(\boldsymbol{Q}(t)) = \frac{1}{2}\left(\sum_{s=1}^{S} Q_s(t)^2 + \sum_{j=1}^{M} Q_j^E(t)^2 + \sum_{j=1}^{M} Q_j^I(t)^2\right). \tag{13}$$

This equality quantitatively reflects the congestion [14] of all queues. On the premise of all queues having strong stability, a small value of $L(\boldsymbol{Q}(t))$ directly implies that queue backlogs are small. To keep the stabilities of queues, the Lyapunov function needs to be persistently pushed towards a lower congestion state. Thus, we are inspired to introduce one-step Lyapunov drift $\Delta(\boldsymbol{Q}(t))$ [14], which is the expected[1] change of queue backlogs over one time slot.

$$\Delta(\boldsymbol{Q}(t)) = \mathbb{E}\left\{L(\boldsymbol{Q}(t+1)) - L(\boldsymbol{Q}(t))|\boldsymbol{Q}(t)\right\}. \tag{14}$$

In addition to stabilize the queue backlog to ensure the constraints on deadlines and link capacities, we also need to consider the transmission cost given by the objective function of the problem **P2**. In this case, the problem **P2** can be approximately solved by minimizing the *drift-plus-cost* in each time slot, which jointly considers the queue backlogs and the incurred transmission cost. Mathematically, we have the following sub-problem **P3** in each time slot t

$$\min\ \Delta(\boldsymbol{Q}(t)) + V\mathbb{E}\{\sum_{j=1}^{M} c_j\beta_j(t)|\boldsymbol{Q}(t)\}, \text{s.t. Eq. (8).} \tag{15}$$

where V is a control parameter that represents an importance weight on how much we emphasize the transmission cost minimization, compared to constraints on deadline guarantees (Eq. (5)) and link capacities (Eqs. (6) (7)). This provides a flexible design choices among various tradeoff points between transmission cost minimization and deadline guarantees. For example, one may prefer to incur as smaller expected transmission cost as possible, while keeping $\Delta(\boldsymbol{Q}(t))$ small to guarantee deadlines as well as ensuring the consumed bandwidth to not exceed the corresponding link capacity.

B. Pricing-Aware Online Control Algorithm (POCA)

It is easy to check that directly minimizing the objective in Eq. (15) involves unknown backlog information $\boldsymbol{Q}(t+1)$.

[1]We assume that the data sizes of all the flows in all time slots are random variables, and these variables are independent and identically distributed (i.i.d). Similarly, the deadlines of all the flows in all time slots are also assumed to be i.i.d. Hence, when we refer to taking expectations of a certain term, e.g., $\boldsymbol{Q}(t)$ and $c_j\beta_j(t)$, this term contains random variables, and thus is random.

Algorithm 1 Pricing-Aware Online Control Algorithm (POCA)

1: In the beginning of each time slot t, update the expected rate $e_s(t)$ for each flow s based on its remaining data size and the remaining time to deadline;
2: Given $e_s(t)$ and current queue backlogs $\boldsymbol{Q}(t)$, determine the control decisions $r_s(t), \forall s \in S$ to minimize the objective $\sum_{j=1}^{M}\mathbb{E}\{Vc_j\beta_j(t) + Q_j^E(t)x_j^E(t) + Q_j^I(t)x_j^I(t)|\boldsymbol{Q}(t)\} - \sum_{s=1}^{S}\mathbb{E}\{Q_s(t)(r_s(t) - e_s(t))|\boldsymbol{Q}(t)\}$ in problem **P4**.
3: Update the queue backlogs $\boldsymbol{Q}(t)$ according to equalities (10) (11) (12) and the newly determined decisions.

We therefore seek to minimize its supremum bound, without undermining the optimality and performance.

1) Bounding Drift-Plus-Cost: A key derivation is to obtain the upper bound for the *drift-plus-cost*, which relies on the following theorem.

Theorem 3: Assume that there exist certain peak levels of expected rates e_{max}, such that $e_s(t) \le e_{max}, \forall s, \forall t$. Then, in each time slot t, given any value of $\boldsymbol{Q}(t)$, the drift-plus-cost can be bounded as follows

$$\begin{aligned}\Delta(\boldsymbol{Q}(t)) + V\mathbb{E}\{\sum_{j=1}^{M} c_j\beta_j(t)|\boldsymbol{Q}(t)\} \le H \\ -\sum_{j=1}^{M}(Q_j^E(t)U_j^E + Q_j^I(t)U_j^I) - \sum_{s=1}^{S}\mathbb{E}\{Q_s(t)(r_s(t) - e_s(t))|\boldsymbol{Q}(t)\} \\ + \sum_{j=1}^{M}\mathbb{E}\{Vc_j\beta_j(t) + Q_j^E(t)x_j^E(t) + Q_j^I(t)x_j^I(t)|\boldsymbol{Q}(t)\}.\end{aligned} \tag{16}$$

where the constant $H \triangleq 2MU_{max}^2 + \frac{1}{2}SU_{max}^2 + \frac{1}{2}Se_{max}^2$, $U_{max} = \max_j\{U_j^I, U_j^E\}$.

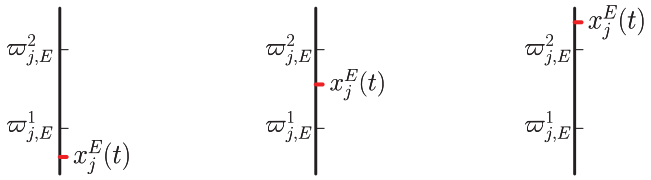
Proof: Please see the Appendix C in the online supplementary material. ∎

Given the theorem above, we are essentially solving the following sub-problem **P4** in each time slot t.

$$\begin{aligned}\min\ &\sum_{j=1}^{M}\mathbb{E}\{Vc_j\beta_j(t) + Q_j^E(t)x_j^E(t) + Q_j^I(t)x_j^I(t)|\boldsymbol{Q}(t)\} \\ &-\sum_{s=1}^{S}\mathbb{E}\{Q_s(t)(r_s(t) - e_s(t))|\boldsymbol{Q}(t)\} \text{ s.t. Eq. (8).}\end{aligned} \tag{17}$$

With the above problem **P4**, we now can design our **POCA** algorithm, as shown in **Algorithm 1**. In each time slot t, based on the *online* observation of the queue backlogs $\boldsymbol{Q}(t)$ and the updated expected rate $e_s(t), \forall s$, **POCA** strives to solve the problem **P4** for the purpose of determining the transmission rates for all inter-DC flows in time slot t. Finally, **POCA** updates the queue backlogs $Q_s(t)$, $Q_j^E(t)$, $Q_j^I(t)$ according to equalities (10) (11) (12) and the newly determined transmission rates.

To solve problem **P4**, we need to first know the form of the term $\beta_j(t) = \max\{\beta_j^E(t), \beta_j^I(t), B_j\}$. Since **P4** is solved in each time slot t with knowing all the past (historical) information, $\beta_j^E(t)$ can then be expressed as



(a) If $x^E_j(t)<\varpi^1_{j,E}$, then $\beta^E_j(t)=\varpi^1_{j,E}$. (b) If $\varpi^1_{j,E}\le x^E_j(t)\le\varpi^2_{j,E}$, then $\beta^E_j(t)=x^E_j(t)$. (c) If $x^E_j(t)>\varpi^2_{j,E}$, then $\beta^E_j(t)=\varpi^2_{j,E}$.

Fig. 3. Let $\varpi^1_{j,E}$ and $\varpi^2_{j,E}$ denote the $\lceil\frac{100-q_j}{100}N\rceil$-th largest number and the $\lceil\frac{100-q_j}{100}N-1\rceil$-th largest number respectively, in the sequence $(x^E_j(\lfloor t/N\rfloor N),\cdots,x^E_j(t-1),0,\cdots,0)$. Then, the $\lceil\frac{100-q_j}{100}N\rceil$-th largest number in the sequence $(x^E_j(\lfloor t/N\rfloor N),\cdots,x^E_j(t),0,\cdots,0)$ can be calculated as $\beta^E_j(t)=\max(\min(\varpi^2_{j,E},x^E_j(t)),\varpi^1_{j,E})$.

$\beta^E_j(t)=\max(\min(\varpi^2_{j,E},x^E_j(t)),\varpi^1_{j,E})$, as shown in Fig. 3. Note that both $\varpi^1_{j,E}$ and $\varpi^2_{j,E}$ are constants, because the sequence $(x^E_j(\lfloor t/N\rfloor N),\cdots,x^E_j(t-1),0,\cdots,0)$ is already known in time slot t. Similarly, we can calculate $\beta^I_j(t)$ as $\max(\min(\varpi^2_{j,I},x^I_j(t)),\varpi^1_{j,I})$. With the expressions of $\beta^E_j(t)$ and $\beta^I_j(t)$, we can easily check that **P4** is actually a linear programming problem and can be solved with classical linear programming approaches, e.g., simplex and interior point methods [30]. This implies that our **POCA** is computationally efficient, as it only needs to solve a much smaller scale LP problem **P4**. Furthermore, **P4** can also be solved in a decentralized fashion, which will be shown in Section IV-C.

2) Performance Analysis: We now analyze the optimality of **POCA** algorithm, in terms of a well-balanced tradeoff between transmission cost minimization and deadline guarantees.

*Theorem 4: For any $V>0$, the **POCA** algorithm can achieve the following performance guarantee:*

$$\lim_{T\to\infty}\frac{1}{T}\mathbb{E}\{\sum_{t=0}^{T-1}\sum_{j=1}^{M}c_j\beta_j(t)\}\le\frac{H_1}{V}+\widetilde{C}_*, \tag{18}$$

$$\overline{Q}\le\frac{H_1+V\widetilde{C}_*}{\epsilon}, \tag{19}$$

*where $H_1=H+\frac{5}{2}M(N-1)U^2_{max}+\frac{1}{2}M(N-1)e^2_{max}$ and $\widetilde{C}_*$ is the optimal averaged transmission cost (equivalent to the minimum value in problem **P2**), $\epsilon>0$ and $\overline{Q}$ is the time-averaged queue length, given as*

$$\overline{Q}\triangleq\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}\sum_{j=1}^{M}\mathbb{E}\{Q^E_j(kN)+Q^I_j(kN)\}+\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}\sum_{s=1}^{S}\mathbb{E}\{Q_s(kN)\}$$

Proof: Please see the Appendix D in the online supplementary material. ■

Insights: This theorem demonstrates an $O(1/V)$ gap between the objective $\widetilde{C}=\lim_{T\to\infty}\frac{1}{T}\mathbb{E}\{\sum_{t=0}^{T-1}\sum_{j=1}^{M}c_j\beta_j(t)\}$ and the optimum $\widetilde{C}_*$ of problem **P2**. Specifically, by using an arbitrarily larger V, we can make $\widetilde{C}$ arbitrarily close to the optimum, while maintaining the virtual queues are stable. As such, the time-averaged transmission cost $\overline{C}$ defined in **P1** can, in turn, be bounded by using **POCA** algorithm, because that $\overline{C}\le N\widetilde{C}$ (proved in **Theorem** 1). Note that the transmission cost reduction may be achieved at the cost of a larger deadline miss rate, as Eq. (19) implies that the time-averaged queue backlogs grows linearly with parameter V. However, if the total capacity is insufficient to serve all transfer requests, **POCA** will strive to reduce the transmission cost as much as possible while serving all transfer requests. Furthermore, **POCA** can also be flexibly extended to enforce a budget by denying an appropriate amount of transfer requests [31].

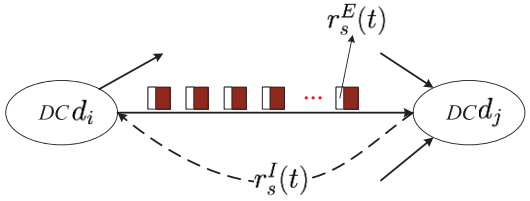


Fig. 4. An illustrative example of the decentralized implementation of the POCA algorithm.

C. Decentralized Implementation of POCA

An alternative way to implement the POCA algorithm is to leverage a centralized controller across all DCs [1], known as "centralized way". More precisely, such centralized way uses the controller to periodically collect the global information of all DCs (i.e., U^E_j and U^E_j) as well as the associated parameters (i.e., $Q^E_j(t)$, $Q^I_j(t)$ and $Q_s(t)$). It then solves the problem **P4** to derive the decisions on how much rate should be allocated to each inter-DC transfer. Finally, the decisions are forwarded to the underlying network devices (perhaps SDN switches) with the purpose of enforcing the bandwidth allocated to each inter-DC flows via standardized SDN functions (i.e., the MeterTable in OpenFlow).

The centralized way may be efficient; however, it inevitably faces some practical challenges. First, it involves high computation complexity when solving the problem **P4**. Compared to the long-term original problem **P1**, problem **P4** may have a smaller scale. However, it still has high computational complexity. Furthermore, such complexity significantly increases as the number of DCs and inter-DC flows grows, as it has $|\mathcal{M}|\times|\mathcal{S}|$ variables. In fact, the number of DCs is around $O(10^2)$ and the number of inter-DC flows is around $O(10^5)$ in some production clouds [3], [32]. Second, the centralized way causes high delay between the time that a time slot begins and the time that rate decision really works. The root reason is that it involves substantial communication overhead to collect the parameters needed for the POCA algorithm and to forward the decisions to underlying network as well.

In response to the above challenges, we seek to implement our POCA algorithm in a decentralized way. Such decentralized implementation is shown in Fig. 4. First, in the beginning of each time slot t, each DC $d_i\in\mathcal{M}$ collects the set $S(l^E_i)$ of flows on its uplink l^E_i, and then performs a locally computation to derive a sending rate for each flow issued from it ($r^E_s,\forall s\in S(l^E_i)$). Specifically, this involves the following

sub-problem **P5**:

$$\begin{aligned}\min\ & \mathbb{E}\{Vc_i \max(\beta_i^E(t), B_i) + Q_i^E(t)x_i^E(t)|\boldsymbol{Q}(t)\} \\ & - \sum_{s\in S(l_i^E)} \mathbb{E}\{Q_s(t)(r_s(t) - e_s(t))|\boldsymbol{Q}(t)\} \\ \text{s.t. } & r_s(t) \geq 0, \quad \forall s \in S(l_i^E). \end{aligned} \tag{20}$$

The sub-problem **P5** is of much smaller scale than problem **P4**, with only $|S(l_i^E)|$ variables. For each $s \in S(l_i^E)$, once the decision $r_s^E(t)$ is derived, the DC d_i then marks the value of sending rate in the ToS field of each packet belonging to the flow. Note that the sub-problem **P5** can be solved in a way similar to that solves problem **P4**.

On the other hand, each DC $d_j \in \mathcal{M}$ collects the set $S(l_j^I)$ of flows on its downlink l_j^I, by counting the number of TCP connections it established. Then, it computes a receiving rate $r_s^I(t)$ for each flow $s \in S(l_j^I)$, by solving the following sub-problem **P6**.

$$\begin{aligned}\min\ & \mathbb{E}\{Vc_j \max(\beta_j^I(t), B_j) + Q_j^I(t)x_j^I(t)|\boldsymbol{Q}(t)\} \\ & - \sum_{s\in S(l_j^I)} \mathbb{E}\{Q_s(t)(r_s(t) - e_s(t))|\boldsymbol{Q}(t)\} \\ \text{s.t. } & r_s(t) \geq 0, \quad \forall s \in S(l_j^I). \end{aligned} \tag{21}$$

Upon obtaining the receiving rate $r_s^I(t)$, it is the need to feedback the value $r_s^I(t)$ to the source DC associated with s. As such, the source DC can change the sending rate of flow s to the minimum value between $r_s^E(t)$ and $r_s^I(t)$. Note that we perform such feedback action in a laissez-fair manner. More precisely, the DC d_j extracts the value of $r_s^E(t)$ from its received packets, and conducts comparison between this value and the corresponding $r_s^I(t)$. The DC d_j will feed back the receiving rate to the source DC, only if $r_s^I(t)$ is smaller than $r_s^E(t)$ for a certain flow s. Otherwise, it will not trigger the feedback action. This directly reduces the communication overhead when implementing the POCA algorithm. On receiving a feedback for a flow s, a source DC will change the sending rate of the flow to $\min(r_s^E(t), r_s^I(t))$.

Finally, each DC updates the queue backlogs $\boldsymbol{Q}(t)$ based on Eqs. (10) (11) (12) and the decisions $r_s(t) \forall s \in S$.

V. *TrafficShaper*: Performance Evaluation

In this section, we evaluate the performance of our proposed *TrafficShaper* scheduler through large-scale simulations as well as small-scale testbed implementation.

A. Large-Scale Trace-Driven Simulation

Simulation setup: We simulate an inter-DC network with 40 DCs, which is a common network size in typical service companies, e.g., Microsoft [2]. In this 40-DC setup, we vary the link bandwidth from 1Gbps to 2Gbps, hoping to mimic the heterogeneous bandwidths across different DCs.

Datasets: Our experiments are conducted on Yahoo! network flow datasets [33]. These datasets, collected from the border routers of five major Yahoo! DCs during a period of 24-hour, contain not only traffic between Yahoo! servers and clients, but also traffic across different Yahoo! DCs.

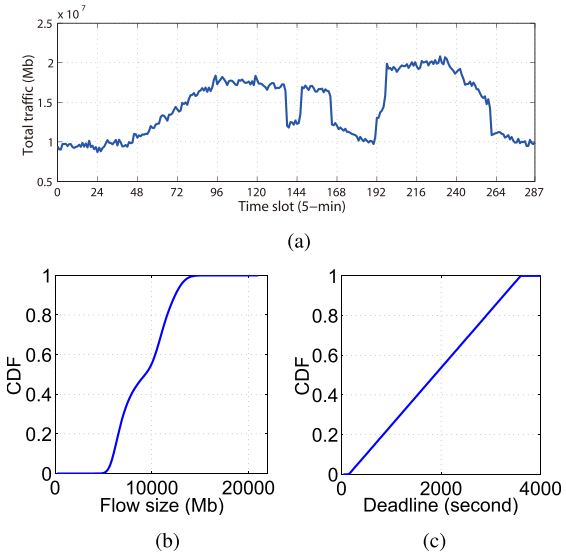


Fig. 5. Total inter-DC traffic extracted from the Yahoo network datasets. (a) Total traffic. (b) CDF of per flow size. (c) CDF of per flow deadline.

Each line in these datasets includes the following fields: 1) timestamp, 2) source and destination IP address, 3) source and destination port, 4) protocol, 5) number of packets and bytes transferred from the source to the destination. All IP addresses in these datasets are permuted to hide the identities of Yahoo! application providers. Hence, we extract the inter-DC traffic based on the study of [3], and use 10% of them to represent the inter-DC transfers. We scale the extracted traffic to our simulated inter-DC network, and set each charging period to be 2 hours. Each time slot is 5 minutes. Each transfer's deadline is set to be a random value within $[0, 12]$ time slots, so as to construct different time-sensitivities for inter-DC transfers. Fig. 5(a) plots the total inter-DC traffic every 5-minute for 24 hours. To have complete understanding of the datasets, we also show the CDFs of per flow size and per flow deadline in Fig. 5(b) and Fig. 5(c), respectively. From these two figures, we observe that the flows are short relative to their deadlines. This provides more flexibility in scheduling them to minimize the transmission cost.

Parameters settings: The per unit bandwidth price is determined from a tiered structure based on the link capacity, where a DC with larger link capacity has a lower cost. To determine the exact per unit bandwidth price, we use Amazon EC2 data transfer prices [34]. For instance, if the transfer price is 0.0012$/Mb, then the per unit bandwidth price is calculated as the multiplier of this transfer price and the time-length of a complete charging period. Similarly, we consider a tiered structure of percentiles, where a DC with lower per unit bandwidth price has a higher billing percentile. TABLE I summarizes such tiered bandwidth prices and percentiles. Without loss generality, each DC is equipped with a same committed bandwidth (e.g., $B_j = B, \forall j$).

Compared methods: We compare *TrafficShaper* with the following four scheduling methods:

- **SDF** (Shortest-Deadline-First): SDF [16], [17] method prioritizes flows based on their deadlines, and schedules a flow with smallest deadline each time.

TABLE I
TIERED BANDWIDTH PRICES AND PERCENTILES

Bandwidth Capacity (Mbps)	Price ($/Mbps)	Percentile
< 1200	$0.0012\times7200=8.64$	75
$1200-1400$	$0.0009\times7200=6.48$	80
$1400-1600$	$0.0007\times7200=5.04$	85
$1600-1800$	$0.0005\times7200=3.60$	90
> 1800	$0.0003\times7200=2.16$	95

- **ES** (Equal Splitting): ES [13] method divides each arriving flow into a uniform (i.e., $D+1$) sets of equal size in each time slot, and then serves each set in this time slot as well as each of the following D time slots. Specifically, we consider that all flows can be delayed by 2 time slots for the ES method (i.e., D=2), as this is a convincible setting for the performance of ES method [13].
- **Heuristic1**: This heuristic method first seeks to schedule flows in time slots that currently are traffic peaks, and then balances the traffic scheduled in other time slots.
- **Heuristic2**: Different from Heuristic1, this heuristic method balances the bandwidth consumption among all time slots when scheduling flows.

Impact of control parameter V: From Theorem 4, we note that the performance of *TrafficShaper* depends on the control parameter V, which indicates how much we emphasize the transmission cost minimization. To evaluate the impact of control parameter V on both the time-averaged cost and time-averaged queue backlog during a period of 24-hour, we fix B to be 500Mbps, and vary V from 10^{-4} to 10^{10}. As shown in Fig. 6(a), the time-averaged cost achieved by *TrafficShaper* decreases as the value of V increases, and can converge to a stable and low value with larger V. In contrast, the time-averaged backlog increases as V increases. One way wonder at this point that the time-averaged backlog is relatively stable at the beginning and around the equilibrium (maximum value). The root reason is that when V is small, the queue backlog dominates the minimization of Eq. (15). On the other hand, when V is large, the transmission cost takes the dominate place. In each of the above two cases, the time-averaged queue backlog will be maintained at a stable value. Recall that the time-averaged queue backlog could be caused by missing the deadlines of flows or overloading the uplink or downlink capacities of DCs. However, in our simulations, the queue backlog is dominated by backlog in the virtual queues $Q_s(t)$. This implies that a higher time-averaged queue backlog can cause more inter-DC transfers to get a data rate lower than the corresponding expected data rate, leading to a higher deadline miss rate (we will show this point in Fig. 10(a)). From Fig. 6(a), we observe that by choosing an appropriate value of V, e.g., $V = 10^6$, *TrafficShaper* can achieve a significantly lower transmission cost while guaranteeing an acceptable queue backlog as well as deadline miss rate.

Impact of committed bandwidth B: As aforementioned, a service provider will contractually commit to an amount of bandwidth B. Different values of B can lead to different cost of the service provider. To evaluate the impact of committed bandwidth, we fix V to be 10^6, and plot the time-averaged

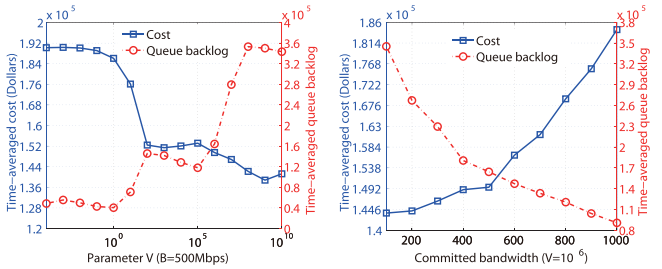


Fig. 6. Impact of parameter V, committed bandwidth B and percentile q on the performance of the proposed POCA algorithm. (a) Impact of V. (b) Impact of B.

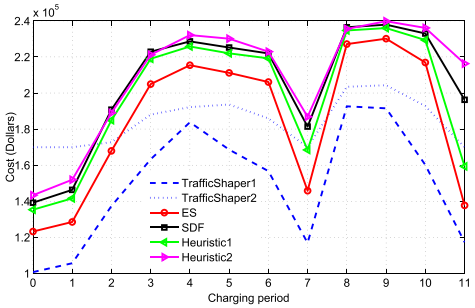


Fig. 7. Total transmission cost across different charging period.

cost and time-averaged queue backlog with different values of B in Fig. 6(b). Clearly, the time-averaged cost increases significantly while the time-averaged queue backlog decreases significantly, as the value of B increases. The root reason is that a higher committed bandwidth results in a higher transmission cost, but incurs less deadline miss rate (which will be shown later). Note that one can flexibly trade off the transmission cost and deadlines by choosing an appropriate value of B (e.g., $B = 500$Mbps). To make the figures easy to read, we mainly show the simulation results in the following two scenarios for *TrafficShaper*: 1) $V = 10^6, B = 500$Mbps; 2) $V = 10^6, B = 1000$Mbps.

Transmission cost: We now present the main performance metric, the total transmission cost across all DCs and all charging periods. As we can see in Fig. 7, *TrafficShaper1* is substantially more cost-effective than all the other methods, because it can efficiently exploit the *"free"* time slots in the q-th percentile charging model. On the other hand, *TrafficShaper2* cannot bring much benefit in reducing the transmission cost. This is because that *TrafficShaper2* is configured with a relatively high value of committed bandwidth B. It is worth noting that Heuristic1 makes some efforts on exploiting the *"free"* time slots in the q-th percentile charging model, since it incurs less transmission cost than Heuristic2 which purely balances traffic across all time slots. Even so, the time slots it heuristically found for each flow are essentially the current traffic peaks and may not be viewed as "free" when coming to the end of a certain charging period. We can further observe that our *TrafficShaper* can reduce the transmission cost by up to 40.23%, 26.19%, 30.63% and 45.78%, compared to

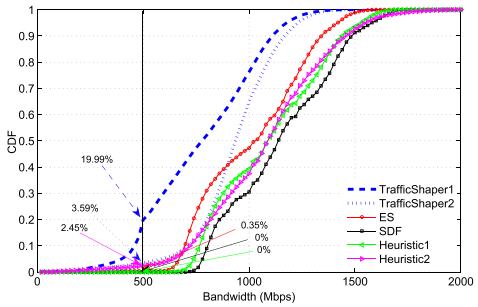


Fig. 8. CDF of bandwidth consumption across all time slots and all DCs.

SDF, ES, Heuristic1 and Heuristic2, respectively. Moreover, the average cost reductions can also reach 27.46%, 18.86%, 24.78% and 28.74%. These results directly demonstrate that our *TrafficShaper* is efficient in reducing the transmission cost for a large amount of inter-DC transfers.

Bandwidth consumption: To have a comprehensive understanding of the transmission cost, we record the bandwidth consumption across all DCs and all time slots for all methods. Note that when recording these bandwidth consumptions, we only use the traffic that is successfully transferred before its corresponding deadline. In other words, once a flow is past its deadline, we will discard this flow immediately for all methods, and the traffic after the deadline will not be transferred accordingly. Bearing this point in mind, we plot the CDF of bandwidth consumption in Fig. 8. As we can see, the fraction of bandwidth consumptions that are lower than 500Mbps can be up to 19.99% for *TrafficShaper*, while those values are only 0.35%, 0%, 0% and 2.45% for ES, SDF, Heuristic1 and Heuristic2 methods, respectively. Furthermore, all bandwidth consumptions for *TrafficShaper1*, *TrafficShaper2*, ES, SDF, Heuristic1 and Heuristic2 are lower than 1480Mbps, 1540Mbps, 1620Mbps, 1780Mbps, 1760Mbps, 1996Mbps, respectively. The observant readers may notice that why different methods involve different bandwidth consumption on average, since they need to send a same amount of total traffic. The reasons are two folds: 1) different methods will cause different traffic distribution across all time slots and all DCs; 2) different methods will drop different amounts of traffic that is past its corresponding deadline, resulting in different deadline miss rates (we will show this point later on).

Average gap between consumed bandwidth and billed bandwidth: To characterize the *"more peak, less differentiation"* feature of *TrafficShaper*, we define a metric — *the gap between consumed bandwidth and billed bandwidth*. For each time slot t and each DC d_j, such gap is defined as $x_j^I(t) - b_j^I(k)$ for downlink or $x_j^E(t) - b_j^E(k)$ for uplink. With such definition, we can quantitatively describe the bandwidth usage on *"free"* time slots when such gap is greater than 0, as well as describe such usage on other time slots when such gap is less than 0. Fig. 9(a) and Fig. 9(b) present the average gap across all DCs at two sampled charging periods, 7 and 10, respectively. Charging periods 7 and 10 correspond to the cases of a low and a high traffic demand, respectively. As we can see, at both sampled charging periods, *TrafficShaper1* maintains higher value of average gap in the beginning three time slots, and then approaches a relatively stable average gap in other time slots,

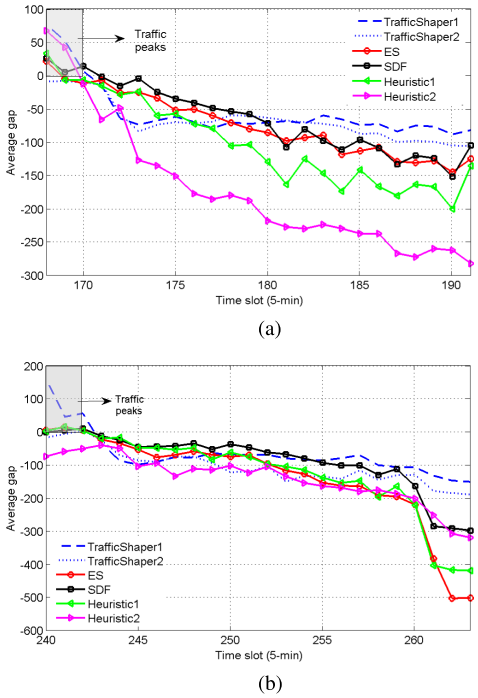


Fig. 9. Average gap between bandwidth consumption and billed bandwidth over all DCs at two sampled charging periods. (a) Charging period 7. (b) Charging period 10.

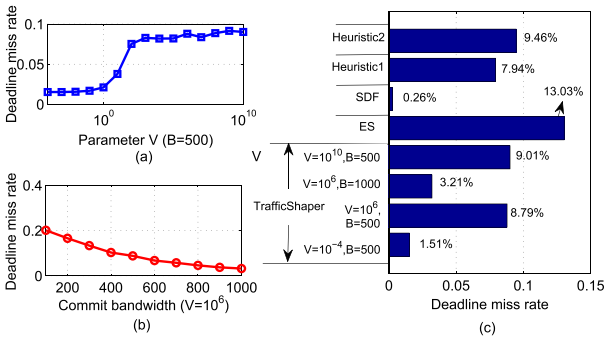


Fig. 10. An overview of the deadline miss rate. (a) Deadline miss rate with varying V for TrafficShaper. (b) Deadline miss rate with varying B for TrafficShaper. (c) Deadline miss rate for different methods.

compared to the ES, SDF, Heuristic1 and Heuristic2 methods. This implies that *TrafficShaper* can schedule more traffic in *"free"* time slots and utilize the billed bandwidth as much as possible, with appropriate values of V and B. One may question why Heuristic1 cannot schedule more traffic in the *"free"* time slots, since it schedules flows in time slots that are traffic peak times. The reason is that the peaks it found only reflect current traffic condition, and may not appear to be traffic peaks later on. These results quantitatively confirm the *"more peak, less differentiation"* feature of *TrafficShaper*.

Deadline miss rate: As most inter-DC transfers have different time-sensitivities, we conduct quantitative analysis on the deadline miss rate. Fig. 10(a) first shows the deadline miss rate with different values of the control parameter V. It is clear that the deadline miss rate for *TrafficShaper* increases as the value of V increases, which indirectly demonstrates the trend of queue backlog in Fig. 6(a). An interesting observation is that the curve in Fig. 10(a) approximately follows a sigmoidal

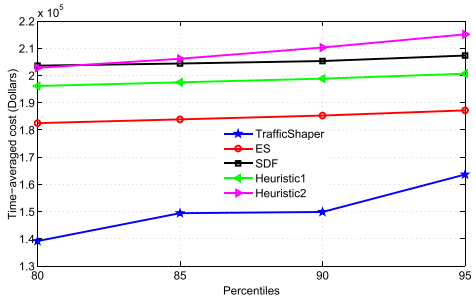


Fig. 11. Time-averaged cost with varying values of percentiles q.

trend with gradual change rate at the beginning and around the stable value. The root reason is that when V is relatively small or large, the change of the queue backlogs does not have much impact on the minimal value of the objective function in Eq. (15). While the queue backlogs are dominated by the virtual queues $Q_s(t)$ which are closely related to the deadline miss rate. We proceed to depict the deadline miss rate with varying committed bandwidth B in Fig. 10(b). We can easily check that the deadline miss rate decreases significantly as the value of B increases. Fig. 10(c) further shows the deadline miss rate for different methods. We make the following observations from this figure: (1) SDF performs best while ES performs worst in guaranteeing the deadlines of inter-DC flows; (2) Heuristc1, Heuristic2 and *TrafficShaper* can achieve a deadline miss rate between that of SDF and that of ES; (3) *TrafficShaper* can achieve an acceptable deadline miss rate by choosing appropriate values of parameters V and B. More specifically, when $V = 10^{-4}$ and $B = 500$Mbps, *TrafficShaper* accommodates 11.5% more transfers with deadlines guaranteed than ES, with incurring only 1.27% more deadline miss rate than SDF. Thus, this gives a flexible design for the trade-off point between the objectives of minimizing transmission cost and guaranteeing deadlines.

Impact of q on the transmission cost: In q-th percentile charging model, the percentile q directly determines the number of "free" time slots, and accordingly impacts the transmission cost. To evaluate the impact of q, we fix $V = 10^6$, $B = 500$Mbps and consider that all DCs comply to a same percentile (i.e., $q_j = q, \forall j$). Fig. 11 shows the time-averaged transmission cost with varying values of q for all methods. We can easily observe that the time-averaged cost achieved by each method increases with the increasing of q. This is mainly because that a higher value of q immediately leads to a smaller number of "free" time slots in the percentile charging model, leaving little space for scheduling to take effect for reducing the transmission cost. We can further observe that our *TrafficShaper* can always maintain a lower time-averaged cost, compared to the other four methods. Moreover, the cost reduction ratio achieved by *TrafficShaper* is highest when $q = 80$ (i.e., 31.65%) and is lowest when $q = 95$ (i.e., 23.75%). This implies that the smaller value of q, the more benefit our algorithm can have. One may wonder at this point that why *TrafficShaper* achieves very similar time-averaged cost when $q = 85$ and $q = 90$. This is because that the committed bandwidth B equals to 500Mbps, while both the $\lceil \frac{100-85}{100} \times 24 \rceil$-th and the $\lceil \frac{100-90}{100} \times 24 \rceil$-th largest numbers in the bandwidth usage sequences of each charging period are around 500.

TABLE II
TrafficShaper vs. DEFAULT FAIR SHARING

Method	Average	
	Transmission cost ($)	Num. of Failed Flows
TrafficShaper	3.91×10^4	4.8
Default Fair Sharing	4.85×10^4	8

As aforementioned, temporary overloading may happen, due to the long-term average capacity constraints. Hence, we have also evaluated how often can such temporary overloading happen, and what buffer sizes would in general be necessary. Please find the detailed results in the Appendix E in the online supplementary material.

B. Small-Scale Testbed Implementation

We build a small testbed with 8 servers to emulate an inter-DC network, with each server representing a DC. All servers are connected to a Pica8 3297 48-port Gigabit switch with an 1Gbps link. Each server has installed Ubuntu, 12.04 64bit version system, and has a 2-core Intel(R) Pentium(R) 3.00GHz CPU, 2GB of RAM, and 1G Ethernet NICs.

With such emulated inter-DC network, we perform distributed per-flow rate limiting on end hosts. At the beginning of each time slot, each end host intercepts all outgoing packets, computes the sending rate for each flow, and marks the sending rate as well as expected rate in the header of each packet. The modified packets are then delivered to Linux Traffic Control (TC) for rate limiting. Like the study of [35], we use two-level Hierarchical Token Bucket in TC to implement rate limit. That is, the root nodes classifies all packets of each flow to a leaf node, and the leaf nodes enforce per-flow rates. From another point of view, the corresponding destination will similarly compute a receiving rate for each flow. Moreover, it will send a feedback to source end host once the receiving rate is smaller than sending rate, such that the source end host can reduce the sending rate to the feedback value.

We conduct our experiments 10 rounds. In each round, it runs one hour, with each time slot being one minute. Following an all-to-all communication pattern, we generate more than 300GB of traffic with 55 flows. Each flow's deadline is set to be a random value within $(0, 60)$ minutes. Similarly, we set the committed bandwidth to be 500Mbps, and use 95-th percentile pricing charging model, for each link. The baseline is the default fair sharing method.

Table II first lists the average transmission cost and the average number of failed flows across all DCs and all experiment times. We can easily check that our *TrafficShaper* can reduce the transmission cost by 19.38% on average, while accommodating more flows with deadlines guaranteed, compared to the default fair sharing. It should be noted that due to the dynamic nature of network, each round of experiment may lead to different number of failed flows. This is why the average number of failed flows may come out to be a decimal. To have a comprehensive understanding of the transmission cost, we plot the average bandwidth consumption across all DCs in Fig. 12. It is clear that *TrafficShaper*

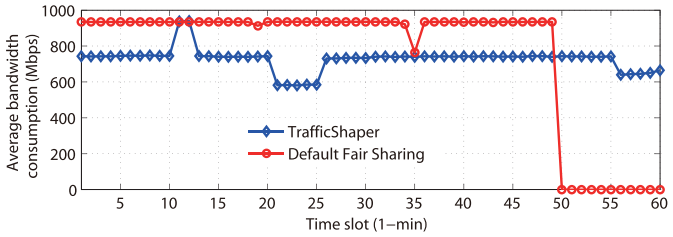


Fig. 12. Average bandwidth consumption.

achieves a lower bandwidth consumption at most of the time, compared to default fair sharing. Furthermore, *TrafficShaper* can schedule two time slots (e.g., time slot 10 and 11) as traffic peaks, and maintain less traffic differentiation among other time slots. Under the 95-th percentile charging model, the billed bandwidths for *TrafficShaper* and default fair sharing are 747.49Mbps and 935.13Mbps, respectively. These results demonstrate that *TrafficShaper* can practically reduce the transmission cost for inter-DC traffic, while provide acceptable deadline guarantees.

VI. Discussion

Why percentile-based charging models are popular? *First*, from the perspectives of ISPs, the q-th percentile pricing is the most prevalent method that transit ISPs use for charging their customers [6]–[9]. This method is simple to implement and uses data (e.g., SNMP) that an ISP typically already collects. *Second*, from the perspectives of customers, the most attractive property in the q-th percentile charging model is that it allows a customer to burst beyond its committed bandwidth, and a few bursts can be carried *free of charge*. This can be beneficial to customers whose peak bandwidth usage is limited to less than $q\%$ (e.g., 5%) of the time every month, e.g., customers that utilize more bandwidth during mornings and nights [11]. Moreover, when customers utilize our *TrafficShaper* scheduler to shape their traffic, the transmission cost could be significantly reduced.

What promise can ISPs provide to their customers under the q-th percentile model? The committed bandwidth is a guarantee in the q-th percentile charging model. Once the committed bandwidth is contractually negotiated, service providers (or customers) will have to pay for it, even if it cannot be used up. On the other hand, it will be regarded as a breach of contract if an ISP fails to guarantee the committed bandwidth, and accordingly an ISP needs to pay a penalty to the service provider. As for missing deadlines of service provider's flows, no penalty will be incurred for the ISP because the q-th pricing model does not cover latency.

Determining optimal value of the control parameter V. Finding the optimal value of the control parameter V remains an open problem. A simple approach to determine the control parameter V would be to leverage the historical traffic matrix to run our POCA algorithm multiple rounds, with each round using different values of V. To be more practical, one can also dynamically change the value of V based on online learning method. We leave this point as one direction of future work.

Handling interactive traffic. The relaxation of inter-DC flows' deadlines, i.e., Eq. (5), makes *TrafficShaper* less efficient for interactive traffic with tight deadlines; however, interactive flows only account for around 10% of the inter-DC traffic [3]. One can use a simple approach to statically divide the bandwidth into two parts: one part of bandwidth is used for interactive traffic while another part can be used by our *TrafficShaper* scheduler. To support interactive traffic in *TrafficShaper*, we need to add new constraints in our model and design new algorithms. We leave it as another direction of future work.

How to optimize the committed bandwidth B? One may wonder at this point that the performance of *TrafficShaper* is correlated with the committed bandwidth B. In this case, how to optimize the committed bandwidth is also an important problem. Actually, one can formulate a new optimization problem, where the objective is to minimize the amount of committed bandwidth and the constraints are three-fold: 1) the link capacity constraint; 2) the deadline constraint; 3) the constraint of limiting the percentile billed bandwidth less than or equal to the committed bandwidth. Then, one can design new algorithms or still exploit the advantages of Lyapunov optimization techniques to solve the problem. We leave this point as an open issue.

Determining the optimal percentile. Determining the optimal percentile to maximize both the utilities of the ISP and the customer (e.g., DC provider) is out of the scope of this work and can be difficult as the utilities of the ISP and the customer are conflicting with each other. But then again, one can design some heuristics. For instance, one can design a scheme that assigns lower percentile to customers whose peak traffic does not contribute significantly to the ISP's cost, and higher percentiles to customers that contribute most to the ISP's cost. Such scheme would retain the attractive properties of the percentile-based charging model, while better accounting for a customer's contribution to ISP's cost.

VII. Related Work

Inter-DC traffic optimization has become an active research topic recently. However, none of the existing work can directly solve the problem proposed in this paper. Regarding the traffic engineering in inter-DC networks, B4 [1] and SWAN [2], designed by Google and Microsoft, respectively, focus on improving the utilization of inter-DC WAN based on the popular software-defined networking technology. Unfortunately, they both are deadline-agnostic, yet are unaware of the percentile pricing model because of the private WANs they owned.

Regarding the transmission cost on inter-DC traffic, Laoutaris *et al.* [11] present NetStitcher, which uses a store-and-forward approach to schedule inter-DC bulk transfers with the aim of fully utilizing the remaining bandwidth. Similar to NetStitcher, Feng *et al.* present Jetway [12], which conservatively utilize remaining bandwidth for multiple inter-DC video flows. However, they both are incapable of accumulating traffic into *"free"* time slots of the q-th percentile charging model. Moreover, neither of them considers the traffic deadline. However, neither of them is insufficient to minimize the transmission cost incurred by the inter-DC transfers. Most recently, Jalaparti *et al.* [8] present a framework called Pretium, which combines dynamic pricing with wide-area

network traffic engineering for ensuring the service guarantees and maintaining low costs of the platform.

Regarding the deadlines of inter-DC Kandula *et al.* present Tempus [10], which aims to maximize the fraction of transfer delivered before deadline. It achieves fairness among all transfer requests, but does not guarantee the completion of any of them. By taking one step further, Zhang *et al.* propose Amoeba [35], which allows users to explicitly specify the amount of data and deadline. Jia *et al.* [36] propose approximate algorithms to schedule transfers over optical WANs, with the goal of minimizing the makespan for finishing all transfers; however, they don't provide deadline guarantee on any transfer.

In the context of Internet, there are some studies focused on designing appropriate pricing method for benefiting both ISPs and uses. For example, TUBE [37] focuses on mobile data pricing; it moves delay-tolerant users out of the peak traffic periods by using time-dependent dynamic pricing. Zhang *et al.* uses a similar idea for pricing wireless data [38]. Our work is different as we advise to send more traffic in "free" time slots of percentile pricing model, and construct more peak traffic periods. Valancius *et al.* [22] propose a destination-based tiered pricing for transit ISPs by using the traffic demand and the cost of carrying it. In terms of investigating the impact of pricing model on traffic optimization, Laoutaris *et al.* propose to use already-paid-for off-peak bandwidth for the delay-tolerant bulk data, and design a source scheduling policy and a store-and-forward policy [7]. Unfortunately, their methods inevitably rely on prior knowledge of the traffic arrivals, and ignore deadlines for the inter-DC transfers. More recently, Golubchik *et al.* study the 95-th percentile minimization problem for scheduling data transfers over the Internet, with constraints on delay requirements [13]. They present both offline and online algorithms to solve this problem. Nevertheless, they assume uniform deadline requirements for all traffic, and care only about the single-sender percentile minimization.

In addition, interest has been growing in datacenter networks using Lyapunov optimization. For example, several works have been proposed to leverage Lyapunov optimization techniques [14], [29] to design optimal power or energy management in datacenter networks [15], [39]–[43]. On the hand, Chen *et al.* [27] propose to use the Lyapunov optimization for designing a deadline-driven transport protocol. Most recently, Chen *et al.* [26] apply the Lyapunov optimization to schedule a mix of flows with and without deadlines in datacenter networks. Our work is different, as we leverage the Lyapunov optimization to identify the "free" time slots in percentile pricing model, and accordingly send more traffic on such free time slots.

VIII. Conclusions

In this paper, we argue that a simple principle of *"more peak, less differentiation"* should be followed when scheduling inter-DC traffic under the *q-th percentile charging model*. To this end, we present *TrafficShaper*, a new scheduler that leverages the diverse deadlines of inter-DC transfer requests to exploit the "free" time slots involved in the *q-th percentile charging model*. *TrafficShaper* takes the advantage of Lyapunov optimization technique to design and analyze a pricing-aware online control framework. Without a prior knowledge of traffic arrivals, this framework dynamically determines the transmission rates for each inter-DC flow, by efficiently decomposing a long-term optimization into multiple sub-problems that can be sequentially solved in each time slot. To verify the performance of *TrafficShaper*, we conduct rigorous theoretical analysis, large-scale trace-driven simulations and small-scale testbed implementation. All of them have shown that *TrafficShaper* is capable of reducing the transmission cost, while maintaining satisfactory deadline miss rate.

References

[1] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[2] C.-Y. Hong *et al.*, "Achieving high utilization with software-driven WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–25, 2013.

[3] Y. Chen, S. Jain, V. K. Adhikari, Z.-L. Zhang, and K. Xu, "A first look at inter-data center traffic characteristics via Yahoo! Datasets," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 1620–1628.

[4] Forrester Research. *The Future of Data Center Wide-Area Networking*. Accessed: May 23, 2016. [Online]. Available: http://www.forrester.com

[5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[6] "Route optimization for ebusiness applications," RouteSci. Technol., San Mateo, CA, USA, White Paper, 2013, accessed: Jun. 20, 2016. [Online]. Available: http://www.routescience.com/

[7] N. Laoutaris, G. Smaragdakis, P. Rodriguez, and R. Sundaram, "Delay tolerant bulk data transfers on the Internet," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 229–238, 2009.

[8] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. ACM SIGCOMM Conf.*, 2016, pp. 73–86.

[9] R. Stanojevic, N. Laoutaris, and P. Rodriguez, "On economic heavy hitters: Shapley value analysis of 95th-percentile pricing," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, 2010, pp. 75–80.

[10] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 515–526, 2015.

[11] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez, "Inter-datacenter bulk transfers with netstitcher," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 74–85, 2011.

[12] Y. Feng, B. Li, and B. Li, "Jetway: Minimizing costs on inter-datacenter video traffic," in *Proc. 20th ACM Int. Conf. Multimedia*, 2012, pp. 259–268.

[13] L. Golubchik, S. Khuller, K. Mukherjee, and Y. Yao, "To send or not to send: Reducing the cost of data transmission," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 2472–2478.

[14] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synth. Lect. Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[15] F. Liu, Z. Zhou, H. Jin, B. Li, B. Li, and H. Jiang, "On arbitrating the power-performance tradeoff in SaaS clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 10, pp. 2648–2658, Oct. 2014.

[16] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. ACM SIGCOMM Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, 2012, pp. 127–138.

[17] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, 2013.

[18] Q. Pu *et al.*, "Low latency geo-distributed data analytics," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 421–434, 2015.

[19] Forrester Research. (2014). *Measuring Internet Congestion: A Preliminary Report*. [Online]. Available: https://ipp.mit.edu/sites/default/files/documents/ Congestion-handout-final.pdf

[20] B. Briscoe, "Flow rate fairness: Dismantling a religion," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 63–74, 2007.

[21] Z. Zhang *et al.*, "Optimizing cost and performance in online service provider networks," in *Proc. Usenix NSDI*, 2010, pp. 1–15.

[22] V. Valancius, C. Lumezanu, N. Feamster, R. Johari, and V. V. Vazirani, "How many tiers?: Pricing in the Internet transit market," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 194–205, 2011.

[23] *G4 Communications: 95th Percentile Usage Billing Policy*. Accessed: Mar. 26, 2016. [Online]. Available: http://www.g4communications.com/docs/G4_95th_Percentile_Usage.pdf

[24] *Init7: The 95-Percentile-Rule*. Accessed: Mar. 26, 2016. [Online]. Available: http://www.init7.net/en/backbone/95-percent-rule

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

[26] L. Chen, K. Chen, W. Bai, and M. Alizadeh, "Scheduling mix-flows in commodity datacenters with karuna," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, 2016.

[27] L. Chen, S. Hu, K. Chen, H. Wu, and D. H. Tsang, "Towards minimal-delay deadline-driven data center TCP," in *Proc. 12th ACM Workshop Hot Topics Netw.*, 2013, p. 21.

[28] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, vol. 1. Belmont, MA, USA: Athena Scientific, 1995.

[29] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–144, 2006.

[30] G. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 1998.

[31] M. J. Neely, "Delay-based network utility maximization," *IEEE/ACM Trans. Netw.*, vol. 21, no. 1, pp. 41–54, Feb. 2013.

[32] S. Narayana, J. W. Jiang, J. Rexford, and M. Chiang, "To coordinate or not to coordinate? Wide-area traffic management for data centers," Dept. Comput. Sci., Princeton Univ., Princeton, NJ, USA, Tech. Rep. TR-998-15, 2012.

[33] *Yahoo! Research Webscope Program*. Accessed: Dec. 27, 2013. [Online]. Available: http://labs.yahoo.com/organization/academic-relations

[34] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 854–862.

[35] H. Zhang *et al.*, "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. 10th Eur. Conf. Comput. Syst.*, 2015, p. 14.

[36] S. Jia, X. Jin, G. Ghasemiesfeh, J. Ding, and J. Gao, "Competitive analysis for online scheduling in software-defined optical WAN," in *Proc. IEEE INFOCOM*, May 2016, pp. 1–9.

[37] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "Tube: Time-dependent pricing for mobile data," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 247–258, 2012.

[38] L. Zhang, W. Wu, and D. Wang, "The effectiveness of time dependent pricing in controlling usage incentives in wireless data network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 557–558, 2013.

[39] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 1431–1439.

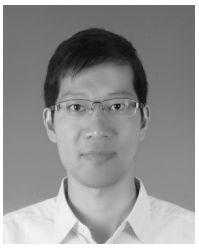
[40] S. Ren, Y. He, and F. Xu, "Provably-efficient job scheduling for energy and fairness in geographically distributed data centers," in *Proc. IEEE ICDCS*, Jun. 2012, pp. 22–31.

[41] W. Deng, F. Liu, H. Jin, and C. Wu, "SmartDPSS: Cost-minimizing multi-source power supply for datacenters with arbitrary demand," in *Proc. IEEE ICDCS*, Jul. 2013, pp. 420–429.

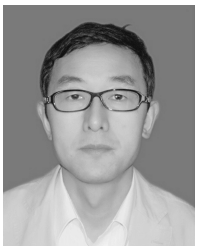
[42] W. Deng, F. Liu, H. Jin, C. Wu, and X. Liu, "Multigreen: Cost-minimizing multi-source datacenter power supply with online control," in *Proc. 4th Int. Conf. Future Energy Syst.*, 2013, pp. 149–160.

[43] Z. Zhou *et al.*, "Carbon-aware online control of geo-distributed cloud services," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2506–2519, Sep. 2016.

Wenxin Li received the B.E. degree from the School of Computer Science and Technology, Dalian University of Technology, China, in 2012, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include datacenter networks and cloud computing.



Xiaobo Zhou received the B.Sc. degree in electronic information science and technology from the University of Science and Technology of China, Hefei, China, in 2007, the M.E. degree in computer application technology from the Graduate University of Chinese Academy of Science, Beijing, China, in 2010, and the Ph.D. degree from the School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan, in 2013. He was a Researcher with the Centre for Wireless Communications, University of Oulu, Finland, from 2014 to 2015. He is currently an Associate Professor with the School of Computer Science and Technology, Tianjin University. His research interests include joint source-channel coding, cooperative wireless communications, network information theory, cloud computing, and software-defined networking.



Keqiu Li (SM'12) received the bachelor's and master's degrees from the Department of Applied Mathematics, Dalian University of Technology, China, in 1994 and 1997, respectively, and the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, in 2005. He held a post-doctoral position at The University of Tokyo, Japan, for two years. He was a Professor with the School of Computer Science and Technology, Dalian University of Technology. He is currently a Professor with the School of Computer Science and Technology, Tianjin University, China. He has published more than 100 technical papers in journals such as the IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *ACM Transactions on Internet Technology*, and *ACM Transactions on Multimedia Computing, Communications, and Applications*. His research interests include Internet technology, data center networks, cloud computing, and wireless networks. He is an Associate Editor of the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS and the IEEE TRANSACTIONS ON COMPUTERS.

Heng Qi received the bachelor's degree from Hunan University in 2004 and the master's and Ph.D. degrees from the Dalian University of Technology, China, in 2006 and 2012, respectively. He was a Lecturer with the School of Computer Science and Technology, Dalian University of Technology. He served as a Software Engineer at GlobalLogic-3CIS from 2006 to 2008. He has published more than 20 technical papers in international journals and conferences, including the *ACM Transactions on Multimedia Computing, Communications and Applications* and *Pattern Recognition*. His research interests include computer network, multimedia computing, and mobile cloud computing.

Deke Guo (SM'17) received the B.S. degree in industry engineering from the Beijing University of Aeronautics and Astronautics, Beijing, China, in 2001, and the Ph.D. degree in management science and engineering from the National University of Defense Technology, Changsha, China, in 2008. He is currently a Professor with the College of Information System and Management, National University of Defense Technology, and a Professor with the School of Computer Science and Technology, Tianjin University. His research interests include distributed systems, software-defined networking, data center networking, wireless and mobile systems, and interconnection networks. He is a member of the ACM.