TINA: A Fair Inter-datacenter Transmission Mechanism with Deadline Guarantee

Xiaodong Dong[1,2], Wenxin Li[3], Xiaobo Zhou[1,2,*], Keqiu Li[1,2] and Heng Qi[4]
[1]College of Intelligence and Computing, Tianjin University, Tianjin, China
[2]Tianjin Key Laboratory of Advanced Networking (TANK)
[3]Hong Kong University of Science & Technology, Hong Kong
[4]School of Computer Science and Technology, Dalian University of Technology, Dalian, China
[*]Corresponding author E-mail: xiaobo.zhou@tju.edu.cn

Abstract—**Geographically distributed cloud is a promising technique to achieve high performance for service providers. For inter-datacenter transfers, deadline guarantee and fairness are the two most important requirements. On the one hand, to ensure more transfers finish before their deadlines, preemptive scheduling policies are widely used, leading to the transfer starvation problem and is hence unfair. On the other hand, to ensure fairness, inter-datacenter bandwidth is fairly shared among transfers with per-flow bandwidth allocation, which leads to deadline missing problem. A mechanism that achieves these two seemingly conflicting objectives simultaneously is still missing. In this paper, we propose TINA to schedule network transfers fairly while providing deadline guarantees. TINA allows each transfer to compete freely with each other for bandwidth. More specifically, each transfer is assigned a probability to indicate whether to transmit or not. We formulate the competition among the transfers as an El Farol game while keeping the traffic load under a threshold to avoid congestion. We then prove that the Nash Equilibrium is the optimal strategy and propose a light-weight algorithm to derive it. Finally, both simulations and testbed experiments results show that TINA achieves superior performance than state-of-art methods in terms of fairness and deadline guarantee rate.**

I. Introduction

With the development of cloud computing, service providers (SPs), large enterprises and organizations heavily rely on geographically distributed public cloud platforms to achieve high performance and improve customers quality of experience (QoE). For purposes of data backup, synchronization, and computation, etc., data transfers between each pair of datacenters are always required by these applications to provide users with better reliability, flexibility, and quality of service (QoS). Therefore, SPs have to rent wide area network (WAN) connections from software-defined wide area network (SD-WAN) vendors or cloud service providers (CSPs). Although the price of Internet transmission continues to decline by approximately 30% per year [1], the traffic volume increases even faster [2]. Therefore, how to efficiently utilize the inter-datacenter bandwidth is the most critical problem for SPs.

To solve the problem, many studies try to fully utilize the inter-datacenter bandwidth. Google presents an inter-datacenter SD-WAN solution, call B4, to improve bandwidth utilization by leveraging the idea of software-defined networking. It utilizes a centralized traffic engineering server to make path decisions and allocate network bandwidth according to transfers' priorities [3]. However, such a high link utilization rate easily causes control traffic disruption and congestion, which may disrupt the control rules and decrease network performance. Microsoft proposes SWAN to solve these problems when the bandwidth utilization is high by leaving a small fraction of bandwidth unused [4]. Leveraging fine-grained per-flow traffic control, these methods are efficient in achieving high link utilization. However, the fine-grained traffic control leaves a high load on the control plane. BwE [5] solves the problem by adopting a hierarchical bandwidth allocation infrastructure that allocates bandwidth in applications, tasks, users and datacenters grains. However, with the rising time-sensitivity of applications and tenants, completing transfers before their specific deadlines become one of the most important performance requirements, especially for transfers with the *all-or-nothing*[1] feature, such as inter-datacenter backup, synchronization, and real-time database query. However, these methods fail to provide any deadline guarantee[2].

To fill this gap, many researches provide deadline guarantee to the transfers with the *all-or-nothing* feature by properly scheduling the transmission order and allocating inter-datacenter bandwidth during their lifetime, e.g., Amoeba [7] and DCRoute [8]. However, these methods will result in unfair situations. Take Amoeba as an example, its objective is finishing as many transfers as possible before their deadlines. Therefore, short transfers that have smaller transfer volumes are transmitted first and assigned with higher bandwidth by Amoeba. It results in that some transfers, especially large transfer, will be delayed for insufficient inter-datacenter bandwidth allocation at the beginning and even get rejected if their deadlines cannot be guaranteed. It is sure that such an

[1]The *all-or-nothing* feature means that a transfer can only obtain utility when it is completed before its deadline. In other words, a transfer cannot complete it before its deadline not only lose the utility but also wastes the precious bandwidth resource.

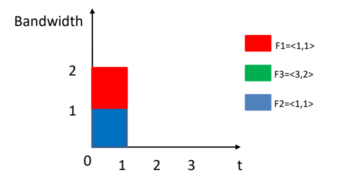
[2]In this paper, without otherwise specified, the deadline refers to the hard deadline. With hard deadline, the transfer will get a penalty if it misses its deadline. With soft deadline, a fractional completion of the transfer is allowed if the network bandwidth is not enough [6]. This will be considered in our future work.

unfair scheduling decision will hurt the performance of some specific applications or tenants [9]. For example, in modern public cloud computing platforms, the network resource is required to be fairly shared by different tenants according to how much money they spend [10]. Thus, these scheduling methods conflict with the fairness requirement of public cloud computing platforms [11].

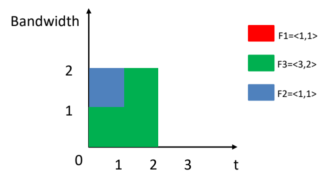
A lot of work has been done to achieve fairness among network transfers, e.g., max-min fairness bandwidth allocation [12]. A max-min fairness bandwidth allocation maximizes the minimum bandwidth allocation. However, these bandwidth allocation methods cannot provide deadline guarantees, especially for transfers with the *all-or-nothing* feature. This is because the scare inter-datacenter bandwidth is simply fairly allocated to each transfer without being properly scheduled. Hence, some transfers with high bandwidth requirements may receive insufficient bandwidth to complete before their deadlines. Therefore, these methods cannot satisfy the time-sensitivity requirements of applications and tenants. In summary, fairly scheduling transfer with deadline guarantee is a challenging problem with two seemingly conflicting objectives. On the one hand, the system needs to concentrate the scare inter-datacenter bandwidth on some specific transfers to provide deadline guarantees. On the other hand, the scare inter-datacenter bandwidth is also required to be fairly shared by all transfers to achieve fairness.

In this paper, we investigate the transfer scheduling problem in inter-datacenter networks, and propose a fair inter-datacenter transmission mechanism with deadline guarantees, called TINA, to schedule the transfers between each pair of datacenters in a fair way. The key idea of TINA is allowing each transfer to compete freely with each other for inter-datacenter bandwidth resource. More specifically, each transfer will be assigned a probability to indicate whether to transmit or not. First, we formulate the competition between all the transfers as an El Farol game with the aim of maximize the utility of each transfer, while avoiding congestion and collision by keeping the traffic load of the link under a certain threshold to guarantee transfer deadlines. Then, we prove that its Nash Equilibrium is the optimal traffic schedule strategy, and present a light-weight algorithm to derive the Nash Equilibrium. In summary, the main contributions of this paper are outlined as follows:

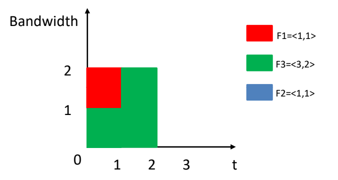
- We propose a fair inter-datacenter transfers scheduling mechanism with deadline guarantees, called TINA, which allows each transfer to compete freely with each other for the bandwidth resource, while avoiding congestion and collision by keeping the traffic load of the link under a certain threshold to guarantee transfer deadlines.
- We model the competition among all the transfers between each pair of datacenters as an El Farol bar game, where each transfer aims at maximizing its own utility. We further prove that its Nash Equilibrium is the optimal traffic schedule strategy.
- We show that TINA can be extended to a tenant level fairness traffic schedule algorithm that schedules transfers fairly from tenant's perspective.
- We conduct both simulation and testbed experiments against state-of-art methods, the experiment results show that TINA achieves fair inter-datacenter transfer scheduling with deadline guarantee, and the network performance in terms of link utilization, acceptance ratio, and total utility.



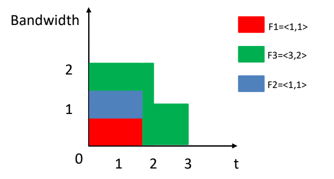
(a) Amoeba tries to finish as many transfers as possible before their deadlines.



(b) Feasible scheduling strategy.



(c) Feasible scheduling strategy.



(d) Max-min fairness strategy.

Fig. 1: Fairness and deadline guarantee are two conflicting objectives when scheduling inter-datacneter transfers.

The rest of the paper is structured as follows. Section II presents the motivation and preliminary of this paper. In Section III, we overview the system and formulate the problem as an El Farol game. The algorithm that derives the optimal sending probability of each transfer is presented in Section IV. Section V presents performance evaluations. Related work is reviewed in Section VI. At last, Section VII concludes the paper.

II. Motivation

A. The Unfairness of Existing Work

Consider the fact that transfers compete with each other for the scare inter-datacenter bandwidth. To accommodate these two conflicting objectives, we introduce the concept of Rabin fairness to describe the fairness of the competition relationship between transfers, which is used to describe the fairness in wireless network admission control [13]. The Rabin fairness model implies that each participant will treat the other participants kindly if the other participants treat him kindly, and each participant will treat the other participants badly if the other participants treat him badly. A strategy is referred to as *fair equilibrium* if such a situation holds, which means that the competition between these participants is fair. Leveraging Rabin fairness, we can judge whether transfers are fairly competing with each other for the network bandwidth or not. As long as the competition is fair, the scheduling decision is fair for all transfers in the network system. Motivated by [13], we argue that an admission control mechanism can achieve such a fair competition environment by assigning each transfer a probability that indicates whether to transmit it or not, while avoiding the link from being congested, which can further guarantee the accepted transfers' deadlines.

We present a simple example to illustrate the unfairness of the existing work in Fig. 1. More specifically, at the current time slot, there are three transfers each belong to one specific application waiting to be scheduled, denoted by a tuple consists of traffic volume and deadline, i.e., $f_1 = \{1, 1\}$, $f_2 = \{1, 1\}$ and $f_3 = \{3, 2\}$. As shown in Fig. 1a, in order to complete as many transfers as possible before their deadline, Amoeba accepts f_1 and f_2 for the low cost they introduce, but rejects f_3 for insufficient bandwidth. Nevertheless, f_3 can also be completed before its deadline with the strategy shown in Fig. 1b and Fig. 1c. According to [13], such a deterministic scheduling strategy achieved by Amoeba is the most unfair strategy, which will cause serious problems mentioned in the previous section. As shown in Fig. 1d, with a max-min fairness bandwidth allocation strategy, the minimum bandwidth allocation is maximized. Hence, without being assigned with sufficient bandwidth, all three transfers miss their deadlines. This example clearly shows that fairness and deadline guarantee are two seemingly conflicting objectives.

B. Preliminary

The El Farol bar game was first introduced by Arthur [14] as a framework to investigate how to model the bounded rationality in economics. The original problem is described as follows: there is a finite population of people and every Thursday night all of them want to go to the El Farol bar. However, the El Farol bar is quite small, and it is not enjoyable to go there if it is too crowded where the following rules are in place:

- If less than 60% of the population go to the bar, those who go have a more enjoyable evening at the bar than they would have had if they stayed at home.
- If 60% or more of the population go to the bar, those who go have a worse evening at the bar than they would have had if they stayed at home.

It is proved that there exists a symmetric unique mixed Nash equilibrium [15] that each person determines to go to the bar with a specific probability. According to [16], the Nash equilibrium of this game is proved to be *fair equilibrium*.

Therefore, leveraging the idea of El Farol bar game, we can design an admission control mechanism to assign each transfer with a probability to determine whether to transmit it or not, while guaranteeing the fairness of competition between transfers without modifying transmission protocol. However, in our scenario, transfers between each pair of datacenters have various bandwidth requirements, which makes our problem different from the original El Farol bar game where each person occupies only one seat of the bar. Thus, the symmetric mixed Nash equilibrium of the original model is inapplicable to our model. We proved that there is an asymmetric mixed Nash equilibrium to the problem considered in this paper, which is detailed in **Theorem 1** in Section IV.

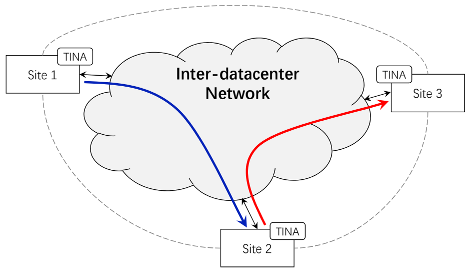


Fig. 2: The architecture of inter-datacenter network.

III. Design of TINA

In this section, we present the detail of TINA. First, we present the system overview. Then we formulate the competition between all the transfers as an El Farol game.

A. System Overview

Geographically distributed public cloud is a technology to provide resources, e.g. virtual machines, storage and network to users from multiple geographic locations. To connect datacenters form different locations, the cloud service provider has to rent WAN connections with fixed uplink bandwidth from SD-WAN vendors or ISPs. A direct graph (N, L) is used to represent the inter-datacenter network. N denotes the node set, where each node $n \in N$ can be a datacenter. L denotes the link set, where each $l_{mn} \in L$ represents the virtual link from datacenter m to datacenter n with corresponding link capacity C_{mn}. Fig. 2 illustrates the system architecture of the geographically distributed public cloud platform, where the inter-datacenter is congestion free. TINA works distributively at the side of each datacenter independently.

B. Problem Formulation

We consider the system that operates in a discrete-time mode with T time units. Since TINA works independently at each datacenter and makes scheduling decisions for each link, we omit the source datacenter index m and destination datacenter index n for simplicity. Then, on a specific link l, A transfer represents a tenant's data delivery demand from the source datacenter to the destination datacenter, which is specified as a tuple $f_i = \{V_i, T_i^s, T_i^d, e_i\}$, where V_i is the traffic volume, T_i^s is the transmission starting time, T_i^d is the deadline and e_i is the expected transfer rate. e_i can be obtained by

$$e_i = \frac{V_i}{T_i^d - T_i^s}. \tag{1}$$

Further, at a specific time slot t, we organize all existing transfers that are waiting to be transmitted from the source site as a set F_t and denote the total number of transfers as $|F_t|$. Similarly, the transmitting transfers set is represented as A_t, and we use W_t to represent the bandwidth these transfers have occupied, which is calculated as

$$W_t = \sum_{f_i \in A_t} e_i. \tag{2}$$

TABLE I: Notations

Symbols	Definitions
N	Datacenter set
L	Virtual link set
$f_i=\{V_i,T_i^s,T_i^d,e_i\}$	Transfer
V_i	Traffic volume of f_i
T_i^s	Starting time of f_i
T_i^d	Deadline of f_i
e_i	Expected transfer rate of f_i
F_t	Waiting transfer set at time slot t
A_t	Transmitting transfer set at time slot t
W_t	Occupied bandwidth at time slot t
C	Link capacity
α	Congestion threshold
ϑ	Link state index
S_i	The utility for not sending f_i
π_i	Utility function of f_i
G_i	The utility for sending f_i without causing congestion
B_i	The utility for sending f_i but causing congestion
p_i	The probability for sending f_i to the network
$\mathbb{S}(\cdot)$	Feasible scheduling decision set
Γ	El Farol bar game

Let C represent the link capacity, and α denote the congestion threshold. We use a boolean variable ϑ to indicate the state of the link, denoted as

$$\vartheta=\begin{cases}0, & \sum\limits_{f_i\in F_t} x_ie_i>\alpha\cdot C-W_t,\\ 1, & \sum\limits_{f_i\in F_t} x_ie_i\le\alpha\cdot C-W_t,\end{cases} \tag{3}$$

where $x_i=1$ denotes f_i is accepted and $x_i=0$ denotes that f_i is not going to be sent at this moment. If $\vartheta=0$, the link is regarded as congested. Otherwise, we believe the link is free of congestion. More specifically, $x_i\in\{0,1\}$ follows Bernulli distribution, which is denoted as

$$x_i=\begin{cases}0, & 1-p_i\\ 1, & p_i.\end{cases} \tag{4}$$

Here, p_i represents the probability that f_i will be transmitted.

Further, the utility function can be written as

$$\pi_i(\vartheta)=\begin{cases}S_i, & \vartheta=1,x_i=0,\\ S_i, & \vartheta=0,x_i=0,\\ G_i, & \vartheta=1,x_i=1,\\ B_i, & \vartheta=0,x_i=1.\end{cases} \tag{5}$$

More specifically, each flow f_i consists of an unconditional utility for not sending, denoted by S_i, and a condition utility for sending the transfer, denoted by G_i or B_i, depending on the state of the link. Clearly, there are two states of the link, congested or not, and the state is determined by the remaining $|F_t|-1$ flows. Consider the *all-or-nothing* feature of transfers in this paper, we have $G_i>S_i>B_i$. The notations are summarized in Table I.

Consider the fact that each transfer competes with each other for bandwidth and make independent decision to maximize their expected utility in the network. According to von Neumann's Minimax Theorem, the objective of each transfer in such a fair competition scenario can be modeled as

$$\begin{aligned}&\max_{p_i} E\big(\pi_i(\vartheta)\big)\\ &s.t.\quad p_j\in(0,1),\forall f_j\in F_t,\end{aligned} \tag{6}$$

where

$$\begin{aligned}E\big(\pi_i(\vartheta)\big)=&(1-p_i)S_i+p_iP_i(F_t,W_t,\mathbf{p}_t)G_i\\ &+p_i\big(1-P_i(F_t,W_t,\mathbf{p}_t)\big)B_i,\end{aligned} \tag{7}$$

and $P_i(F_t,W_t,\mathbf{p}_t)$ represents the probability that the other transfers of the remaining $|F_t|-1$ transfers choose to send occupy less than $\alpha C-W_t-e_i$ bandwidth with $\mathbf{p}_t=<p_1,...,p_{|F_t|}>$, which is defined by the following binomial probability

$$P_i(F_t,W_t,\mathbf{p}_t)=\sum_{\mathbb{S}(W_t,i)}\prod_{f_j,j\neq i}^{F_t}p_j^{x_j}(1-p_j)^{1-x_j},\forall f_i\in F, \tag{8}$$

where $\mathbb{S}(W_t,i)$ is the feasible scheduling decision set, which is represented as

$$\mathbb{S}(W_t,i)=\Big\{<x_j>_{f_j,j\neq i}^{F_t}\mid\sum_{f_j,j\neq i}^{F_t}x_je_j\le\alpha C-W_t-e_i\Big\}. \tag{9}$$

Therefore, the objective of TINA can be represented as optimization problem **P1**

$$\begin{aligned}&\max_{\mathbf{p}_t}\sum_{f_i\in F_t}E\big(\pi_i(\vartheta)\big)\\ &s.t.\quad p_j\in(0,1),\forall f_j\in F_t,\\ &p_i=\arg\max_{p_i}E\big(\pi_i(\vartheta)\big),\forall f_i\in F_t,\end{aligned} \tag{10}$$

Given the above preliminaries, at each time slot, we define the El Farol bar game as a single stage game with objective described as **P1**.

Definition 1: The El Farol game is defined as a single stage game, denoted as a vector, $\Gamma=<F_t,\mathbf{p}_t,\pi,W_t>$.

IV. Algorithms

In this section, we first prove its Nash Equilibrium is the optimal solution to maximize the total system utility while guaranteeing the fairness of each transfer and propose a light-weight algorithm to derive a near-optimal solution. Then, we also design another tenant level fairness traffic schedule algorithm to provider tenant fairness.

A. Nash Equilibrium and Solution

According to [17], each transfer obtains the maximum expected utility only when it adopts the Nash equilibrium which is also the solution of problem **P1**.

Theorem 1: In the El Farol game, there is a mixed strategy equilibrium where each transfer play the mixed strategy defined by the strategy tuple $\{p_i,1-p_i\}$. Furthermore, p_i is defined by the following relationships

$$\sum_{\mathbb{S}(W_t,j)}\prod_{f_i,i\neq j}^{F_t}p_i^{x_i}(1-p_i)^{1-x_i}=\frac{S_j-B_j}{G_j-B_j},\quad\forall f_j\in F, \tag{11}$$

Proof: The proof can be referred to the Appendix. ■

Organizing $\mathbf{p}_t =< p_1, ..., p_{|F_t|} >$, (11) can be written into a matrix, $\mathbf{p}_t = Y(\mathbf{p}_t)$. It is obvious that the Nash equilibrium of El Farol game forms a series of nonlinear equations. To derive the optimal $\mathbf{p}_t$, a direct iteration method is adopted to solve the problem. To stop the algorithm in finite steps, we introduce an escape threshold $\omega \in (0, 1)$. The iteration stops when $|\mathbf{p}_t^{n+1} - \mathbf{p}_t^n| < \omega$ and the current solution $\mathbf{p}_t^{n+1}$ is viewed as the solution.

To accelerate the computing procedure, at each time slot t, we introduce a baseline flow $\widetilde{f} =< \widetilde{V}, T^s, T^d, \widetilde{e} >$ where $\widetilde{V}$ and $\widetilde{e}$ are the average transfer volume and expected transfer rate of transfers in F_t. Correspondingly, $\widetilde{f}$ is a associated with utility function $\widetilde{\pi}$ where $\widetilde{G} = \sum_i^{|F_t|} \frac{G_i}{|F_t|}$, $\widetilde{S} = \sum_i^{|F_t|} \frac{S_i}{|F_t|}$ and $\widetilde{B} = \sum_i^{|F_t|} \frac{B_i}{|F_t|}$. Since transfers with higher utility and smaller expected transfer rate have higher probability to get access to the network, we define

$$p_i = \widetilde{p}^{\frac{\widetilde{e}}{e_i} + \frac{\lambda(\widetilde{S}-\widetilde{B})(G_i - B_i)}{(\widetilde{G}-\widetilde{B})(S_i - B_i)}}, \qquad (12)$$

where $\widetilde{p}$ is the probability of sending $\widetilde{f}$ and λ is the weight. When calculating $\widetilde{p}$, we assume that each transfers in F_t is same as $\widetilde{f}$. According to (11), we have

$$\sum_{m=1}^{\lfloor \frac{\alpha C - W_t}{\widetilde{e}} \rfloor} \mathcal{C}_m^{|F_t|} \widetilde{p}^m (1 - \widetilde{p})^{|F_t| - m} = \frac{\widetilde{S} - \widetilde{B}}{\widetilde{G} - \widetilde{B}}. \qquad (13)$$

Moreover, the total expected transfer rate of accepted transfers may exceed the link capacity because each transfer has a certain probability to get access to the network. This problem can be solved by iteratively conducting TINA's admission control mechanism until the link capacity constraint is satisfied. The details of TINA are summarized in Algorithm 1. The lines 3-8 remove the transfers that cannot be completed before their deadlines. The lines 10-20 calculate $\mathbf{p}_t$. The line 21 determines whether each transfer should be transmitted or not according to $\mathbf{p}_t$. The lines 22-27 guarantee the traffic load is under the predefined threshold α and line 28 updates the system state. It is clear that the complexity of TINA is $O(|F_t|)$ at each time slot. Moreover, without leveraging per-flow bandwidth allocation, we make the following two improvements to guarantee transfer deadlines: 1) Only remove transfer from A_t when its deadline is passed. 2) Assign transfers approaching their deadlines with high priority. Besides, the higher e_i is, the higher the priority is assigned. Note that existing work has already proved that per-flow rate control and bandwidth allocation leaves 17% CPU and 30% memory overhead on end host [18], without leveraging per-flow bandwidth allocation and rate control on the end host, TINA achieves traffic scheduling in a light-weight style. In the following section, the experiment results show the effectiveness of TINA.

B. Tenant Level Fair TINA

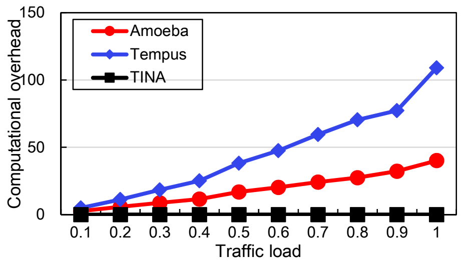
According to the above descriptions, TINA allows each transfer to compete with each other to maximize its own utility. In this subsection, we show that TINA can also be easily extended to a tenant level fair transmission mechanism. In tenant level transmission mechanism, the tenants with a huge amount of transfers may occupy most of or even all of the bandwidth, which results in unfairness in tenant level, even if the transfer level fair transmission mechanism is applied.

```
Algorithm 1: TINA
Input: Link capacity $C$; Maximum iteration number $T$;
       Waiting transfer set $F$; Transmitting transfer set $A$;
       Utility function $\pi$;
Output: $\mathbf{p}_t$;
Initialize: Congestion threshold $\alpha$;
       Link capacity $C$; Escape threshold $\omega$;
1:  for(t=1;t≤ $T$;t++)
2:    Update $F_t$;
3:    foreach($f_i \in F_t$)
4:      calculate $e_i$ according to (1);
5:      if($e_i > \alpha C - W_t$)
6:        $F_t = F_t - f_i$;
7:      endif
8:    endfor
9:    while(True)
10:     calculate $\widetilde{f} =< \widetilde{V}, Ts, Td, \widetilde{e} >$ and $\widetilde{\pi}$;
11:     Initialize $\widetilde{p} = Y(\widetilde{p})$ according to (13);
12:     Initialize $n = 0$ and $\widetilde{p}^0$
13:     while(True)
14:       $\widetilde{p}^{n+1} = Y(\widetilde{p}^n)$;
15:       if($|\widetilde{p}^{n+1} - \widetilde{p}^n| < \omega$)
16:         $\widetilde{p} = \widetilde{p}^{n+1}$;
17:         break;
18:       $n = n + 1$;
19:     endwhile
20:     calculate $\mathbf{p}_t$ according to (12);
21:     Determine $X$ according to $\mathbf{p}_t$;
22:     if($\sum_{f_i \in F_t} x_i e_i \le \alpha \cdot C - W_t$)
23:       break;
24:     else
25:       $F_t = \{f_i | x_i = 1\}$;
26:     endif
27:   endwhile
28:   Update $A_t$ and $W_t$;
29: endfor
```

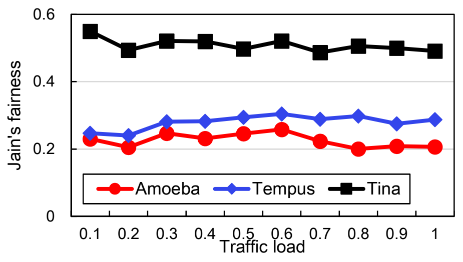
At a time slot t, we assume that there are M tenants coexisting in the system, and each one of them is associated with its waiting transfers set F_t^m and transmitting transfers set A_t^m, respectively. For each transfer, its utility function π_i^m is pre-known. To provide tenant level fairness, the link capacity of each tenant is set to $C^m = C/M$. Then the problem is transformed into determining $\mathbf{p}_t^m$ for F_t^m with the constraints C^m and W_t^m, which can be solved in the same way as TINA. Note that the transfer scheduling decision of each tenant is independent with each other, it can be divided into a set of sub-problems of each tenant, which can be solved in parallel. Besides, we can also set C^m according to some specific weights to implement weighted fairness.

V. Performance Evaluation

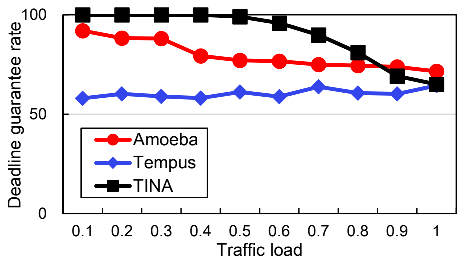
In this section, we evaluate the performance of TINA against Tempus [21] and Amoeba through large-scale simulations and small-scale testbed experiments. Tempus aims to maximize the minimum fraction of transfers completes before their deadline, while Amoeba tries to finish as many transfers before their deadlines as possible.



(a) Computational overhead of different mechanisms.

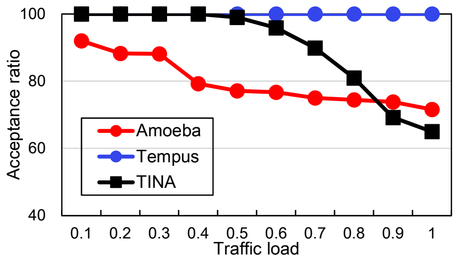


(b) Average fairness of different mechanisms.

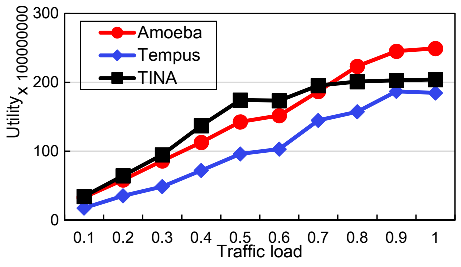


(c) Deadline guarantee rate of different mechanisms.

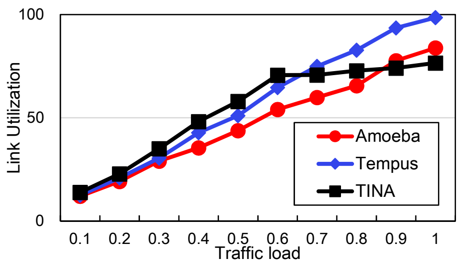
Fig. 3: Simulation results on computational overhead, fairness and deadline guarantee rate.



(a) Acceptance ratio of different mechanisms.



(b) Total utility of different mechanisms.



(c) Link utilization of different mechanisms.

Fig. 4: Simulation results on acceptance ratio, total utility and link utilization.

A. Large-scale Simulation

Experiment Setting: We first simulate a WAN with 15 sites, where each pair of sites is connected with a bandwidth committed link. The bandwidth capacity of each link is set to be 200 Gbps. Moreover, according to [6], we reserve 5% to 15% bandwidth for background traffic. Meanwhile, we set $\alpha = 0.8$, $G = V$, $S = 0$ and $B = -V$, and each time slot is 5 minutes. The lifetime of the simulation experiments is 288 time slots. We randomly select a pair of datacenters as the source and the destination. Similar to previous work [19] [20], inter-datacenter transfers are generated with the following parameters:

- Request arrival time is modeled as a Possion process with arrival rate μ per second.
- $\frac{V}{T^d - T^s}$ follows an exponential distribution with a mean of 2 Gbps.
- The transfer deadline follows an exponential distribution with a mean of 10 time slots.
- Each transfer contains 1 session.

The following metrics are taken into consideration: link utilization, utility, deadline guarantee rate, acceptance ratio, fairness, and computational overhead. As TINA is proved to be *fair equilibrium*, to quantitatively measure the fairness of the inter-datacenter transmission mechanisms, we use Jain's fairness index which is defined as

$$\mathbb{J}(F_t) = \frac{(\sum_{i=1}^{|F_t|} p_i e_i)^2}{|F_t| \sum_{i=1}^{|F_t|} (p_i e_i)^2}. \tag{14}$$

Experiment Results: We compare TINA, Amoeba, and Tempus in terms of computational overhead, fairness, and deadline guarantee rate in Fig. 3a, 3b, and Fig. 3c, respectively. As depicted in Fig. 3a, the computational overhead for Tempus, Amoeba, and TINA increase as the traffic load increases. It is clear that the computational overhead of TINA is significantly lower than the computational overhead of Tempus and Amoeba. This is because TINA utilizes a simple direct iteration method to derive the optimal strategy without solving complex optimization problems adopted by Aemon and Tempus. The results of average Jain's fairness with different traffic loads are shown in Fig. 3b. It is obvious that TINA is the fairest method among them and Tempus obtains slightly better fairness than Amoeba. Fig. 3c depicts the results of the deadline guarantee rate. It can be found that the deadline guarantee for both TINA and Amoeba decreases

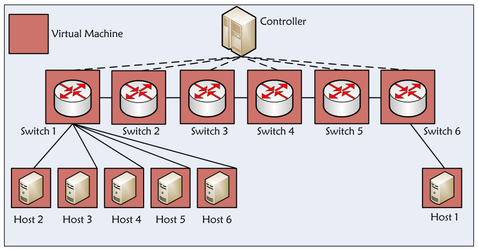


Fig. 5: The topology of testbed.

as the increase of traffic load, and Tempus achieves the lowest deadline guarantee rate. This is because TINA offers each transfer with a fair competition environment where each transfer has a probability to transmit. Tempus tends to transmit a fraction of each transfer without providing *all-or-nothing* deadline guarantee. Amoeba aims to complete as many as transfers as possible and simply rejects transfers that cannot be completed before their deadlines, which results in transfers with large traffic volume are more likely rejected. Moreover, Amoeba achieves a higher deadline guarantee rate than TINA when the traffic load is higher than 0.8. This is because that TINA views the network as congested when the traffic load is larger than 0.8. To avoid network congestion and deadline missing, TINA does not fully utilize the bandwidth.

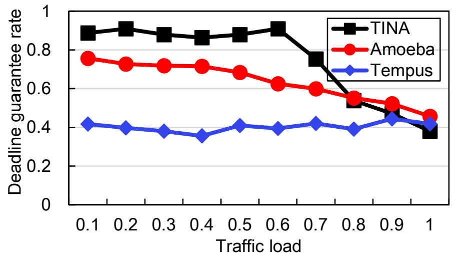
We also compare TINA, Amoeba, and Tempus in terms of acceptance ratio, utility and link utilization in Fig. 4a, 4b, 4c, respectively. Fig. 4a depicts the results of the acceptance ratio. It is clear that the acceptance ratio decreases for both TINA and Amoeba when the traffic load increases, and Tempus accepts all transfers. Meanwhile, TINA achieves a higher acceptance ratio than Amoeba when the traffic load is lower than 0.8. Fig. 4b and 4c depict the utility and link utilization. Obviously, TINA achieves the highest utility and link utilization when the traffic load is lower than 0.6, and Amoeba obtains the best performance when the traffic load is larger than 0.6, while Tempus achieves the highest link utilization but the lowest utility. These results are consistent with the objectives of Tempus, Amoeba, and TINA. Tempus aims to maximize the minimum faction of transfers before their deadline. However, it fails to confirm that transfers can complete before their deadline, which leads to high acceptance ratio and link utilization but low deadline guarantee rate. While Amoeba aims to complete as many transfers as possible before their deadlines and rejects transfers that cannot be fully served, which leads to low acceptance ratio and link utilization but a high deadline guarantee rate. TINA rejects transfers when the traffic load is too high to avoid network congestion. The reason why TINA is inferior to Amoeba when the traffic load is high is because the congestion threshold is set to be 0.8 during the experiment.

B. *Small-scale Testbed Implementation*

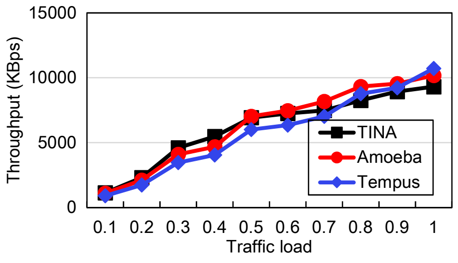
Experiment Setting: We also emulate a small topology on a server with 8 10-core Intel Xeon E7-4820 2.00 GHz CPU, 256 G memory, 500 GB hard disk, Broadcom BCM5719 NetXtreme Gigabit Ethernet NIC, running Ubuntu 16.04 with Linux 4.4.0 kernel. The topology contains 6 switches, 6 hosts and 11 bidirectional links, which is shown in Fig. 5. Each host and switch runs in separate virtual machine (VM) with 4 cores, 2 G memory and 20 GB hard disk. The end host VMs run Ubuntu 12.04 operating system with 2.26.38 kernel, the switch VMs run Ubuntu 16.04 operating system with 4.4.0 kernel. The capacity of each link is 100 MB and RTT=100 ms. We assume all the switches are homomorphism and each switch has 8 M memory shared by all its' queues. We also set a large enough Ternary Content Addressable Memory (TCAM) space to store flow entries. To eliminate the impact of control traffic, the controller runs on the server and is connected in an out-band style, and communicates with all switches with OpenFlow 1.3.

We develop a client/server model packet generator to generate traffic. Host 1 generates request to other hosts to fetch data, and the other hosts respond with the requested data. Each time slot is 1s, and the experiment consists of 120 time slots.

Experiment Results: The performance of TINA on testbed matches with the results of simulation experiments. Since the testbed experiment results on fairness, computational overhead, acceptance ratio, and utility are similar to the results of simulation experiments, for the sake of space limitation, we just omit the discussion on these metrics. The deadline guarantee rate and the throughput are shown in Fig. 6. The deadline guarantee rates of different methods are shown in Fig. 6a. It is clear that the experiment results almost match the results in the simulation experiment. However, the deadline guarantee rates of different methods in the testbed experiment are lower than the results in the simulation experiment. This is because there are many short transfers that miss their deadlines even before their connection are established. Fig. 6b shows the



(a) Deadline guarantee rate of different mechanisms.



(b) Throughput of different mechanisms.

Fig. 6: Testbed experiment results.

throughput of different methods. Note that Tempus performs worst when the traffic load is lower than 0.7, which conflicts with the results of simulation experiments. This is because that there are many short flows which are so short that the bandwidth allocation granularity is even larger than their size when the traffic load is low. Complete a fraction of them will cause bandwidth allocation redundancy. In summary, the experiment results further validate the conclusion we made previously.

VI. Related Work

Inter-datacenter traffic scheduling: There is plenty of work focus on traffic scheduling in inter-datacenter networks from different aspects. To increase link utilization, NetStitcher [22] uses a store-and-forward algorithm to schedule data transfers, and adapts to resource and traffic fluctuations. Google presents their inter-DC SD-WAN in 2013, called B4 [3]. With a centralized traffic engineering controller, it drives links to full utilization by per-flow routing management and bandwidth allocation. To avoid control traffic congestion under high link utilization, SWAN [4] reserves a small amount of link capacity to apply network configuration and updates. To avoid the high control plane overhead caused by per flow management, BwE [5] uses a hierarchical bandwidth allocation infrastructure to allocate bandwidth in applications, service, users, datacenters groups. In MON [23], a mission optimized overlay network architecture is proposed to maximize total utility with a combination of two tiers system where an offline tier makes scheduling and routing decision and an online tier monitors the network condition. In [24], the authors present QuickCast, which minimizes the total completion time by multicasting inter-datacenter transfers leveraging optimized multiple forwarding trees. However, these methods fail to manage transfer deadline that is a very critical performance requirement.

Deadline guarantee: In [21], the authors present an online traffic scheduling method, called TEMPUS, which appropriately schedules long-term transfers across network paths and future time slots. In [6], the authors maximize the total utility with taking both soft deadline and hard deadline into account, where a transfer's utility will decrease after the soft deadline is violated. However, these methods do not provide an *all-or-nothing* deadline guarantee. While Amoeba [7] considers the *all-or-nothing* feature of transfers and tries to finish as many transfers as possible before their deadlines. Nevertheless, it relies on solving a high complexity optimization problem to make scheduling decisions. DCRoute [8] is proposed to schedule traffic with deadline guarantee in a fast and efficient way with a greedy algorithm only utilizing the remained bandwidth. Nevertheless, both of them neglect the fact that fairness is required in a modern public cloud computing platform, so that there is a mismatch between these methods and the fairness requirements of the modern public cloud service providers that we have already explained.

Fairness: A common and well studied notion of fairness is max-min fairness. However, finding a max-min fair allocation requires an iterative solution of multiple linear programs, which brings in high computational overhead. To solve this problem, a novel relaxation of max-min fairness and an efficient combinatorial algorithm which is proved to be converged are presented in [12]. HUG [9] also shares the network bandwidth for coflows in a max-min fairness manner while maintaining a high network utilization rate. However, these methods only fairly share network bandwidth among transfers. As mentioned above, how to fairly schedule inter-datacenter transfers while guaranteeing transfer deadlines is still a challenging problem.

VII. Conclusion

In this paper, we investigate whether inter-datacenter transfers can be scheduled in a fair style while still guaranteeing their deadlines. To this end, we present TINA, a novel inter-datacenter transfer mechanism, which allows each transfer to compete freely with each other for the bandwidth resource. More specifically, each transfer will be assigned a probability to indicate whether to transmit or not. First, we model the competition between transfers as an El Farol bar game with the aim of maximize the utility of each transfer, while keeping the traffic load under a threshold to avoid congestion and collisions. Then, we obtain the optimal sending probabilities by deriving its Nash Equilibrium. Moreover, we show TINA can also be easily extended to a tenant level fair inter-datacenter transmission mechanism. Finally, to verify the performance of TINA, we conduct both simulation and testbed experiments against the state-of-art methods. The simulation and testbed experiment results show that TINA achieves superior performance in terms of both fairness and deadline guarantee rate, without sacrificing the throughput, link utilization, total utility, and acceptance ratio. As future work, we will consider soft deadlines to enable a more flexible transmission mechanism.

Acknowledgment

This work was supported in part by National Key R&D Program of China under Grant 2019QY1302, in part by the NSFC General Technology Basic Research Joint Funds under Grant U1836214, in part by the State Key Program of National Natural Science of China under Grants 61832013, in part by the Artificial Intelligence Science and Technology Major Project of Tianjin (Grant No. 18ZXZNGX00190), in part by the National Natural Science Foundation of China under Grant 61672379, in part by National Key R&D Program of China under Grant 2019YFB2102404, in part by NSFC-Guangdong Joint Funds under Grant U1701263, in part by the Natural Science Foundation of Tianjin under Grant No. 18ZXZNGX00040, in part by National Key R&D Program of China under Grant 2018YFB1004700, in part by the National Natural Science Foundation of China under Grant 61872265, in part by the National Natural Science Foundation of China under Grant 61672131, in part by the Key research and Development Program for Guangdong Province 2019B010136001.

Appendix
Proof of Theorem 1

Existence: In particular, in the mixed strategy Nash equilibrium, each transfer f_i should be indifferent between sending or not sending. Therefore, we have

$$E[\pi_i(\vartheta)|x_i = 1] = E[\pi_i(\vartheta)|x_i = 0], \forall f_i \in F_t. \tag{15}$$

Consider (5), we have

$$\begin{aligned} E[\pi_i(\vartheta)|x_i = 1] &= G_i P_i(F_t, W_t, \mathbf{p}_t) \\ &\quad + B_i\big(1 - P_i(F_t, W_t, \mathbf{p}_t)\big), \quad \forall f_i \in F_t, \end{aligned} \tag{16}$$

$$E[\pi_i(\vartheta)|x_i = 0] = S_i, \forall f_i \in F_t. \tag{17}$$

By substituting (16) and (17) into (15), we have

$$P_i(F_t, W_t, \mathbf{p}_t) = \frac{S_i - B_i}{G_i - B_i}, \forall f_i \in F_t. \tag{18}$$

Therefore, the existence of Theorem 1 is proved.

Uniqueness: It is clear $P_i(F_t, W_t, \mathbf{p}_t)$ is continuous function over interval $[0, 1]$. Given that $\lim_{p\to 0} P_i(F_t, W_t, \mathbf{p}_t) = 1$ and $\lim_{p\to 1} P_i(F_t, W_t, \mathbf{p}_t) = 0$, Theorem 1 has a unique solution if the partial derivative of $P_i(F_t, W_t, \mathbf{p}_t)$ is less than 0. For the sake of simplicity, we calculate one partial derivative of $P_i(F_t, W_t, \mathbf{p}_t)$ as the following

$$\begin{aligned} &\frac{\partial P_i(F_t, W_t, \mathbf{p}_t)}{\partial p_k} = \frac{\partial \sum_{\mathbb{S}(W_t,i)} \prod_{f_j, j\neq i}^{F_t} p_j^{x_j}(1-p_j)^{1-x_j}}{\partial p_k} \\ &= \sum_{\mathbb{S}(W_t,i)} \Big\{ x_k p_k^{x_k - 1}(1-p_k)^{1-x_k} - \\ &(1-x_k)p_k^{x_k}(1-p_k)^{-x_k} \Big\} \prod_{f_j, j\neq i,k}^{F_t} p_j^{x_j}(1-p_j)^{1-x_j} \end{aligned} \tag{19}$$

Obviously, if $x_k = 1$, we have

$$\frac{\partial P_i(F_t, W_t, \mathbf{p}_t)}{\partial p_k} = \sum_{\mathbb{S}(W_t,i|x_k=1)} \prod_{f_j, j\neq i,k}^{F_t} p_j^{x_j}(1-p_j)^{1-x_j}, \tag{20}$$

otherwise, if $x_k = 0$, we have

$$\frac{\partial P_i(F_t, W_t, \mathbf{p}_t)}{\partial p_k} = -\sum_{\mathbb{S}(W_t,i|x_k=0)} \prod_{f_j, j\neq i,k}^{F_t} p_j^{x_j}(1-p_j)^{1-x_j}, \tag{21}$$

where $\mathbb{S}(W_t, i|x_k)$ denotes the feasible set with corresponding x_k value.

Since for every element in $\mathbb{S}(W_t, i|x_k = 1)$, there is a corresponding element in $\mathbb{S}(W_t, i|x_k = 0)$ that only changes x_k from 1 to 0, therefore we have

$$|\mathbb{S}(W_t, i|x_k = 1)| \leq |\mathbb{S}(W_t, i|x_k = 0)|. \tag{22}$$

Moreover, consider that $\prod_{f_j, j\neq i,k}^{F_t} p_j^{x_j}(1-p_j)^{1-x_j}$ is a nonnegative fixed value, therefore, we have

$$\frac{\partial P_i(F_t, W_t, \mathbf{p}_t)}{\partial p_k} <= 0, \tag{23}$$

Similar conclusion can be obtained by replacing p_k. Thus, the uniqueness of **Theorem** 1 is proved.

References

[1] W. B. Norton. Drpeering white paper. http://drpeering.net/white-papers/Internet-Transit-Pricing-Historical-And-Projected.php.

[2] C. Labovitz, S. I.-Johnson, D. McPherson, J. Oberheide, and F. Jahanian. Internet inter-domain traffic. In *Proc. of ACM SIGCOMM 2010 Conference*, pp. 75–86, 2010.

[3] S. Jain, A. Kumar and S. Mandal et al. B4: experience with a globally-deployed software defined wan. In *Proc. ACM SIGCOMM 2013 Conference*, pp. 3–14, 2013.

[4] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *Proc. ACM SIGCOMM 2013 Conference*, pp. 15–26, 2013.

[5] A. Kumar, S. Jain et al. Bwe: Flexible, hierarchical bandwidth allocation for WAN distributed computing. In *Proc. of ACM SIGCOMM 2015 Conference*, pp. 1–14, 2015.

[6] L. Luo, H. Yu, Z. Ye, and X. Du. Online deadline-aware bulk transfer over inter-datacenter wans. In *Proc. IEEE INFOCOM 2018 Conference*, pp. 630–638, 2018.

[7] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and Ming Zhang. Guaranteeing deadlines for inter-data center transfers. *IEEE/ACM Trans. Netw.*, vol. 25 no. 1, pp. 579–595, 2017.

[8] M. Noormohammadpour, C. S. Raghavendra, and S. Rao. Dcroute: Speeding up inter-datacenter traffic allocation while guaranteeing deadlines. In *Proc. IEEE HiPC 2016 Confernce*, pp. 82–90, 2016.

[9] M. Chowdhury, Z. Liu, A. Ghodsi, and I. Stoica. HUG: multi-resource fairness for correlated and elastic demands. In *Proc. USENIX NSDI 2016 Conference*, pp. 407–424, 2016.

[10] J. Guo, F. Liu, T. Wang, and J. C. S. Lui. Pricing intra-datacenter networks with over-committed bandwidth guarantee. In *Proc. USENIX ATC 2017 Conference*, pp. 69–81, 2017.

[11] L. Popa, G. Kumar, M. Chowdhury, A. Krishnamurthy, S. Ratnasamy, and I. Stoica. Faircloud: sharing the network in cloud computing. In *Proc. ACM SIGCOMM 2012 Conference*, pp. 187–198, 2012.

[12] E. Danna, A. Hassidim, H. Kaplan, A. Kumar, Y. Mansour, D. Raz, and M. Segalov. Upward max min fairness. In *Proc. IEEE INFOCOM 2012 Conference*, pp. 837–845, 2012.

[13] M. Azarafrooz, R. Chandramouli, and K.P. Subbalakshmi. Reciprocity, fairness and learning in medium access control games. *Elsiver Computer Communications*, vol. 46 pp.22–28, 2014.

[14] W. B. Arthur. Inductive reasoning and bounded rationality. *American Economic Review*, vol. 84, no. 2, pp. 406–411, 1994.

[15] D. Whitehead. The el farol bar problem revisited: Reinforcement learning in a potential game. *Ese Discussion Papers*, no. 4, pp. 561–562, 2008.

[16] M. Rabin. Incorporating fairness into game theory and economics. *The American economic review*, pp. 1281–1302, 1993.

[17] D. Yang, G. Xue, X. Fang, S. Misra, and J. Zhang. A game-theoretic approach to stable routing in max-min fair networks. *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1947–1959, 2013.

[18] A. Saeed, N. Dukkipati, V. Valancius, V. T. Lam, C. Contavalli, and A. Vahdat. Carousel: Scalable traffic shaping at end hosts. In *Proc. ACM SIGCOMM 2017 Conference*, pp. 404–417, 2017.

[19] L. Chen, K. Chen, W. Bai, and M. Alizadeh. Scheduling mix-flows in commodity datacenters with karuna. In *Proc. ACM SIGCOMM 2016 Conference*, pp. 174–187, 2016.

[20] Wenxin Li, X. Zhou, K. Li, H. Qi and D. Guo TrafficShaper: Shaping Inter-Datacenter Traffic to Reduce the Transmission Cost. *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1193–1206, 2018.

[21] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula. Calendaring for wide area networks. In *Proc. ACM SIGCOMM 2014 Conference*, pp. 515–526, 2014.

[22] N. Laoutaris, M. Sirivianos, X. Yang, and P. Rodriguez. Inter-datacenter bulk transfers with netstitcher. In *Proc. ACM SIGCOMM 2011 Conference*, pp. 74–85, 2011.

[23] B. Spang, A. Sabnis et al. MON: mission-optimized overlay networks. In *Proc. IEEE INFOCOM 2017 Conference*, pp. 1–9, 2017.

[24] M. Noormohammadpour, C. Raghavendra, S. Kandula, and S. Rao. Quickcast: Fast and efficient inter-datacenter transfers using forwarding tree cohorts. In *Proc. IEEE INFOCOM 2018 Conference*, pp. 225–233, 2018.