Bulk Transfer Scheduling with Deadline in Best-Effort SD-WANs

Arshia Hosseini
University of Calgary
seyedarshia.hosseini@ucalgary.ca

Mahdi Dolati
University of Tehran
mahdidolati@ut.ac.ir

Majid Ghaderi
University of Calgary
mghaderi@ucalgary.ca

***Abstract*—Many cloud providers have multiple geo-distributed inter-connected datacenters around the globe. These datacenters are increasingly being inter-connected using software-defined WANs (SD-WANs), which extend the capabilities of SDN architecture to wide-area networks. While conventional MPLS tunneling has proven to be a practical approach for inter-connecting datacenters, such tunnels have a static nature and incur substantial maintenance costs. Given the centralized control and programmability of SDN, it is possible to utilize multiple Internet tunnels to provide a low-cost alternative to MPLS tunnels in SD-WANs. However, the best-effort nature of Internet tunnels means that they undergo capacity fluctuations throughout the day, making it difficult to provide any service guarantees such as completion time for inter-datacenter transmissions. In this paper, we consider the problem of scheduling bulk transfer requests with deadline in a best-effort SD-WAN. We propose an approximate scheduling algorithm called xBESD which utilizes tunnel capacity estimations to design a robust transfer schedule that maximizes a cloud provider's profit by transmitting bulk transfers before their deadlines. We analyze xBESD and show that it attains an approximation ratio that only depends on the number of overlapping requests that have the same profit to bandwidth ratio. Furthermore, we provide extensive simulation as well as realistic Mininet experimental results to assess the performance of xBESD in a variety of network scenarios. Our results show that xBESD improves the provider's profit by approximately 60% on average compared to other baseline scheduling methods, in addition to cutting down the Internet service provider costs.**

I. Introduction

Cloud service providers rely on multiple geo-distributed datacenters (DCs) to provide online services more efficiently to users across the globe [1]–[4]. These service providers, however, have to spend a substantial amount of money on the inter-DC wide area network (WAN) that connects their DCs [2]. Multi-Protocol Label Switching (MPLS) has been the dominant transport technology over WANs. However, Software-Defined Wide Area Networks (SD-WANs) have been recently utilized by companies such as Google, Microsoft, and Facebook, as an alternative to MPLS to overcome some of its shortcomings [1], [2] such as poor efficiency and lack of a global view. SD-WAN extends Software Defined Networking (SDN) concepts to WANs. In SDN, the control plane is decoupled from the data plane and moved to a logically centralized controller, enabling a global view of the network and consequently efficient centralized control and management using unified and open programming APIs.

978-3-903176-32-4 © 2021 IFIP

A common approach to connect geo-distributed DCs through multiple paths (for the purpose of reliability and traffic engineering), is leasing dedicated WAN lines that provide reliable transmission of data between the DCs. However, the cost of establishing and maintaining leased lines can be quite significant [2]. A cheaper alternative is tunnel-based forwarding over the Internet best-effort tunnels. We argue that, by utilizing SDN and best-effort Internet tunnels, SD-WAN can provide network performance similar to that of MPLS. In this approach, several IP tunnels are established between the DCs, where each might correspond to a different Internet Service Provider (ISP). This is also referred to as *multi homing* in the literature [5]. This approach, however, faces a challenge. The capacities of Internet best-effort tunnels fluctuate over time, making the instantaneous measurements valid only for a limited time. Thus, when scheduling bulk transfers that take a long time to finish, we have to take this uncertainty into consideration, which complicates the problem. However, with proper scheduling, it is possible to account for capacity fluctuations in order to avoid overloading these tunnels or losing traffic data as much as possible. In order to address uncertainty and fluctuations, stochastic programming and robust optimization have been used in the literature [6], [7]. The former approach requires knowledge of the distribution of pertinent random variables - link capacities here - which is hardly practical. Robust optimization, on the other hand, relies on the characterization of the range of fluctuations, which can be estimated for short periods using prediction techniques based on statistical analysis and machine learning [8], [9].

Considering different characteristics, requirements, and associated utility of traffic that goes over inter-DC WANs helps service providers better manage limited WAN resources and therefore maximize their revenue. Inter-DC traffic can be classified as interactive, elastic, or background traffic [2]. Interactive traffic has the highest priority and must be transmitted upon arrival. Elastic traffic is composed of bulk transfers that usually have long deadlines. Finally, background traffic has no deadline and can be sent in a best-effort fashion without any service guarantees. While interactive traffic involves only $5-15\%$ of WAN traffic, bulk transfers constitute a considerable portion of it [2]. A cloud service provider's profit comes from completing user data transfer requests according to agreed-upon SLAs. For example, in the case of bulk elastic transfers, the SLA may require that the entire data transfer

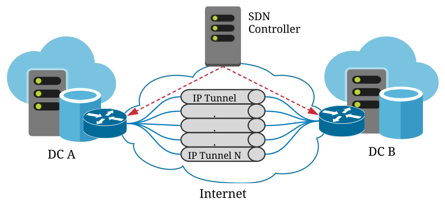


Fig. 1: Inter-Datacenter SD-WAN architecture.

be completed before a deadline. Given the limited WAN resources, it is then crucial for service providers to efficiently schedule bulk transfers so as to guarantee their deadlines.

Even though previous works [3], [4], [6], [10], [11] consider demand fluctuations for high-priority traffic (*i.e.*, interactive traffic), none of them consider using unreliable resources - such as the best-effort Internet tunnels - with fluctuating capacities to reduce operational costs associated with MPLS. In such an environment, scheduling of transfers without considering the fluctuations results in overloading of the tunnels, congestion, and missed deadlines. Our objective is to design a scheduling algorithm that considers the estimated capacity fluctuations of the tunnels and proactively schedules bulk transfers to minimize missed deadlines and thus maximize the profit of the cloud service provider.

In this paper, we present BESD, a **B**ulk Transfer Scheduling with D**E**adline over Best-Effort **SD**-WANs , which maximizes gained profit for bulk transfer requests over best-effort tunnels with uncertain bandwidth. To this end, bulk transfer requests are managed by a centralized controller that has information about the current state of the network, as well as estimates of its future states. Using this information, the controller schedules bulk transfers over space (*i.e.*, tunnels) and time, and enforces the decisions on the network. Fig. 1 shows the architecture of the proposed inter-DC SD-WAN.

Our main contributions can be summarized as follows:

- We formulate the scheduling of bulk transfers with deadlines in SD-WAN over Internet tunnels with time-varying capacity using robust optimization as a mixed-integer linear program (MILP), which can be solved for small problem instances.
- In order to reduce the computational complexity of the problem, we design an approximate algorithm called xBESD, which attains an approximation ratio that only depends on the number of overlapping requests that have the same utility-to-volume ratio.
- We present extensive comparative simulation results to study the behavior of our algorithms in numerous scenarios and demonstrate their gained profit and acceptance ratio.
- We also present Mininet experiments to demonstrate the performances of xBESD in a realistic network environment. Our results show that our robust formulation achieves up to twice as much profit as the other baseline algorithms from successfully transmitting bulk transfers, and reduces the cost paid to ISPs for the networking resources.

Paper Organization. The paper is organized as follows. In Section II, we review the related works. The system model and formal problem definition, respectively, are presented in Sections III and IV. In Section V, we propose our algorithm and its theoretical analysis. The experimental results are presented in Section VI. Section VII concludes the paper.

II. Related Work

In this section, we review relevant works that have focused on SD-WANs that connect geo-distributed sites.

Fixed Provisioning. This category includes works that consider paths with a guaranteed fixed capacity and provisioning. In the majority of works, interactive traffic demand is predictable in short durations for which a proportion of the available bandwidth is reserved to guarantee transfers. Microsoft and Google have proposed SWAN and B4 solutions for their Inter-DC networks [1], [2] in order to maximize their network's utilization, however, they only consider deadline-agnostic flows. Other works have been proposed that are deadline-aware [3], [4], [6], [10], [12]–[14]. Tempus [3] brings fairness to the flows by maximizing the minimum transfer rate between them. However, since there is no admission control in this approach, some admitted flows may lose their deadline. PGA [10], on the other hand, admits flows only if it can guarantee meeting their deadlines. All these works assume a stable network capacity with no fluctuations. Authors in [12] assume that there is a fixed prediction error for realtime traffic. If the realtime traffic volume exceeds the predicted value, they delay the bulk transfers that have farther deadlines. Amoeba [4] handles capacity uncertainty by reserving a headroom, which is scaled with time to handle extra uncertainty in the distant future. None of these works can flexibly characterize capacity uncertainty to achieve the desired trade-off between resource utilization and deadline guarantee. Works [13]–[16] either lack an admission control mechanism and assume that all capacities are known in advance, or fail to account for capacity fluctuations. Work [6] uses the Robust Optimization technique to design a proactive method that accounts for the uncertain demand of realtime transfers, however, they assume that path capacities are fixed and known in advance.

Dynamic Provisioning. In this group of works, although there is a fixed maximum capacity, the provisioning is usage-based in which the cost of transmission depends on the scheduling of transmitted data. The majority of works focus on minimizing transmission cost [7], [11], [17]–[20]. Postcard [17] uses the store-and-forward mechanism in the intermediate nodes to save traffic in peak hours to be sent in non-peak hours. TrafficShaper [7] utilizes the q-percentile pricing scheme in a decentralized fashion to send as much traffic as possible in the free burstable timeslots. The Pretium framework [11] provides a dynamic pricing scheme and focuses on social welfare as its objective. Jetway [18] proposes a solution for video flows in inter-DC networks based on historical information and does not consider information about the future. It admits as many flows as possible using current resources, then acquires more resources to admit the remaining flows. Work [19] uses the same approach, however, it is proposed for deadline-aware bulk transfers. Yang *et al.* in [21] attempt to minimize the bandwidth cost while guaranteeing transfer deadlines. In this work, the available bandwidth is dynamically allocated to the transfers and there are no limitations to the bandwidth capacity. Work [22] proposes a dynamic traffic management solution for inter-cloud SD-WANs in order to reduce inter-domain transit traffic costs. Even though some of these works consider predictable fluctuations of interactive traffic, they fail to address capacity uncertainty in the network. The recent survey in [23] acknowledges this gap as an open problem and presents further discussion regarding its importance.

III. System Model and Assumptions

In this section, we explain our proposed model for the investigation of deadline-aware bulk data transfer between two DCs over a non-dedicated network (*i.e.*, Internet). Similar to the state-of-the-art [3], [6], [10], we divide the time into timeslots, where resource allocation and scheduling decisions are made at the beginning of each timeslot and remain unchanged until the end of the timeslot. In the following, we explain the demand and network models considered in our work.

A. Demand Model

We focus on bulk transfers that exchange large data volumes and are subject to delivery deadlines (*e.g.*, database backups). Previous studies found out that these transfers constitute the burdensome part of communication between geo-separated DCs. We consider the batch model which is widely adopted in the literature [3], [6], [10]. In this model, a batch of bulk transfer requests, denoted by $\mathcal{R}$, becomes available in the origin DC. Each transfer request should be explicitly accepted or rejected. Transfer $r \in \mathcal{R}$, upon acceptance, must be able to transmit B_r bytes of data between timeslots τ_r^1 and τ_r^2 to the destination DC. Specifically, τ_r^1 is the earliest timeslot that transfer r can start its transmission and τ_r^2 is its delivery deadline. If the entirety of data is transferred before the deadline, the WAN provider receives U_r units of profit, otherwise, the profit is assumed to be zero. Furthermore, we let $\mathcal{P}_r$ denote the set of tunnels that request r is allowed to use (for security or accounting reasons, a request may not use some of the available tunnels). Each DC, upon receiving a batch request, should only accept a subset of requests that can be fully served and explicitly reject the others.

B. Network Model

We assume that there is a set of distinct tunnels, denoted by $\mathcal{P}$, between two DCs. For instance, when the interconnection network is the Internet, the tunnels work in a best-effort fashion, meaning that the *exact* amount of tunnel capacities are not known a priori. With no information about capacities, it is impossible to guarantee the deadline of accepted requests. Thus, we assume that the capacity of tunnel $p \in \mathcal{P}$ in timeslot t fluctuates in an interval given by,

$$[\overline{C}_p(t) - \widetilde{C}_p(t), \overline{C}_p(t) + \widetilde{C}_p(t)], \tag{1}$$

where, $\overline{C}_p(t)$ is the expected capacity and $\widetilde{C}_p(t) = \delta\,\overline{C}_p(t)$, for $0 \leq \delta \leq 1$, is the maximum *capacity fluctuation*. Previous

TABLE I: Important Mathematical Notations

Inputs	Definition
$\mathcal{R}$	Set of all requests
$\mathcal{P}$	Set of all paths
B_r	Demand (volume) of request r
U_r	Profit gained by successfully transmitting request r
$\overline{C}_p$	Average capacity of path p at timeslot t
$\widetilde{C}_p(t)$	Deviation of path p's capacity from its average
Γ	Maximum number of paths that deviate from their maximum capacity
δ	The Ratio of maximum capacity to the average capacity
τ_r^1	Arrival time of request r
τ_r^2	Deadline of request r
Variable	**Definition**
$x_p^r(t)$	Transmission volume of request r at timeslot t over path p
a_r	Admission status of request r

studies show that it is feasible to obtain these intervals from historical records with prediction methods [9]. Since the probability of all the tunnels hitting their lowest capacity (*i.e.*, $\overline{C}_p(t) - \widetilde{C}_p(t)$) in the same timeslot is low, we assume that in each timeslot at most $\Gamma \leq P = |\mathcal{P}|$ tunnels will exhibit the worst-case fluctuation. Observe that higher values of Γ model an unreliable environment that requires a considerable accommodation in term of headroom reservation and accepting fewer requests to guarantee the deadlines. On the other hand, lower values of Γ show a stable environment that allows us to accept more requests and utilize the bandwidth without missing deadlines.

IV. Problem Formulation

In this section, we formally define the problem of admission control and bandwidth allocation of bulk transfer requests. Specifically, we aim at admitting a set of requests and specify their transmission rates on the available tunnels such that their deadlines are guaranteed, and their sum of profits is maximized. To control the complexity of the constructed model, we use linear functions throughout the problem formulation, which facilitates obtaining efficient solutions. Table I provides a list of employed notations and their definitions.

A. LP Formulation

In the following, we elaborate on different parts of the formulation, which consists of deriving the objective, scheduling of bulk transfers, and enforcing capacity constraints.

System Profit. The objective is to maximize the profit that is gained from the accepted requests:

$$\text{Max.} \sum_{r \in \mathcal{R}} a_r \times U_r, \tag{2}$$

where, a_r is a decision variable to indicate whether request r is accepted or not.

Scheduling Bulk Transfers. We define decision variables $x_p^r(t)$ to compute the *transmission rate* of request r over tunnel p in timeslot t. The following constraints ensure that each request only uses permitted tunnels:

$$x_p^r(t) \geq 0, \quad \forall p \in \mathcal{P}_r, t, r \tag{3}$$

$$x_p^r(t) = 0. \quad \forall p \notin \mathcal{P}_r, t, r \tag{4}$$

Tunnel Capacity Constraints. In each timeslot t, the total transmission rate on each tunnel p should be less than or equal to its expected capacity,

$$\sum_{r \in \mathcal{R}} x_p^r(t) \leq \overline{C}_p(t). \quad \forall t, p \tag{5}$$

Furthermore, the total transmission on all tunnels should be less than or equal to the available bandwidth in each timeslot, regardless of which tunnels experience a capacity deficit compared to the estimated values. To this end, we use the following constraint,

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} x_p^r(t) \leq \sum_{p \in \mathcal{P}} \overline{C}_p(t) - \max_{\substack{\pi(t) \subseteq \mathcal{P}, \\ |\pi(t)| \leq \Gamma}} \sum_{p \in \pi(t)} \widetilde{C}_p(t) \quad \forall t. \tag{6}$$

In this equation, $\pi(t)$ is the set of tunnels that hit their lowest capacity in timeslot t. In this notation, $\Gamma = 0$ is a scenario with perfectly accurate estimations and $\Gamma = P$ is the worst-case scenario where all capacities deviate maximally from their average at every time-slot. Note that we do not know the value of $\pi(t)$ at the moment of admission control and bandwidth allocation. Rather, we use the optimization process (*i.e.*, the max operator) to fill it with tunnels that lead to the worst situation. Therefore, constraint (6) allows us to be prepared for the worst case scenario that might prevail within the boundary of our assumptions about the accuracy of estimated values. The max operator, however, is non-linear and complicates the optimization process. To linearize constraint (6), similar to [24], we extract the non-linear term and write it as a separate program. As a result, we have:

$$\text{Max.} \sum_{p \in \mathcal{P}} z_p(t) \times \widetilde{C}_p(t), \tag{7}$$

$$\text{s.t.} \sum_{p \in \mathcal{P}} z_p(t) \leq \Gamma, \tag{7a}$$

$$0 \leq z_p(t) \leq 1, \tag{7b}$$

where, $z_p(t)$ is a decision variable that indicates whether tunnel p deviates maximally in timeslot t and Γ restricts the number of fluctuating tunnels. Then, we calculate the dual of the linear program (7), by defining two dual variables λ_t and ν_t^p, respectively, associated with constraints (7a) and (7b). The dual linear program is as follows:

$$\text{Min.} \ \lambda_p \times \Gamma + \sum_{p \in \mathcal{P}} \nu_p^t, \tag{8}$$

$$\text{s.t.} \ \widetilde{C}_p(t) \leq \lambda_p + \nu_p^t, \tag{8a}$$

$$0 \leq \lambda_p, 0 \leq \nu_p^t. \tag{8b}$$

Next, we can replace the non-linear term of constraint (6) with the objective of (8) and include constraints (8a) and (8b) in the original problem formulation. Specifically, constraint (6) is replaced with the following constraints:

$$\sum_{r \in \mathcal{R}} \sum_{p \in \mathcal{P}_r} x_p^r(t) \leq \sum_{p \in \mathcal{P}} \overline{C}_p(t) - \lambda_p \times \Gamma - \sum_{p} \nu_p^t \tag{9}$$

$$\widetilde{C}_p(t) \leq \lambda_p + \nu_p^t, \tag{10}$$

$$0 \leq \lambda_p, 0 \leq \nu_p^t. \tag{11}$$

Deadline Guarantee. For each accepted request r, the accumulated data transmission between its arrival time τ_r^1 and its deadline τ_r^2 should be equal to B_r, the demand of request r. This constraint is formulated as,

$$a_r \times B_r \leq \sum_{t \in [\tau_r^1, \tau_r^2]} \sum_{p \in \mathcal{P}_r} x_p^r(t) \times \Theta, \quad \forall r \tag{12}$$

where, Θ is the duration of a timeslot in seconds. We use Θ to compute the volume of transmitted data (*e.g.*, in bits) from the allocated transmission rates (*e.g.*, in bits/second).

Objective (2) along with constraints (3)-(5) and (9)-(12) formally define the problem of **B**ulk Transfer Scheduling with **DE**adline over Best-Effort **SD**-WANs (BESD).

V. Scheduling Algorithm

Integer variables in BESD make its exact solution computationally hard and time-consuming, which affects its applicability in practice. To alleviate this problem, we develop a polynomial-time approximation solution based on the *iterative rounding* technique. We call our proposed algorithm xBESD that stands for appro**x**imate BESD. In xBESD, we relax the problem by removing the integrality constraint of a_r variables. This transformation allows us to obtain an optimal *fractional* solution in polynomial time. Then, we iteratively select a variable and round it to either 0 or 1 until all integrality constraints are satisfied. Specifically, in each iteration, among the requests with the highest value of profit-per-volume (defined as $\rho_r = \frac{U_r}{B_r}$), xBESD selects request r with the highest value of B_r and if the network capacity allows, rounds a_r to 1. Theorem 2 proves that this method of selecting requests guarantees an approximation ratio that depends on the number of overlapping requests with an equal value of profit-per-volume. In the following, we explain xBESD in detail with reference to its pseudo-code in Algorithm 1. xBESD starts by constructing the BESD problem and relaxing its integrality constraints in line 2. Then, two empty lists, $\mathcal{A}$ and $\mathcal{J}$, are created to store the accepted and rejected requests, respectively. In line 5, the relaxed model $\widetilde{M}$ is solved to obtain the optimal fractional values for decision variables $\widetilde{a}_r$ (we use tilde to denote relaxed variables). The while loop in lines 6 to 24, then, iteratively computes a feasible 0-1 value assignment for $\widetilde{a}_r$. Each iteration of the while loop starts by accepting all requests whose $\widetilde{a}_r$ variables are equal to one. This happens by setting a lower-bound of 1 for their corresponding decision variables to fix their values in the optimization model and storing them in list $\mathcal{A}$ (see lines 7 to 9). Similarly, in lines 10 to 12, all requests with $\widetilde{a}_r = 0$ are rejected. Then, in line 14, the request with the highest value of ρ_r and then B_r that is not accepted or rejected in current or previous iterations is selected and stored in $r^\star$. The algorithm attempts to accept $r^\star$ by setting the lower-bound of 1 for $\widetilde{a}_{r^\star}$ and tests the feasibility of this decision by solving the model in line 17. If the model becomes infeasible, the lower and upper-bounds of $\widetilde{a}_{r^\star}$ are set to 0 and $r^\star$ is added to $\mathcal{J}$. Moreover, the model is solved one more time to release the partial resources that are allocated to $r^\star$ and assign them to other requests. However, if the model becomes

Algorithm 1: xBESD – Approximate BESD

```
   procedure xBESD()
1      $M \leftarrow$ MIP()                                  /* create integer model */
2      $\widetilde{M} \leftarrow$ relax($M$)
3      $\mathcal{A} \leftarrow \{\}$                           /* accepted requests */
4      $\mathcal{J} \leftarrow \{\}$                           /* rejected requests */
5      $status, \{\widetilde{a}_r\}, \{x_p^r(t)\} \leftarrow$ solve($\widetilde{M}$)
6      while True do
7          foreach $r \in \mathcal{R} - \mathcal{A} \cup \mathcal{J}$ and $\widetilde{a}_r = 1$ do
8              $\widetilde{a}_r$.lower_bound $\leftarrow 1$    /* fix the decision variable to 1 */
9              $\mathcal{A}$.append($r$)
10         foreach $r \in \mathcal{R} - \mathcal{A} \cup \mathcal{J}$ and $\widetilde{a}_r = 0$ do
11             $\widetilde{a}_r$.upper_bound $\leftarrow 0$    /* fix the decision variable to 0 */
12             $\mathcal{J}$.append($r$)
13         if $\mathcal{A} \cup \mathcal{J} \neq \mathcal{R}$ then
14             $r^\star \leftarrow \arg\max_{r \in \mathcal{R} - \mathcal{A} \cup \mathcal{J}} \{\rho_r, B_r\}$
15             $\widetilde{a}_{r^\star}$.lower_bound $\leftarrow 1$
16             $status, \{\widetilde{a}_r\}, \{x_p^r(t)\} \leftarrow$ solve($\widetilde{M}$)
17             if $status$ = INFEASIBLE then
18                 $\widetilde{a}_{r^\star}$.lower_bound $\leftarrow \widetilde{a}_{r^\star}$.upper_bound $\leftarrow 0$
19                 $\mathcal{J}$.append($r^\star$)
20                 $status, \{\widetilde{a}_r\}, \{x_p^r(t)\} \leftarrow$ solve($\widetilde{M}$)
21             else
22                 $\mathcal{A}$.append($r^\star$)
23         else
24             return $\{\widetilde{a}_r\}, \{x_p^r(t)\}$
```

feasible, it is sufficient to only add $r^\star$ to $\mathcal{A}$. Finally, when all the requests are either accepted or rejected, the acceptance decision variables $\widetilde{a}_r$ and bandwidth allocation variables $x_p^r(t)$ are returned in line 24.

Theorem 1. xBESD *runs in* $O(|\mathcal{R}| \times (|\mathcal{R}||\mathcal{P}|T)^{3.5})$.

Proof. In each iteration of the while loop at least one request is added either to the rejected requests (in line 19) or to the accepted requests (in line 22). Thus, the while loop terminates after at most $|\mathcal{R}|$ iterations. Furthermore, in each iteration at most 2 linear programs with $O(|\mathcal{R}| \times |\mathcal{P}| \times T)$ decision variables are solved, where T is the length of the data transmission period in time-slot that starts from the smallest τ_r^1 and ends with the largest τ_r^2. In the worst case, an interior point method solves a general linear program, such as our formulation, in $O(n^{3.5})$ [25], where n is the number of decision variables, which completes the proof. ☐

Theorem 2. *Let* Ψ *be the maximum number of requests that overlap a timeslot and their* ρ_r *is equal. The approximation ratio of the algorithm is at most* $\Psi - 1$.

Proof. During the rounding procedure, all the requests with $\widetilde{a}_r = 1$ are admitted and all the requests with $\widetilde{a}_r = 0$ are rejected. Thus, in the rest of the proof we assume that $\widetilde{a}_r \in (0, 1)$. In this situation, we have to accept a subset of requests (*i.e.*, round respective $\widetilde{a}_r$ to 1) by rejecting others (*i.e.*, round respective $\widetilde{a}_r$ to 0). Specifically, to accept r_1 with $\widetilde{a}_{r_1} \in (0, 1)$, it is necessary to reject a set of requests $\overline{\mathcal{R}}_{r_1}$, such that their total network consumption in the $[\tau_{r_1}^1, \tau_{r_1}^2]$ interval is at least $(1 - a_{r_1})B_{r_1}$ (see the shaded area in Fig. 2). Note that request rejection only affects the profit gained by the provider (*e.g.*, results in profit loss). If $\overline{\mathcal{R}}_{r_1}$ is empty, there is no way to accept request r_1. Thus, we set $a_{r_1} = 0$ and solve the fractional solution again, which does not change the gap between the rounded solution and the optimal. Note that the size of $\overline{\mathcal{R}}_{r_1}$ is at most $\Psi - 1$.

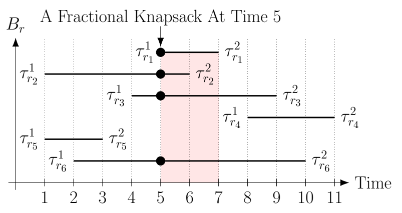


Fig. 2: Effect of requests on bandwidth allocation of each other

Notice that since we know that none of the requests are complete, it is particularly possible to take bandwidth from one request and give it to another one. Observe from Fig. 2 that in each timeslot the relaxed problem is similar to a fractional knapsack, where each request that has a higher profit per volume (*i.e.*, $\rho_r = U_r/B_r$) gets more bandwidth. Furthermore, for all requests that have received bandwidth in a time-slot, but *none of them are complete*, ρ_r is equal. Otherwise, it is possible to take bandwidth from requests with lower values of ρ_r and allocate it to the requests with higher values of ρ_r and increase the objective of the *relaxed* optimization program, which is impossible because it is solved optimally. It is worth mentioning that the optimum amount of allocated bandwidth to each request is not known a priori and is computed by the linear program solver based on the values of Γ and δ, and constraint (6). Thus, we cannot directly use fractional knapsack algorithms to solve our problem. Nevertheless, by accepting r_1, the rounded solution gains U_{r_1}, while the optimal solution can gain, at most, $\sum_{r' \in \overline{\mathcal{R}}_{r_1}} U_{r'}$. Thus, to characterize the performance we have to bound the ratio $\frac{\sum_{r' \in \overline{\mathcal{R}}_{r_1}} U_{r'}}{U_{r_1}}$. Let r'' be the request in $\overline{\mathcal{R}}_{r_1}$ with the maximum $B_{r''}$. An upper bound for the approximation ratio is

$$\frac{\sum_{r' \in \overline{\mathcal{R}}_{r_1}} \rho_{r'} B_{r'}}{\rho_r B_{r_1}} \leq \frac{\sum_{r' \in \overline{\mathcal{R}}_{r_1}} B_{r''}}{B_{r_1}} = (\Psi - 1)\frac{B_{r''}}{B_{r_1}} \quad (13)$$

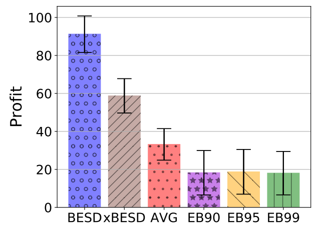
Thus, performing the rounding in the descending order of transfer volume sizes (*i.e.*, $B_{r''} \leq B_{r_1}$) guarantees the inequality $\frac{B_{r''}}{B_{r_1}} \leq 1$, which completes the proof. ☐

VI. Performance Evaluation

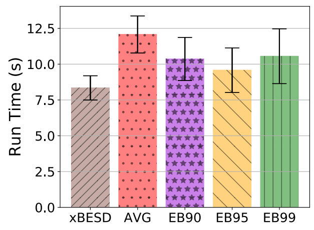
Methodology. We present two sets of evaluation results, namely simulations and Mininet experiments. We used the acceptance rate of transfers and the total profit as our performance metrics. The simulation studies focus on benchmarking the performance of both exact and approximate algorithms (*i.e.*, BESD and xBESD), while Mininet experiments focus on characterizing the performance of our approximate algorithm (*i.e.*, xBESD) in a realistic network environment.

Algorithms. State-of-the-art methods are **reactive** with respect to capacity fluctuations, and they are orthogonal to our **proactive** algorithm. Thus, we used two flexible baselines to show the advantages of uncertainty-aware scheduling.

1) **Average Algorithm:** This algorithm, denoted by AVG, schedules requests based on predicted average capacities and does not consider capacity fluctuations.
2) **Effective Bandwidth Algorithm:** The **E**ffective **B**andwidth algorithm, denoted by EB, calculates



(a) Profit.



(b) Run-time.

Fig. 3: Benchmark of different algorithms.

an effective bandwidth for each tunnel based on predicted average capacities as well as maximum capacity fluctuations. This algorithm is implemented using the algorithm in [26], with three percentiles of 90, 95, and 99 using the Gaussian approximation algorithm.

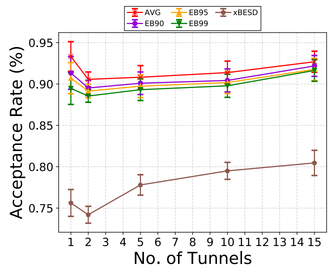
A. Simulation Experiments

Setup. We use a source/destination topology in our evaluations. In different experiments, we evaluate the effect of different values for various parameters whose default values are as follows. Each experiment is repeated 33 times and a 95% confidence interval is used to present the results. There are 10 tunnels that are end-to-end IP tunnels with time-varying capacities, all of which are available to every bulk transfer request. The average capacities are chosen uniformly in the range $[50, 200]$ Mbps. The start times of bulk transfer requests are modeled as a Poisson process with an arrival rate of λ requests per second. We set λ to be 4. Transfer volume B_r is an exponentially distributed random variable with the mean 10 GBytes. Furthermore, the deadline τ_r^2 is also an exponentially distributed random variable whose mean is 10 timeslots from the starting time τ_r^1. We also generate a utility for each request uniformly chosen in the range $[1, 10]$. The overall data transmission period is fixed to be 50 timeslots, each of which lasts for 3 minutes. We set δ to be 40%, and we let the 7 tunnels with the lowest capacities fluctuate (*i.e.*, $\Gamma = 7$) at every timeslot in order to account for the worst-case scenario. The results of our measurements indicate that a range of 15% to 20% variation is typical. However, capacity variations can be as high as 50% due to many underlying factors such as traffic shaping by ISPs, congestion control, and link outages. This is consistent with the state-of-the-art [9], [27], [28]. All simulations were implemented in Python 3 and conducted on a machine with an Intel® Core™ i7-8700 processor at 3.20 GHz and 8 GBs of RAM.

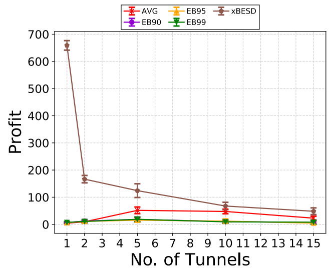
Metrics. We consider the following performance metrics:

- **Acceptance Rate:** The fraction of bulk transfers that are admitted for transmission.
- **Profit:** The total profit from successfully transmitting transfer requests before their deadlines.
- **Run-time:** The total time it takes to schedule all bulk transfers in a transmission period.

Results. We first evaluate the performance of our proposed algorithms under the general setup described at the beginning of this subsection. Then, we present a series of microbenchmarks where each one focuses on a specific aspect of

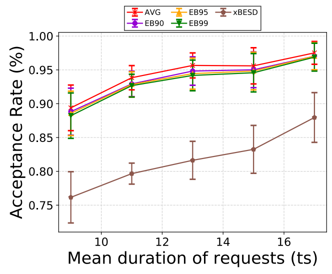


(a) Effect on acceptance rate.

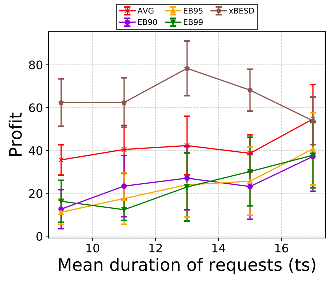


(b) Effect on profit.

Fig. 4: Effect of number of tunnels.



(a) Effect on acceptance rate.

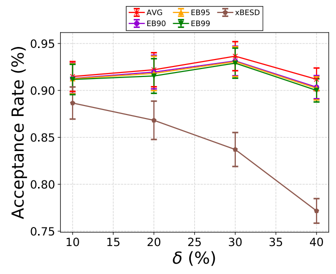


(b) Effect on profit.

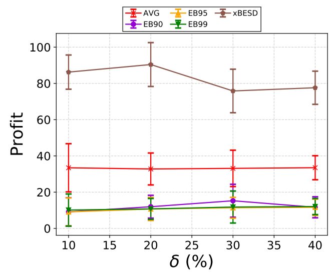
Fig. 5: Effect of request deadline.

the algorithms or environment to provide a broader insight into the performance of the algorithms. Consider Fig. 3 for a high-level comparison of all algorithms. Fig. 3(a) shows the total profits, which is the objective of the algorithms. Evidently, both our exact algorithm, BESD and our approximate algorithm, xBESD outperform all other algorithms in terms of profit. Furthermore, Fig. 3(b) demonstrates that while all algorithms have a run-time between 7 to 13 seconds, xBESD slightly outperforms other algorithms. Considering the frequency of running our algorithm (once every 50 timeslot), an approximate run-time of 8 seconds is reasonable and does not impose a burden on the performance of the system. Since the exact algorithm takes a long time to run, in the rest of the experiments we focus on the approximate algorithm.

Effect of Number of Tunnels. Figs. 4(a) and 4(b) demonstrate the benefits of our robust formulation for different numbers of tunnels with a fixed total network capacity. We set Γ to be 1, 2, 3, 7, and 10 for 1, 3, 5, 10, 15 number of tunnels. For one tunnel, our algorithm performs close to optimal since there is only one tunnel and our algorithm is aware of its fluctuations, while the other algorithms fail to successfully transmit any transfers before their deadline, therefore, do not gain any profit. By increasing the number of available tunnels, we notice a slight increase in the acceptance rate, however, the achieved profit of our algorithm drops. This is due to the reason that as the number of tunnels increase, the chance of mispredicting the fluctuating tunnels increases as well. Nonetheless, having fewer tunnels is not always possible in practice, as high-capacity tunnels are generally more expensive to purchase. Note that our algorithm only has partial information about the fluctuations and cannot fully predict them. For the baseline algorithms, we notice an increase in the profit when the number of tunnels goes from 1 to 2. This is anticipated because other algorithms no longer suffer from fluctuations all the time and can successfully transmit a number of requests. Afterward,

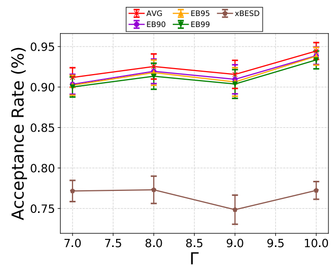


(a) Effect on acceptance rate.

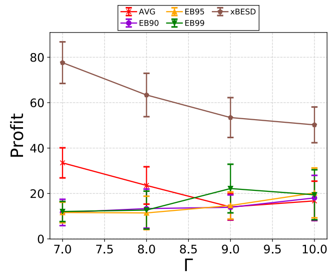


(b) Effect on profit.

Fig. 6: Effect of maximum capacity deviation from average.



(a) Effect on acceptance rate.



(b) Effect on profit.

Fig. 7: Effect of number of fluctuations per time window.

they follow a trend similar to our algorithm. In summary, our algorithm maintains a superior performance compared to other ones, which indicates its ability to handle fluctuations better.

Effect of Request Deadline. In Fig. 5(a) we can observe that the acceptance rate of all algorithms increases as the duration of requests increases. For the baseline algorithms that do not take fluctuations into consideration, this leads to an increase in the profit. This is due to less tight deadlines which allow more requests to complete successfully on the tunnels that do not fluctuate. For our algorithm, however, as the duration of requests increases, the probability of experiencing a misprediction increases, which leads to a higher risk of deadline violation and loss of profit. In other words, our algorithm is more likely to over-allocate the fluctuating tunnels and under-allocate the non-fluctuating ones when the deadlines are extended. Hence, profit has a downward trend for xBESD as shown in Fig. 5(b). Compared to AVG, which is the superior performing baseline algorithm in this scenario, xBESD provides a 62% improvement on average.

Effect of Capacity Fluctuations. In this scenario, we study the effect of maximum capacity deviation from the estimated values. We let δ be 10%, 20%, 30%, 40%, and 50% to demonstrate the effect of small and large fluctuations. In Fig. 6(a), we observe that while the baseline algorithms do not react to the severity of the fluctuations in the network, our algorithm admits fewer requests in such a situation. Furthermore, Fig. 6(b) shows that our algorithm outperforms other algorithms in terms of profit under different fluctuations.

Effect of Number of Fluctuations Per Timeslot. In this scenario, we study the effect of Γ (*i.e.*, number of tunnels whose capacities fluctuate in a timeslot) on the performance of our algorithm. We set the number of available tunnels to 10, and increase Γ from 7 to 10. The tunnels with capacity fluctuations are chosen randomly. Fig. 7(a) shows that our algorithm admits fewer requests when this number increases. In terms of profit, our algorithm outperforms the other algorithms significantly, which is shown in Fig. 7(b).

B. *Mininet Experiments*

We used Mininet [29] to emulate a realistic network setup. Then, we allow the available bandwidth change according to two different fluctuation models:

- **Tunnel Deviation Model:** In this model, Γ number of randomly chosen tunnels fluctuate maximally in each timeslot.
- **Time Deviation Model:** In this model, each tunnel fluctuates in Υ number of randomly chosen timeslots maximally over the course of the transmission period, T.

Setup. To carry out Mininet experiments, we set up an environment as follows:

- **Trials:** We conducted 5 trials, one for each of the algorithms discussed earlier. Each trial took 150 minutes (*i.e.*, 50 timeslots of 180 seconds) during which we generated traffic and monitored the network.
- **Topology:** We implemented a topology that consists of 12 switches and 2 hosts. Two of the switches that act as ingress/egress switches are each connected to a host and 10 transit switches, each of which creates a distinct tunnel between the two hosts. Fig. 8 demonstrates the topology. `OpenVSwitch2.13` is used to emulate `OpenFlow1.3` switches. Furthermore, we used the same parameters as in the simulations for the links. The capacity of each of the links was set using the `tc` command. Capacity fluctuations were introduced to the links by generating background traffic with transmission rates that are equal to the number of capacity fluctuations, and sending them over the tunnels. Moreover, each Mininet host is an interface in a distinct network namespace. We set one of the hosts as the sender and the other one as the receiver. The sender both generates the background traffic and the bulk transfers.
- **Controller:** We used an instance of the `ONOS` SDN controller developer edition, built from the source code. The controller receives from the solver the set of available tunnels for each request at every timeslot, and uses this information to install the relevant flows on the switches.
- **Multi-tunnel Routing:** The `multi-tunnel routing` module is a software module that we developed on top of the `ONOS` controller using the `ONOS Java API 1.13.1` in order to enable our algorithm to schedule each request over multiple tunnels in a network of `OVS` switches in Mininet. The module uses the `TopologyService` interface to find all the tunnels (*i.e.*, 10 tunnels here) between the ingress/egress switches.
- **Traffic Monitoring:** We implemented another custom module on the controller that utilizes `PortStatistics` interface in order to collect port utilization statistics periodically throughout the course of each experiment.
- **Traffic Generator:** We implemented a custom traffic generator that generates UDP traffic with a unique destination port number for each transfer request. While the sender runs

the traffic generator, a traffic server runs on the receiver in order to receive the transmitted traffic.

- **Environment:** All experiments were conducted on an Ubuntu 20.0 LTS VM on Amazon AWS with four vCPUs and 8 GBs of RAM.

Tunnel Deviation Model. During this experiment, we let 7 tunnels with the lowest capacities fluctuate in each time-slot and their capacities fall 40% below average, while the other 3 maintain a capacity close to the average estimations (*i.e.*, $\Gamma = 7$). We used the same parameters as the simulations to study the following metrics:

- **Link Utilization:** The ratio of the total amount of traffic over a link to the total link capacity.
- **Profit:** The total profit from successfully transmitting transfer requests before their deadlines.

The results are presented in Figs. 9(a) and 9(b). Compared to other algorithms, xBESD consumes up to 12% less bandwidth in most timeslots. Furthermore, the other algorithms achieve very similar link utilization. This is anticipated since the simulation results demonstrated that they have a similar acceptance rate (*i.e.*, they roughly admit the same requests). Similarly, the results of this experiment are consistent with those of the simulations with regards to the objective, *i.e.*, profit. As we can see, our algorithm achieves a higher profit compared to other algorithms. This is due to other algorithms over-allocating or under-allocating the links by not taking into account the fluctuations, thus, losing the majority of the packets. Particularly, the EB algorithm calculates a bandwidth that is somewhere between the average and the maximally deviated bandwidth. Consequently, it over-allocates the tunnels in the timeslots when they are fluctuating and under-allocates them when they are not fluctuating. Therefore, the problem is twofold as compared to the AVG algorithm, which only over-allocates the tunnels when their capacities fluctuate.

Moreover, the consistent lower utilization of xBESD can be even more beneficial, especially with a burstable or usage-based billing scheme [30], [31], which is very common for Internet tunnels. When using leased lines that have flat-rate pricing, a desirable objective is to achieve high utilization in order to efficiently use the resources for which we have already paid. However, when using Internet best-effort tunnels with usage-based pricing, low utilization can actually lead to saving money. In other words, we have to pay less for the traffic that does not gain any profit. In summary, our algorithm not only achieves more profit by successfully transmitting more bulk transfers but also cuts costs that have to be paid for the network services to the ISP.

Time Deviation Model. During this experiment, we let each tunnel fluctuate maximally 70% of the time (*i.e.*, $\Upsilon = 35$). Similar to the previous experiment, we study the link utilization and profit. Figs. 10(a) and 10(b) show that our algorithm's performance is superior to its counterparts under a different scenario where the capacity estimations are provided in a different fashion. Our algorithm consumes up to 15% less

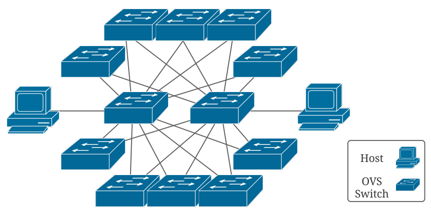
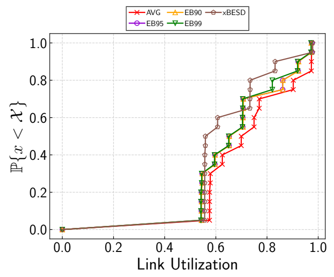
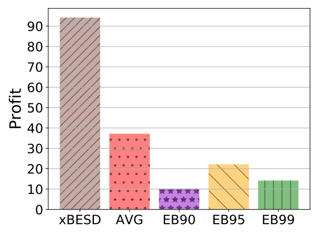


Fig. 8: Topology of the experimental network.

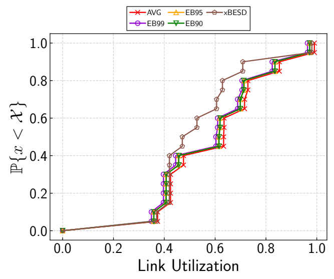


(a) CDF of link utilization.

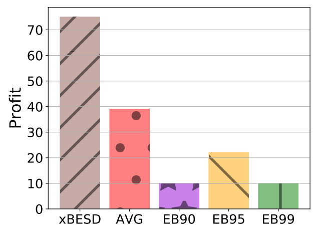


(b) Profit of different algorithms.

Fig. 9: tunnel deviation experimental results.



(a) CDF of link utilization.



(b) Profit of different algorithms.

Fig. 10: Time deviation experimental results.

bandwidth in most timeslots while achieving up to twice as much profit. Not taking the fluctuations into account has two disadvantages for the baseline algorithms. First, they admit too many requests and over-allocate the tunnels. Second, they can only transmit a request successfully if the tunnels on which the transfer is transmitted do not fluctuate during the whole time of the transmission. Consequently, considering the number of times that the tunnels fluctuate randomly in total, the chance of successfully transmitting a request is very low for the baseline algorithms, as is evident from the results.

VII. Conclusion

In this paper, we considered the problem of scheduling bulk transfer requests over the Internet best-effort tunnels with uncertain capacity. We formulated the problem as a robust optimization problem, where the exact capacity of every tunnel is unknown and we only have an estimated average, an estimated maximum fluctuation, and a maximum number of tunnels whose capacities fluctuate at the same timeslot. We showed that the problem can be well approximated, which can be solved in polynomial time. Using a combination of simulations and experimental trials, we studied the performance of our algorithm in different scenarios. Our results show that by properly scheduling bulk transfers over multiple best-effort tunnels, not only can we guarantee transfer deadlines, but we can also reduce resource consumption and gain more profit than the baseline algorithms.

References

[1] S. Jain, A. Kumar, S. Mandal, J. Ong *et al.*, “B4: Experience with a globally-deployed software defined WAN,” in *Proc. ACM SIGCOMM*, 2013, p. 3–14.

[2] C.-Y. Hong, S. Kandula, R. Mahajan *et al.*, “Achieving high utilization with software-driven WAN,” *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, p. 15–26, 2013.

[3] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, “Calendaring for wide area networks,” in *Proc. ACM SIGCOMM*, 2014, pp. 515–526.

[4] H. Zhang, K. Chen, W. Bai, D. Han *et al.*, “Guaranteeing Deadlines for Inter-Data Center Transfers,” *IEEE/ACM Trans. Netw.*, vol. 25, pp. 579–595, 2017.

[5] D. K. Goldenberg, L. Qiuy, H. Xie *et al.*, “Optimizing cost and performance for multihoming,” *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, pp. 79–92, 2004.

[6] M. Dolati, M. Ghaderi, and A. Khonsari, “Proactive inter-datacenter multicast with realtime and bulk transfers,” in *Proc. IEEE/ACM IWQoS*, 2019, pp. 1–10.

[7] W. 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.

[8] N. Krishnaswamy, M. Kiran, K. Singh, and B. Mohammed, “Data-driven learning to predict WAN network traffic,” in *Proc. ACM SNTA*, 2020, p. 11–18.

[9] S. K. Khangura, M. Fidler, and B. Rosenhahn, “Neural Networks for Measurement-based Bandwidth Estimation,” in *Proc. IFIP Networking*, 2018, pp. 1–9.

[10] L. Luo, H. Yu, Z. Ye, and X. Du, “Online Deadline-Aware Bulk Transfer Over Inter-Datacenter WANs,” in *Proc. IEEE INFOCOM*, 2018, pp. 630–638.

[11] 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*, 2016, pp. 73–86.

[12] Z. Yang, Y. Cui, X. Wang *et al.*, “Cost-efficient scheduling of bulk transfers in inter-datacenter wans,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1973–1986, 2019.

[13] M. Noormohammadpour, C. S. Raghavendra, S. Rao, and S. Kandula, “DCCast: Efficient Point to Multipoint Transfers Across Datacenters,” Open Science Framework, preprint, Jul. 2017. [Online]. Available: https://osf.io/fg2e5

[14] M. Noormohammadpour, C. S. Raghavendra, S. Kandula, and S. Rao, “QuickCast: Fast and Efficient Inter-Datacenter Transfers Using Forwarding Tree Cohorts,” *Proc. IEEE INFOCOM*, p. 9, 2018.

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

[16] Y. Wang, S. Su, A. X. Liu, and Z. Zhang, “Multiple bulk data transfers scheduling among datacenters,” *Computer Networks*, vol. 68, pp. 123–137, 2014.

[17] Y. Feng, B. Li, and B. Li, “Postcard: Minimizing Costs on Inter-Datacenter Traffic with Store-and-Forward,” in *Proc. IEEE ICDCSW*, 2012, pp. 43–50.

[18] ——, “Jetway: Minimizing costs on inter-datacenter video traffic,” in *Proc. ACM Multimedia*, 2012, pp. 259–268.

[19] T. Nandagopal and K. P. Puttaswamy, “Lowering inter-datacenter bandwidth costs via bulk data scheduling,” in *Proc. IEEE/ACM CCGrid*, 2012, pp. 244–251.

[20] W. Li, K. Li, D. Guo, G. Min, H. Qi, and J. Zhang, “Cost-minimizing bandwidth guarantee for inter-datacenter traffic,” *IEEE Trans. Cloud Comput.*, 2016.

[21] Z. Yang, Y. Cui, X. Wang *et al.*, “Cost-Efficient Scheduling of Bulk Transfers in Inter-Datacenter WANs,” *IEEE/ACM Trans. Netw.*, vol. 27, no. 5, pp. 1973–1986, 2019.

[22] Z. Duliński, R. Stankiewicz, G. Rzym, and P. Wydrych, “Dynamic traffic management for sd-wan inter-cloud communication,” *IEEE J. Sel. Areas Commun.*, 2020.

[23] L. Luo, H. Yu, K. Foerster, M. Noormohammadpour, and S. Schmid, “Inter-datacenter bulk transfers: Trends and challenges,” *IEEE Network*, vol. 34, no. 5, pp. 1–7, 2020.

[24] D. Bertsimas and M. Sim, “The Price of Robustness,” *Operations Research*, vol. 52, no. 1, pp. 35–53, 2004.

[25] N. Karmarkar, “A new polynomial-time algorithm for linear programming,” in *Proc. ACM STOC*, 1984, p. 302–311.

[26] F. P. Kelly, “Effective bandwidths at multi-class queues,” *Queueing Systems*, vol. 9, no. 1-2, pp. 5–15, 1991.

[27] Y. Sun, X. Yin, J. Jiang *et al.*, “CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in *Proc. ACM SIGCOMM*, 2016, pp. 272–285.

[28] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, “A control-theoretic approach for dynamic adaptive video streaming over HTTP,” in *Proc. ACM SIGCOMM*, 2015, pp. 325–338.

[29] Mininet. Accessed: October 3, 2020. [Online]. Available: http://mininet.org/

[30] S. Shenker, D. Clark, D. Estrin, and S. Herzog, “Pricing in computer networks: Reshaping the research agenda,” *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 2, pp. 19–43, 1996.

[31] X. Dimitropoulos, P. Hurley, A. Kind, and M. P. Stoecklin, “On the 95-percentile billing method,” in *Proc. Passive and Active Network Measurement*. Springer, 2009, pp. 207–216.