Guarantee Diverse Bandwidth Availability Targets Over Inter-DC WAN

Han Zhang*, Xia Yin†, Xingang Shi* , Jilong Wang*, Zhiliang Wang*, Yingya Guo‡, Tian Lan§, Haijun Geng ¶
*INSC&BNRist, Tsinghua University
† DCST, Tsinghua University
‡College of Mathematics and Computer Science, Fuzhou University
§ Department of Electrical and Computer Engineering, George Washington University
¶ School of automation and software engineering, Shanxi University
Email:zhhan@tsinghua.edu.cn

***Abstract*—Inter-DataCenter Wide Area Network (Inter-DC WAN) that connects geographically distributed data centers is becoming one of the most critical network infrastructures. Due to limited bandwidth and inevitable link failures, it is highly challenging to guarantee network availability for services, especially those with stringent bandwidth demands, over inter-DC WAN. We present** TEDAT**, a novel Traffic Engineering (TE) framework for *bandwidth availability* (BA) provision, where a Service Level Agreement (SLA) is defined to ensure that each bandwidth demand must be satisfied with a stipulated probability, when subjected to the network capacity and possible failures of the inter-DC WAN.** TEDAT **has two core components, i.e., traffic scheduling and failure recovery, which are crystalized through different mathematical models and theoretically analyzed. They are also extensively compared against state-of-the-art TE schemes, using a testbed as well as real trace driven simulations across different topologies, traffic matrices and failure scenarios. Our evaluations show that, compared with the optimal admission strategy,** TEDAT **can speed up the online admission control by 30× at the expense of less than 4% false rejections. On the other hand, compared with the latest TE schemes like FFC and TEAVAR,** TEDAT **can meet the bandwidth availability SLAs for 23%∼60% more demands under normal loads, and when network failure causes SLA violations, it can retain 10%∼20% more profit under a pricing and refunding model.**

***Index Terms*—Traffic engineering, Bandwidth availability, WAN, profit.**

I. Introduction

Nowadays, large scale online services such as finance trading, web search, online shopping, online game and video streaming are posing stringent requirements on the availability and flexibility of the network infrastructures, where Inter-DataCenter Wide Area Network (Inter-DC WAN) that connects geographically distributed data centers has been playing a critical role. Many service providers, including Amazon, Google, Microsoft, etc., are providing various optimizations for their global WAN, especially with the help of the emerging software-defined networking techniques [20], [28], [31]–[34], [37], [39], [42], [43], [46].

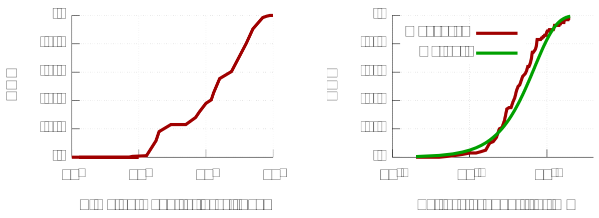
Among various optimization targets, high network availability has been, and will continue to be a major focus. On the one hand, it supports critical uninterrupted services and satisfies fastidious users, while on the other hand, it helps to build a good reputation and improves the competitiveness of network providers. However, guaranteeing network availability for services, especially those with stringent bandwidth demands, over inter-DC WAN is very challenging, since failures may arise from various network components, from data plane to control plane, and could happen anytime [27], [28], [54]. For example, Microsoft reports links in their WAN could fail as often as every 30 minutes [46]. Once a link fails, traffic has to be rescaled and rerouted, resulting in transit or long lasting congestion. Such negative impacts on inter-DC WAN services will ultimately translate into monetary loss (e.g., more refund to customers in the short term, and low customer stickness in the long term). At the same time, as more businesses move to cloud, there are inevitable competitions over the scarce inter-DC WAN bandwidth [31], [39], [59]. Therefore, the design and optimization of inter-DC WANs have to take competitions, heterogeneities and economic objectives into consideration.

In this paper, we argue that although existing traffic engineering schemes [20], [31], [33], [36], [43], [46], [57] have already factored in network risks and aimed for network availability guarantee, they cannot meet the above objectives due to three limitations: *First*, most of them [20], [36], [43], [46], [57] typically make a conservative bandwidth allocation, so that even if a failure occurs, surviving paths could be used and the network can still be free from congestion under traffic rerouting. To prevent congestion, links, including those with negligible failure probabilities, must be kept at low utilization, resulting in significant waste of network bandwidth (and potentially less accommodated users). While such a solution is adopted by existing ISPs that use over provision to avoid congestion, it is highly inefficient for new players, such as content providers that are building their own backbone network (either physically [6], [28], [31], [33] or leasing bandwidth from ISPs [7], [21]), this is quite uneconomic [30]. *Second*, existing techniques mainly focus on the availability of the whole network, but ignore the fact that users' expectations for reliability may vary significantly in practice. Providing reliable bandwidth can be a value-added service for many cloud providers, typically in the form of Service Level Agreements (SLAs) [6], [8]. For example, Microsoft Azure guarantees its customers at least 99.9% availability for its backup service

Corresponding authors: Xingang Shi, email: shixg@cernet.edu.cn and 99.95% availability for its ExpressRoute service [8]. If the availability agreement is violated, a 10% or 25% refund will be returned to the customers. A *one-size-fit-all* approach (e.g., TEAVAR [20]) ignoring these heterogeneities cannot support such SLAs well, and may even hurt critical and uninterruptible applications when there are competitions on bandwidth. *Third*, such heterogeneities and competitions are either not considered by their failure recovery approaches, especially those who allocate bandwidth aggressively [20], [31]. Therefore, without a systematic optimization framework, services may run into congestion when traffic is rerouted under network failures. Such violations of SLAs will inevitably cause hefty revenue loss for service providers. To solve these challenges, in this paper we make the following three **contributions**:

Firstly, we advocate traffic engineering with *bandwidth availability* (*BA*) provision: a BA demand $d = (b_d, \beta_d, t_d^s, t_d^e)$ requests bandwidth b_d for a life duration from t_d^s to t_d^e, and should be guaranteed at least β_d% of the duration, subjected to the network capacity and possible failures. Such a demand is typically represented by a Service Level Agreement (see TABLE I for real world examples), which may differ substantially across users and applications. We show that state-of-the-art traffic engineering schemes fail to meet the heterogeneous bandwidth availability demands, especially under diverse link failure probabilities that may vary by several orders of magnitude (see §II). We note that, although the general concept of bandwidth-based availability has been recognized in some recent TE works (e.g., B4 [32], [33], [42], TEAVAR [20]), their methodologies and evaluations are actually achieving *only a soft guarantee*, i.e., a high ratio of the allocated bandwidth to the negotiated one, while we will provide a **hard guarantee**, i.e., the negotiated bandwidth **must** be met.

Secondly, we design TEDAT, a novel Traffic Engineering framework that aims for guaranteeing Diverse bandwidth Availability Targets over inter-DC WAN (see §III). TEDAT is composed of two core components, i.e., traffic scheduling and failure recovery. After a new BA demand arrives, our traffic scheduling procedure will determine whether it can be admitted or not. If the demand can be admitted, our traffic scheduling procedure will also allocate bandwidth for it to guarantee bandwidth availability. We model the traffic scheduling procedure as a 0-1 Mixed Integer Linear Programming (MILP) problem and then prove it is a NP-hard problem. We propose a heuristic algorithm to strike a balance between efficiency and optimality, so the new demands can be admitted and guaranteed as much as possible. We advocate to use economic interests to guide our design of failure recovery procedure. We model the failure recover procedure as a Linear Programming (LP) problem with manageable number of constraints. Therefore, when failures happen, the surviving tunnels can be used immediately.



(a) Failure intervals (b) Empirical link failure

Fig. 1. A commercial inter-DC WAN empirical data.

TABLE I
SERVICES HAVE DIFFERENT AVAILABILITY TARGETS.

Service	Availability	Refund
Azure Traffic Manager [8]	$< 99.99\%$	10%
Azure VPN Gateway [8]	$< 99.95\%$	10%
Amazon VM Instances [6]	$< 99.99\%$	10%
Azure Cosmos DB [8]	$< 99.999\%$ $< 99\%$	10% 25%
AWS DMS [5]	$< 99.99\%$ $< 99.0\%$ $< 95\%$	10% 30% 100%
Amazon AppFlow [4]	$< 99.99\%$ $< 99.95\%$ $< 95\%$	10% 25% 100%
Alibaba SMS [2]	$< 95\%$ $< 90\%$	10% 30%
Alibaba Data Transmission [1]	$< 99.9\%$ $< 99.0\%$ $< 95\%$	15% 30% 100%

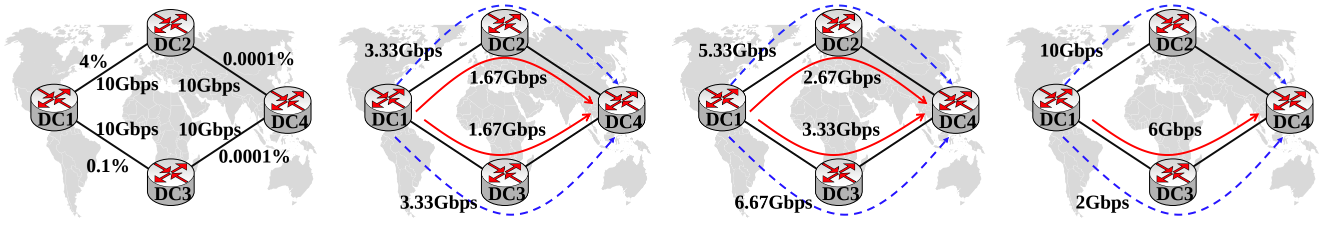
Thirdly, we implement TEDAT as a real system, including a centralized controller and multiple brokers (one for each DC) (see §IV). We conduct extensive experiments using a network testbed as well as trace driven large scale simulations (see §V). We compare TEDAT with state-of-the-art WAN TE schemes such as TEAVAR [20], SMORE [43], SWAN [31], B4 [33] and FFC [46], across different topologies, traffic matrices and failure scenarios. Our evaluations on real network topologies and traces demonstrate that, TEDAT can (1) speed up the online admission control by 30× at the expense of a false rejection ratio that is less than 4%; (2) meet the bandwidth availability SLAs for 23%∼60% more demands under normal loads; (3) retain 10%∼20% more profit when network failure causes SLA violations, under a pricing and refunding model. To our knowledge, TEDAT is the first to tackle bandwidth availability provision over inter-DC WAN, where heterogeneities of demands and link failures are systematically taken into account for profit maximization.

II. Background and Motivation

In this section, we first briefly introduce network failures and common availability requirements in inter-DC WAN, then we use an example to show state-of-the-art traffic engineering schemes' limitations in fulfilling such requirements.

A. Network failures and availability

WAN failures are frequent and follow a heavy-tailed distribution. Failures could occur anywhere, from control plane to data plane across the network [20]. They could also last for long durations, as Google reports, more than 80% of the failures last between 10 mins and 100 mins over their B4 network [3], [28], leading to severe performance degradation and revenue loss. Our failure intervals measurement of a commercial inter-DC WAN shown in Fig. 1(a)



(a) Capacity and failure probability (b) FFC (with one failure) (c) TEAVAR (d) TEDAT

Fig. 2. A simple global wan example, where user1 (solid) requires 6Gbps bandwidth for at least 99% time and user2 (dash) requires 12Gbps bandwidth for at least 90% time, both from DC1 to DC4.

TABLE II
BANDWIDTH AVAILABILITY TARGETS IN B4 [32].

Service	Availability
Search ads, DNS, WWW	99.99%
Photo service, backend, Email	99.95%
Ads database replication	99.9%
Search index copies, logs	99%
Bulk transfer	N/A

indicates failures are common cases (e.g., more than 80% failure intervals are in less than 6 hours). The empirical failure probability demonstrated in Fig. 1(b) shows link failures often follow a *heavy-tailed distribution*, where a small portion of links contribute to most of the failures and the failure rate of a single link can differ by even more than two orders of magnitude. Our measurements also match previous works [26], [27], [54]. Therefore, *network failures, especially their uneven distribution, should be explicitly taken into account by network operators*.

Bandwidth availability guarantee may be beneficial. Availability has attracted major attention both in the industry and research community. A Service Level Objective (SLO) of $\beta\%$ connectivity-based availability specifies that a certain quality of connectivity (i.e., packet loss is below a certain threshold) should be available $\beta\%$ time [32]. However, only connectivity-based availability is insufficient. In recent years, there has been a rapid increase in deploying online services (e.g., online videos, online game, online shopping, live broadcast) over clouds. Concurrent with this trend has been a steady rise in bandwidth demand. Many studies have shown that users will quickly abandon sessions if their minimal bandwidth cannot be guaranteed, leading to significant losses in revenue for content providers [41], [48], [52]. Therefore, for a BA demand $d = (b_d, \beta_d, t_d^s, t_d^e)$, formulating the *hard* guarantee such as "demand d's bandwidth b_d for a life duration from t_d^s to t_d^e is guaranteed at least $\beta_d\%$ of the duration" may be beneficial.

High availability directly translates into profit. Nowadays, high availability is nearly always one of the main items in SLAs [5], [6], [8], and customers are eligible for a credit refund if there are SLA violations. We conduct a survey on the SLA claims of different cloud providers, and TABLE I demonstrates their declared availability targets and corresponding refunding policies. The credit to be refunded is typically represented by a simple step function. For example, Microsoft Azure provides 10% refund if its Traffic Manager service availability falls between 99.99% and 99.0%, and provides 30% refund for any availability below 99.0% [8]. As more real-time and mission-critical applications (financial trading, online game, video streaming, instant messaging, live broadcast, etc.) are deployed on the Internet, *providing hard guarantee of high availability under network failures to retain a good profit is a big challenge*.

Providing a one-size-fit-all network availability is not enough. In recent years, there has been a surging increase in rapid and agile deployment of services over clouds. Many studies have shown that users will quickly abandon sessions if the qualify of service is not guaranteed, leading to significant losses in revenue for content providers [41], [48], [52]. Multiple services might be simultaneously launched over the global infrastructure operated by the same content provider or cloud provider. They might also pose different availability requirements, and will contend for the inter-DC WAN bandwidth. As B4's availability targets [32], [33] shown in TABLE II, the minimal availability demands of DNS and logs are 99.99% and 99%, respectively. *Such heterogeneous availability demands cannot be well captured and handled by a one-size-fit-all approach*, where all users get the same level of availability guarantee (e.g, TEAVAR [20] only considers guaranteeing all users' bandwidth at least $\beta\%$ time).

B. *A motivating example for* TEDAT

Now we use a simple example to illustrate why existing traffic engineering algorithms cannot meet the heterogeneous bandwidth availability demands. The toy topology we use is depicted in Fig. 2(a), where there are 4 data centers. The links connecting them are annotated with their corresponding capacities as well as failure probabilities. Suppose we have two bandwidth demands for inter-DC transmission from DC1 to DC4, i.e., user1 (solid) requires 6Gbps bandwidth with at least 99% availability, and user2 (dash) requires 12Gbps bandwidth with at least 90% availability. There are two paths from DC1 to DC4, i.e., DC1→ DC2 → DC4, and DC1 → DC3 → DC4, whose available probabilities are $(1-4\%)\times(1-0.0001\%) = 95.999904\%$ and $(1-0.1\%)\times(1-0.0001\%) = 99.8999001\%$, respectively. We apply FFC [46] and TEAVAR [20], two latest WAN traffic engineering schemes that take network failures into account, to this scenario.

FFC [46] guarantees a total bandwidth from DC1 to DC4 under at most l concurrent node/link failures, and here we simply use $l = 1$. Fig. 2(b) shows FFC can support 10Gbps bandwidth from DC1 to DC4 in 99.996% time even with one failure (the probability that the two paths fail simultaneously is $(1-95.999904\%)\times(1-99.8999001\%) = 0.004004092096\%$). User1 and user2 can respectively get 3.34Gbps and 6.66Gbps, which are evenly distributed on the two paths from DC1 to DC4, and neither of their bandwidth demands can be satisfied. This shows *FFC makes a conservative allocation and does not differentiate between paths with different availabilities.* Path (2) has a much smaller failure probability, and lowering its utilization is wasteful.

On the other hand, TEAVAR [20] exploits the different link failure probabilities and maximizes the network utilization, subject to meeting a *single* desired availability. Fig. 2(c) illustrates the bandwidth allocation result of TEAVAR, where user1 and user2 can get their demanded 6Gbps and 12Gbps bandwidth, both in about 95.9% time. However, this falls below user1's availability demand, i.e., 99%, and will cause BA target violation. This shows *TEAVAR does not consider the heterogeneous user demands on availability.* Since user1 requires a higher availability, it is better to use a path with a lower failure probability.

Our approach: Taking into account the diverse link failure probabilities and user bandwidth availability demands, Fig. 2(d) shows a better allocation, where user1 can get 6Gbps over 99.8999001% time (via the path that has a lower failure probability) and user2 can get 12Gbps over 95.999904% time (via both paths). Therefore, both users' bandwidth availability demands are satisfied.

TABLE III
KEY NOTATIONS FOR TEDAT.

Input Variables	
$G(V, E)$	inter-DC WAN with nodes V and Links E
$\mathbf{z} \in Z$	a network failure scenario in the scenario set
$p_{\mathbf{z}}$	the probability that a failure scenario $\mathbf{z}$ occurs
$k \in K$	a s(ource)-d(est) pair in the set of all s-d pairs
T_k	the set of tunnels for a s-d pair k
$d = (\mathbf{b}_d, \beta_d)$	a BA demand d, requiring bandwidth $\mathbf{b}_d$ with availability β_d, where $\mathbf{b}_d$ is a vector of bandwidth demands over all s-d pairs
$D, \hat{D}$	the set of arrived and admitted demands[1]
t	a tunnel for transmitting traffic[2]
u_t^e	whether tunnel t passes link $e \in E$
c_e, c_t	the remaining capacity on link e or a tunnel t
$v_t^{\mathbf{z}}$	whether tunnel t is available under scenario $\mathbf{z}$
$w_e^{\mathbf{z}}$	whether link e is available under scenario $\mathbf{z}$
g_d	the charge for serving demand d
Output Variables	
f_d^t	bandwidth allocated for demand d over tunnel t
r_d	profit (after refunding) for demand d

III. TEDAT FRAMEWORK

In this section, we discuss the details of TEDAT, which contains two parts, i.e., traffic scheduling and failure recovery. Main notations are summarized in Table III. The framework intends to achieve the following objectives:

- **High admission ratio and low admission latency:** Bandwidth availability demands might arrive at anytime. The system should be able to efficiently accommodate as many BA demands as possible under the constraint of network capacity and failure probabilities, as this would increase service agility and might bring more revenue.
- **Guarantee availability for allocated bandwidth:** The system should be able to guarantee the availability of demands according to link failure probabilities, as this would reduce potential penalties and retain a good reputation in the long term. This can be achieved by making a good match between demands on higher availability and paths which fail less probably.
- **Automatic and economical failure recovery:** If any link failure really happens, the system should reroute traffic away from that link, while minimizing any possible collateral damage and revenue loss, i.e., congestion due to contention caused by the rerouted traffic.

A. Abstraction of bandwidth availability

In reality, a bandwidth availability demand asking for inter-DC WAN bandwidth resources could from any application spanning multiple data centers in a private cloud. Our abstractions on network failure scenarios and bandwidth availability demands are as follows.

Network failure scenario model: The inter-DC WAN is modeled as a directed graph $G(V, E)$, where the set of nodes V represent the data centers, and the set of links E represent directed links between them. A network scenario $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{|E|}\}$ is a vector of link states, where each element $\mathbf{z}_i \in \{0, 1\}$ denotes whether the i-th link is up ($\mathbf{z}_i = 1$) or down ($\mathbf{z}_i = 0$). We assume network operators can use historical data to estimate the failure probability x_i for this link, which are statistically independent [3]. Let Z denote the network scenario set, then the expected probability that a network scenario $\mathbf{z} \in Z$ will happen is given by [20], i.e.,

$$p_{\mathbf{z}} \quad = \quad \prod_{i=1}^{|E|} \big(\mathbf{z}_i \times (1 - x_i) + (1 - \mathbf{z}_i) \times x_i\big)$$

Take the simple inter-DC WAN topology in Figure 2 as an example, where $E = \{e_1, e_2, e_3, e_4\}$. Network scenario $\mathbf{z} = \{1, 1, 0, 1\}$ means e_1 ,e_2, e_4 are working fine and e_3 is down. The expected availabilities of e_1, e_2, e_3, e_4 are 96%, 99.9999%, 99.9%, 99.9999%, respectively. Then the probability of $\mathbf{z}$ is $p_{\mathbf{z}} = p_{\{1,1,0,1\}} = 0.96 \times 0.999999 \times 0.001 \times 0.999999 \simeq 0.000959998$.

BA demand model: Let K denote the set of all source-destination (s-d) DC pairs. A bandwidth availability demand d is in the form of $(\mathbf{b}_d, \beta_d)$, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, ..., \mathbf{b}_d^k, ... >$ of bandwidth demands on each s-d pair $k \in K$ [4] and β_d is its bandwidth availability target.

[3] The strong assumption does not affect the network scenario model.

[4] Here we omit the start and end time of this demand, but they will be implicitly considered in our online admission and traffic scheduling.

BA provision model: Similar to [20], [31], [46], TEDAT also adopts tunnel-based forwarding. For each source-destination node pair $k \in K$ of the inter-DC WAN, we pre-compute a set of tunnels T_k with different routing schemes (e.g., k-shortest paths, edge disjoint paths [56], oblivious routing [43], etc.). Each tunnel $t \in T_k$ contains a sequence of links and u_t^e denotes whether tunnel t passes a specific link $e \in E$ or not. Let D and $\hat{D}$ represent *arrived* demands and *admitted* demands, respectively. Given a new demand, the admission control scheme (see § III-B) will decide whether to admit it and makes the bandwidth allocation for the admitted ones.

We use $v_t^{\mathbf{z}}$ to denote whether tunnel t is available (i.e., $v_t^{\mathbf{z}} = 1$) or not (i.e., $v_t^{\mathbf{z}} = 0$) under network scenario $\mathbf{z}$. Given a BA demand $d = (\mathbf{b}_d, \beta_d)$, an allocation result $\{f_d^t\}$ and a network scenario $\mathbf{z}$, for *every* s-d pair k, if the total allocated bandwidth on available tunnels under $\mathbf{z}$, i.e., $\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}$, is no less than the bandwidth demand $\mathbf{b}_d^k$, then we call $\mathbf{z}$ a *qualified scenario* for allocation $\{f_d^t\}$ with respect to demand d, and denote this by $\mathbf{z} \propto < d, \{f_d^t\} >$. The sum of the probabilities of all such qualified scenarios, i.e., $\sum_{\mathbf{z} \propto < d, \{f_d^t\} >} p_{\mathbf{z}}$, is the expected probability that the bandwidth target $\mathbf{b}_d$ will be satisfied. Now we can formally define when a bandwidth availability demand is satisfied: a demand d is satisfied by an allocation $\{f_d^t\}$, if and only if

$$\sum_{\mathbf{z} \propto < d, \{f_d^t\} >} p_{\mathbf{z}} \geq \beta_d$$

.

If a failure indeed occurs, our failure recovery scheme (see § III-C) will try to reroute traffic that is affected by this failure. If any availability target is violated, a refund will be given back to the customer according to our recommending model, and we use r_d to denote the profit (after refunding) for serving demand d.

B. Traffic scheduling

User demands are served in a first-come-first-service (FCFS) manner without preemption. When a new demand d arrives, we have $D = \hat{D} \cup d$. The optimal traffic scheduling strategy would try to accommodate as many demands as possible: *if every demand in D can meet its availability target, then d should be admitted and the traffic scheduling procedure will allocate bandwidth to it, otherwise, it should be rejected.* We now try to formulate the traffic scheduling problem as follows:

For a source-destination pair k of BA demand d, let $R_{dk}^{\mathbf{z}}$ denote the ratio of the effective bandwidth under network scenario $\mathbf{z}$ to the demanded bandwidth:

$$R_{dk}^{\mathbf{z}} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in D, \mathbf{z} \in \mathbf{z}, k \in K. \tag{1}$$

where $v_t^{\mathbf{z}}$ represents whether tunnel t is available under network scenario $\mathbf{z}$. For every source-destination pair k, if the total effective bandwidth on all the available tunnels is larger than $\mathbf{b}_d^k$, then the bandwidth target can be met under $\mathbf{z}$, even some tunnels fail. In this situation, the network scenario $\mathbf{z}$ can be regarded as *qualified*. Let $q_d^{\mathbf{z}}$ denote whether scenario $\mathbf{z}$ is qualified (i.e., $q_d^{\mathbf{z}} = 1$) or not (i.e., $q_d^{\mathbf{z}} = 0$) for the BA demand d:

$$q_d^{\mathbf{z}} = \begin{cases} 1 & \text{if } R_{dk} \geq 1 \text{ for every } k \in K \\ 0 & \text{Otherwise} \end{cases}$$

It can be rewritten as

$$\begin{array}{rll} q_d^{\mathbf{z}} \in & \{0,1\}, & \forall d \in \hat{D}, \mathbf{z} \in Z \\ R_{dk}^{\mathbf{z}} < & M \times q_d^{\mathbf{z}} + 1 - q_d^{\mathbf{z}}, & \forall d \in \hat{D}, k \in K \\ R_{dk}^{\mathbf{z}} \geq & q_d^{\mathbf{z}}, & \forall d \in \hat{D}, k \in K \end{array} \tag{2}$$

where M is a constant larger than the upper bound of $R_{dk}^{\mathbf{z}}$. The achieved bandwidth availability of demand d is the total probabilities of all *qualified* network scenarios, i.e.,

$$s_d = \sum_{\mathbf{z} \in \mathbf{z}} q_d^{\mathbf{z}} \times p_{\mathbf{z}}, \quad \forall d \in D. \tag{3}$$

If s_d is not smaller than the BA target β_d, then the availability target can be satisfied. Use a_d to represent whether the BA target of d can be satisfied, then we have:

$$a_d = \begin{cases} 1 & \text{if } \beta_d \leq s_d \leq 1 \\ 0 & \text{if } 0 \leq s_d < \beta_d \end{cases}$$

which can be further written as

$$\begin{array}{rll} a_d \in & \{0,1\}, & \forall d \in D \\ s_d < & \beta_d \times (1 - a_d) + a_d, & \forall d \in D \\ s_d \geq & \beta_d \times a_d, & \forall d \in D \end{array} \tag{4}$$

In addition, the bandwidth allocation result f_d^t for BA demand d over tunnel t should be non-negative and limited by link capacities, i.e.,

$$f_d^t \geq 0, \quad \forall d \in D, k \in K, t \in T_k. \tag{5}$$

and

$$\sum_{d \in D} \sum_{k \in K, t \in T_k} f_d^t u_t^e \leq c_e, \quad \forall e \in E. \tag{6}$$

Finally, the traffic scheduling intends to maximize the total number of accepted demands with the above constraints, i.e.,

$$\begin{array}{l} maximize \sum_{d \in D} a_d \\ s.t. (1), (2), (3), (4), (5), (6) \end{array} \tag{7}$$

The output variables a_d are chosen from {0,1}, so that the traffic engineering problem is a 0-1 Mixed-integer linear programming. By reducing the NP-hard all-or-nothing multi-commodity flow problem [22] to a special case of the traffic engineering problem, we can obtain the following lemma.

Lemma 1: TEDAT problem is a NP-hard problem.

Proof: TEDAT problem contains the all-or-nothing multi-commodity flow problem as a special case, which is known as an NP-hard problem [22]. Consider an undirected graph $G = (V, E)$ and a set of bandwidth demands $d_1, d_2, ...$, where V is the node set, E is the link set and each demand corresponds to a commodity flow to be sent from the source node s_d to the destination node t_d with bandwidth demand b_d. Let T_d denote the path set for demand d. The all-or-nothing multi-commodity flow problem tries to find a maximum routable set:

$$
\begin{aligned}
& maximize \sum_{d \in D} a_d \\
s.t. \quad & \forall e \in E : \textstyle\sum_{d\in D} \sum_{t \in T_d} f_d^t u_t^e \le c_e \\
& \forall d \in D : a_d = \begin{cases} 1 & \sum_{t\in T_d} f_d^t \ge b_d \\ 0 & \sum_{t \in T_d} f_d^t < b_d \end{cases}
\end{aligned}
\tag{8}
$$

Where a_d denotes whether commodity flow d can be routable. We now consider a special case of TEDAT problem, in which all links/nodes are available, i.e., there is only one network scenario. We further assume there is only one non-zero element in vector $\mathbf{b}_d$ for each demand $d \in D$. In the scenario, if the allocated bandwidth is larger than the demand, then the BA target can be satisfied ($a_d = 1$), otherwise, the target is violated ($a_d = 0$). We consider regarding the BA demands and their bandwidth demands in TEDAT problem as the multi-commodities and their demand in the all-or-nothing multi-commodity flow problem. If we can solve the special case of TEDAT problem with a polynomial time algorithm, we would obtain the routable multi-commodity flow set in the all-or-nothing multi-commodity flow problem. Therefore, TEDAT is at least as hard as the all-or-nothing multi-commodity flow problem, which is known to be NP-hard. This completes the proof. ■

However, in order to support agile deployment of new applications and services, user demands should be admitted as fast as possible, while the time needed to exactly solve this NP-hard problem may be prohibitive. Therefore, we need a better trade-off between efficiency and optimality. The final admission control strategy we use is as follows:

1) When a new demand d arrives, we *fix* the bandwidth allocation for all admitted demands in $\hat{D}$, then we check whether d can be satisfied by the remaining network capacity and failure probability. If the answer is positive, then admit d and make bandwidth allocation for it without rescheduling the whole network.
2) Otherwise, run a greedy algorithm (Algorithm 1) to *conjecture* whether the admitted demands can be rescheduled to accommodate d. If the answer is positive, then we will reschedule the whole network and admit d.
3) If d still cannot be accommodated, reject the demand without rescheduling the whole network.

The greedy algorithm tries to conjecture, in an efficient way, whether an allocation strategy satisfying all demands (i.e., including d) exists. It works iteratively as follows. In each iteration, it finds the demand which has the smallest product of bandwidth target and availability target (i.e., $\sum_{k\in K} \hat{b}_d^k \times \beta_d$) at first (line 4), and tries to allocate bandwidth for each of its s-d pairs one by one. If the remaining network capacity cannot satisfy this demand, we will give up and the network will not reschedule (line 6-7). Otherwise, it allocates tunnel bandwidth for this demand, where a tunnel with a smaller product of remaining capacity and availability has a higher priority (line 9-15). After this, if the availability target cannot be roughly satisfied, it will give up and the network will not reschedule (line 16-17), otherwise, it will go for the next iteration.

The time complexity of Algorithm 1 is $O(|D| * |K| * max(|T_k|))$. *It is also worth to note that, there is no false positive in conjectures made by Algorithm 1*, as indicated by the following lemma.

Lemma 2: If a new demand d can be admitted by Algorithm 1, then there must exist an allocation result $\{f_d^t\}$ to satisfy the bandwidth availability targets of all demands $D = \hat{D} \cup d$.

Proof: We prove by contradiction. Suppose there is a BA demand that is admitted by Algorithm 1 but the network is unable to satisfy its bandwidth availability. There are two possible cases: (i) network bandwidth is insufficient; (ii) The availability provided by the network is not enough. Case (i) is impossible, because if bandwidth is insufficient (i.e., $\mathbf{b}_d^k$ is larger than the remaining network capacity for s-d pair k) , Algorithm 1 won't admit the demand (Line 6-7). Case (ii) is also impossible, because if the bandwidth availability is smaller than its target (i.e., $s_d < \beta_d$) , Algorithm 1 will reject the demand (Line 16-17). This completes the proof. ■

Algorithm 1: Heuristic for Solving Traffic Scheduling

```
   Input: Input parameters shown in TABLE III.
   Output: Bandwidth allocation results.
1  $\hat{b}_d^k = b_d^k, \forall d \in D, k \in K$;
2  $\hat{f}_d^t = f_d^t, \forall d \in D, k \in K, t \in T_k$;
3  while $true$ do
4      $d = \arg_{d' \in D} min\{\sum_{k\in K} \hat{b}_{d'}^k \times \beta_{d'}\}$;
5      for $k \in K$ do
6          if $b_d^k >$ remaining capacity of s-d pair $k$ then
7              return $False$, $\{f_d^t\}$;
8          $T_k' = T_k$;
9          while $\hat{b}_d^k > 0$ do
10             $t = \arg_{t \in T_k'} \min\{c_t * p_t\}$;
11             $\hat{f}_d^t = \min\{c_t, \hat{b}_d^k\}$;
12             $T_k' = T_k' \setminus t$;
13             $s_d = s_d * p_t$;
14             $\hat{b}_d^k = \hat{b}_d^k - \hat{f}_d^t$;
15             update the remaining capacities of links
                 and tunnels;
16     if $s_d < \beta_d$ then
17         return $False$, $\{f_d^t\}$;
18     $D = D \setminus d$;
19 return $True$, $\{\hat{f}_d^t\}$;
```

Analyzing the performance loss of general TEDAT is not the focus of this paper. Therefore, we take the simplest dumb-bell topology (i.e., $G(V, E) = \{v_1, v_2, e\}$) as the example to show the performance bound of our algorithm.

Lemma 3: The approximation of Algorithm 1 for TEDAT problem under $G(V, E) = \{v_1, v_2, e\}$ is 2.

Proof: Algorithm 1 will prior BA demands according to the following sequence (Line 4):

$$\sum_{k\in K} \hat{b}_{d1}^k \times \beta_{d1} \le \sum_{k \in K} \hat{b}_{d2}^k \times \beta_{d2} \le .... \tag{9}$$

We now consider a network state where link e has already admitted n BA demands but it can't admit the $n + 1$ ones.

Let OPT denote the optimal solution and it is obvious that $\sum_{i=1}^{n} a_i \leq OPT$. Also, we have $\sum_{i=1}^{n+1} a_i \geq OPT$. This holds, since we've already made the density of e as high as possible by the greedy method. If we violate the link capacity constraint and put the $n+1$ BA demands into the link, then the link is fulfilled. There is no other way that the density of the link is greater than this, that is, the value is greater than OPT. $\sum_{i=1}^{n+1} a_i \leq 2\sum_{i=1}^{n} a_i$. Therefore, $OPT \leq 2\sum_{i=1}^{n} a_i$. This completes the proof. ■

C. Failure recovery

When failures occur and any tunnel becomes unavailable, traffic can be redistributed across the surviving tunnels. To reduce recovery time, TEDAT proactively computes backup allocation strategies for potential failure scenarios, so that the surviving tunnels can be used immediately, and packet loss can be mitigated [5]. For example, in Fig. 3, there are two users, and the link capacity is 1 everywhere. One user requests a bandwidth of 1 from DC1 to DC2, while the other one requests a bandwidth of 1 from DC1 to DC4. Fig. 3(a) shows the original bandwidth allocation when no failures occur, and Fig. 3(b) depicts the backup allocation pre-computed for a failure of link DC2→DC4.

Bandwidth availability, even well planned, cannot always be guaranteed due to network failures, and this ultimately hurts the reputation of the cloud providers. In reality, many popular cloud services (e.g., Amazon Compute Service [6], Azure Active Directory Domain Service [8]) will refund their customers in case their agreed SLAs are violated. For example, the Amazon Compute Service SLA [6] defines that they will provide 10% refund if the achieved availability (e.g., monthly uptime percentage) is between 99.99% and 99.0%. Although this practice is specified for scenarios other than inter-DC WAN, its principles and policy designs might provide good hints for inter-DC WAN services. We borrow the SLA violation refunding idea from the popular cloud services (e.g., Amazon Compute Service [6], Azure Active Directory Domain Service [8]) and advocate to use economic interests to guide our design of rerouting under failures as follows.

For a specific network scenario $\mathbf{z}$ (where one link failure occurs) in consideration, the ratio of allocated bandwidth to a user's demanded bandwidth is [6]:

$$R_{dk} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in \hat{D}, k \in K \tag{10}$$

If for every k, R_{dk} is larger than its demand (i.e., $R_{dk} \geq 1$), then there is no problem since the demanded availability is still satisfied. However, if any R_{dk} falls below 1, then the corresponding bandwidth availability (BA) target will be violated. For simplicity, here we assume a simple pricing and refunding model, where the charge for serving a user demand d is g_d, and if the bandwidth availability target cannot be guaranteed, a fraction μ_d of g_d will be refunded. We use r_d to denote the profit of demand d with refunding, such that

$$r_d = \begin{cases} g_d & \text{if } R_{dk} \geq 1 \text{ for every } k \in K \\ (1-\mu_d)g_d & \text{Otherwise} \end{cases}$$

We use an auxiliary integer variable y_d to denote the violation condition, where $y_d = 1$ means no violation. Then the profit r_d can be rewritten as

$$\begin{aligned} 0 \leq\ & y_d \leq 1, && \forall d \in \hat{D} \\ r_d =\ & g_d \times \Big(y_d + (1-\mu_d) \times (1-y_d)\Big), && \forall d \in \hat{D} \\ R_{dk} <\ & M \times y_d + 1 - y_d, && \forall d \in \hat{D}, k \in K \\ R_{dk} \geq\ & y_d, && \forall d \in \hat{D}, k \in K \end{aligned} \tag{11}$$

where M is a constant large enough (e.g., at least larger than the upper bound of R_{dk}). We relax the requirement of $y_d \in \{0,1\}$ to $0 \leq y_d \leq 1$ to make the piecewise function linearize.

Besides, the bandwidth allocation result f_d^t should be non-negative and limited by the available network capacity. Let $w_e^{\mathbf{z}}$ denote whether link e is available under scenario $\mathbf{z}$, then we have

$$f_d^t \geq 0, \quad \forall d \in \hat{D}, k \in K, t \in T_k \tag{12}$$

and

$$\sum_{d \in \hat{D}} \sum_{k \in K, t \in T_k^{\mathbf{z}}} f_d^t u_t^e \leq c_e \times w_e^{\mathbf{z}}, \quad \forall e \in E \tag{13}$$

Finally, the failure recovery scheme tries to maximize the total profit (after refunding) by

$$\begin{aligned} & maximize \sum_{d \in \hat{D}} r_d \\ & s.t. (10), (11), (12), (13) \end{aligned} \tag{14}$$

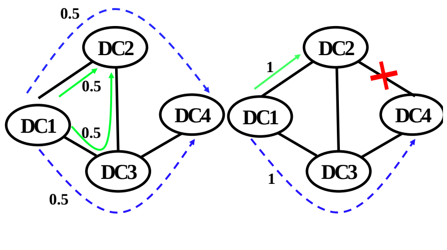
The above failure recovery problem is a LP problem and it is easy to solve it using the optimization problem solver such as Gurobi [29].

IV. SYSTEM IMPLEMENTATION

We have implemented TEDAT on the Linux platform. Fig. 4 shows the whole system architecture, which contains one controller, multiple brokers (one for each DC). The controller is responsible for most decision work of TEDAT, including traffic scheduling, and failure recovery. The brokers and switches are responsible for bandwidth enforcement. The system works as follows: When a user submits a demand to the controller, the traffic scheduling module will determine whether the demand can be admitted or not (see § III-B). If the demand is admitted, this module will allocate its demanded bandwidth on appropriate paths , and notify the brokers for enforcement. In addition, for potential link failures, it also pre-computes backup allocation strategies that will be activated if any link failure indeed happens (see § III-C). These central decisions are distributed to the brokers for bandwidth enforcement. The brokers in each DC monitor link status and bandwidth consumption, report these statistics to the central controller, and ask the switches to enforce rate.

[5] Here we only consider backup allocations for one link, while this scheme can be easily extended to deal with concurrent failures.

[6] This is the same as equation (1), but we omit the superscript $\mathbf{z}$ of $R_{dk}^{\mathbf{z}}$.



(a) Original allocation (b) Backup allocation

Fig. 3. A failure recovery example.

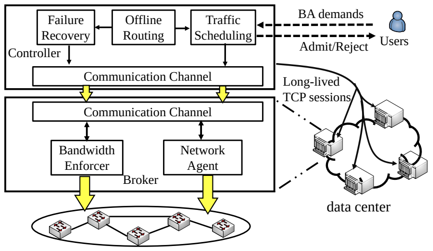


Fig. 4. TEDAT system design.

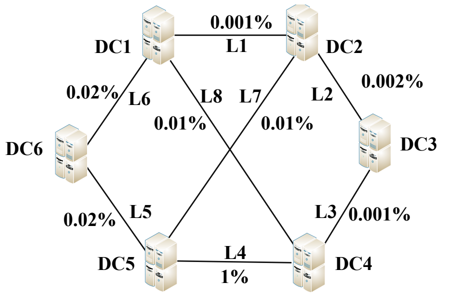


Fig. 5. Testbed topology.

Controller is the brain of the whole system. It is responsible for allocating WAN level bandwidth, and orchestrates all activities with a global view. The four main components in Controller are as follows. (1) Offline Routing. This module maintains the WAN level network topology, and computes TE tunnels between each node pair (i.e., $T_k, \forall$s-d pair $k \in K$), using certain routing algorithms (oblivious routing [43], k-shortest path [31], etc.). These tunnels are used by the admission control module and the online scheduler module as input variables; (2) Traffic scheduling. When a BA demand is submitted, this module uses the traffic scheduling algorithm (see § III-B) to reject it, or accept it and allocate bandwidth over the tunnels in nearly real-time. The results are sent to the corresponding brokers. In addition, our system also supports several other TE algorithms, e.g., SWAN [31], FFC [46] and TEAVAR [20]; (3) Failure recovery. This component will also pre-compute backup allocation (see § III-C) for some potential link failures. For each user demand, the normal bandwidth and backup bandwidth allocated over each tunnel (i.e., f_d^t) are then sent to the corresponding brokers; (4) Communication Channel. This module is responsible for communicating with brokers, where we use long-lived TCP connections to avoid unnecessary delay. Also, controller failures can be remedied by using multiple replications, where the master controller is elected by the Paxos [44] algorithm.

Broker takes care of the data center it resides in. It consists of three modules: (1) Bandwidth Enforcer. It receives the bandwidth allocation results (i.e., f_d^t) from controller, sends them to the corresponding switches connecting with hosts, and limits the actual traffic rate in each tunnel in case something is wrong on the end hosts; (2) Network Agent. We use commodity SDN switches at data center edges to connect DCs into an inter-DC wan. The network agent runs in a SDN controller (we use floodlight [24]), and uses the OpenFlow [50] protocol to installs and updates forwarding rules on the switches in that DC. To reduce rule complexity, our system uses a label-based forwarding scheme, where the first 12 bits of a VxLAN ID represent different demands, and the last 12 bits represent different tunnels. Therefore, 4096 demands and 4096 tunnels can be supported simultaneously, and this can be further expanded if necessary. In this way, a flow (i.e., traffic corresponding to a BA demand) is marked with a label at the ingress switch, and the succeeding switches use this label for forwarding. Group tables in the switch pipelines are used for flow splitting (i.e., traffic corresponding to a BA demand can be split into multiple sub-flows and transmitted in multiple tunnels). Besides, the network agent also tracks the network topology, reports any change or failure to the central Controller module, and monitors the actual traffic rate; (3) Communication Channel. This component is responsible for communication with the controller.

V. Evaluation

In this section, we use a small testbed and large scale trace driven simulations to evaluate the performance of TEDAT. On the testbed, we also implement another two state-of-the-art TE algorithms that consider network availability, i.e., FFC [46] and TEAVAR [20]. For simulation, we implement more TE algorithms, including SWAN [31], SMORE [43] and B4 [33]. Our main results are as follows:

(1) TEDAT consistently outperforms latest TE algorithms under various topologies, traffic matrices and failure scenarios. With TEDAT, 23%∼60% more BA demands can be successfully fulfilled under normal loads. Using data from the 10 Azure cloud services[7],10%∼20% more profit can be retained when failures occur.

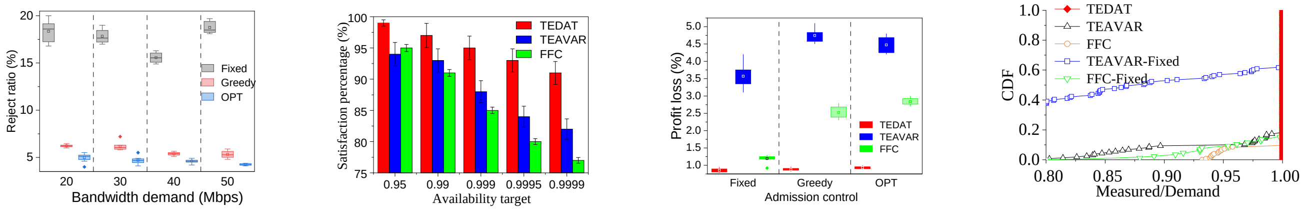
(2) TEDAT achieves a good tradeoff between efficiency and optimality. Compared with the optimal solutions, (i) our admission control algorithm can speed up the admission procedure by 30× at the expense of less than 4% false rejections, (ii) our pruning-augmented scheduling algorithm runs $10^2 \sim 10^4\times$ faster while wasting only 6% bandwidth, and (iii) our greedy failure recovery algorithm can reduce the reaction time by 50×, where profit loss is only about 10% .

(3) TEDAT has a stable performance across different network topologies, demand matrices and routing schemes.

A. Testbed evaluation

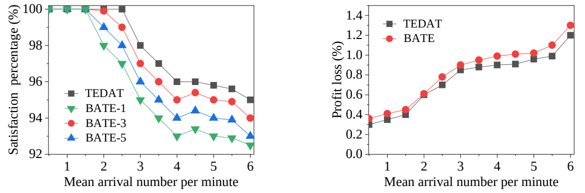
Testbed setup. We build a testbed with 6 servers to emulate a small inter-DC WAN connecting 6 DCs, as shown in Fig. 5. The inter-DC WAN links run at 1Gbps, and we add 100ms delay on each link to emulate a WAN environment. Each server is equipped with 4 Intel Xeon E5-2620 CPUs, 64GB memory and 4 Ethernet NICs, and on each server we start 20 VMs, which are all connected to an Open vSwitch [51]. The VMs run CentOS 7 and use Linux v4.15.6 kernel [40].

[7] API Management [9], App Configuration [10], Application Gateway [11], Application Insights [12], Automation [13], Virtual Machines [18], BareMetal Infrastructure [15], Redis [14],CDN [16], Storage Accounts [17]



(a) Admission control (b) Availability (c) Profit loss after failures (d) Bandwidth ratio

Fig. 6. Testbed evaluation with Poisson demand arrivals.



(a) Availability (b) Profit loss

Fig. 7. TEDAT and BATE comparison.

Every second, we randomly generate an integer p between 0 and 10000 for each link. If $p/10000$ is smaller than the failure probability shown in Fig. 5, we disable the network interface to emulate link failure. Then after x seconds, we enable the network interface to emulate link repair, where default value of x is 3. Each server has enough capacity and there are no negative side effects. We also deploy our controller and brokers on extra VMs. The network agent module in each broker uses Floodlight [24] to control the vSwitch, while the latter monitors link status and reports any failure to the former. If not stated otherwise, we use 4-shortest paths between each source-destination pair as the tunnels in TE algorithms.

Evaluations on continuous demand arrivals. We first conduct experiments where user demands are generated from models used in some latest inter-DC WAN traffic scheduling algorithms [20], [37], [47], [59]. For each source-destination pair, the arrival of user bandwidth demands follows a Poisson Process (mean number is 2 per minute), and the demand duration follows an exponential distribution (mean is 5 minutes). The demanded bandwidth is uniformly generated between 10 Mbps and 50 Mbps. Traffic scheduling is performed each minute. The availability targets are randomly chosen from $\{95\%, 99\%, 99.9\%, 99.95\%, 99.99\%\}$, which are similar to the real inter-DC WAN services shown in TABLE II. The refunding ratio are randomly chosen from 3 cloud services (Redis [14],CDN [16], VMs [18]), and we assume a unit price is charged for 1 Mbps. Each experiment lasts 100 minutes and is repeated 50 times, where link failures occur probabilistically.

Traffic scheduling. We evaluate how demands can be correctly admitted by TEDAT. The two baseline algorithms are the optimal admission strategy by solving the optimization problem shown in Section III-B and the step (1) of TEDAT admission control strategy which assumes a *fixed* bandwidth allocation for admitted demands. Fig. 6(a) demonstrates that TEDAT performs closely to the optimal strategy, i.e., their difference is about 1%, while the difference between the *fixed* algorithm and the optimal strategy is at least 10%. Next, we evaluate that once a user demand is admitted, how often its bandwidth availability target can be met. Since we emulate different link failures according to their probabilities in each second, we can measure the bandwidth a user actually uses deviates from its requirement. If such a downward deviation is less than 1%, we regard the bandwidth availability as *satisfied in that second*. Fig. 6(b) shows the overall fraction of satisfaction, under different levels of availability requirements. We note that, FFC-fixed (or TEAVAR-fixed) in the figure represents applying FFC (or TEAVAR) only to demands admitted by the fixed admission control strategy, where the total bandwidth required for the admitted demands is much lower. TEDAT always achieves the highest availability, even compared with FFC-fixed and TEAVAR-fixed. In particular, it has a clear advantage for high availability requirements (e.g., $\geq 99.95\%$).

Failure Recovery. We evaluate when failures do occur and cause BA target violations, how profit loss can be mitigated by our failure recovery scheme. Fig. 6(c) plots the overall profit of TEDAT, FFC and TEAVAR. Due to its hard guarantee on bandwidth availability and its profit maximization, TEDAT can achieve at least 15% more profit than the other two.

We plot in Fig. 6(d), for each algorithm, the ratio of the allocated bandwidth to the admitted demanded bandwidth, where TEAVAR-Fixed and FFC-Fixed denote TEAVAR and FFC with *fixed* admission control algorithm, respectively. The CDF curve shows FFC is too conservative in bandwidth allocation, and fails to allocate proper bandwidth in almost 60% time. On the other hand, although TEAVAR provides bandwidth well, it ignores the diverse availability requirements of different users, and achieves a lower satisfaction ratio than TEDAT.

BATE [61] makes traffic scheduling every x minutes and models the failure recovery as a mixed integer optimization problem. It is hard to decide the value of x in reality, since a small value can lead to network update oscillation while a large one will decrease network utilization. TEDAT is the extension of BATE [61]. We conduct the performance comparison between TEDAT and BATE, where x is chosen from {1,3,5}. Fig. 7(a) illustrates that TEDAT performs up to 10% better than BATE. BATE proposes a greedy algorithm to derive the solution of failure recovery scheme, while TEDAT models the procedure as a LP problem. Fig. 7(b) shows that

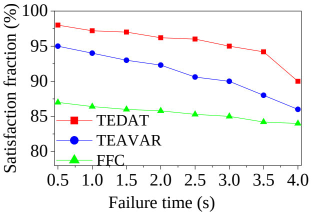


Fig. 8. Different failure time.

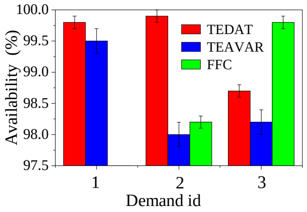


Fig. 9. Bandwidth availability.

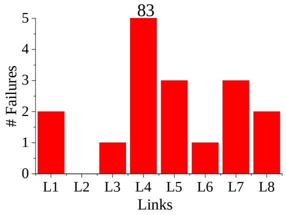


Fig. 10. Total link failures.

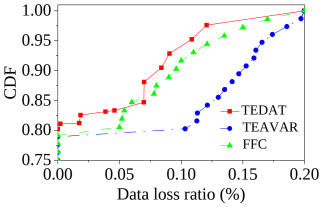
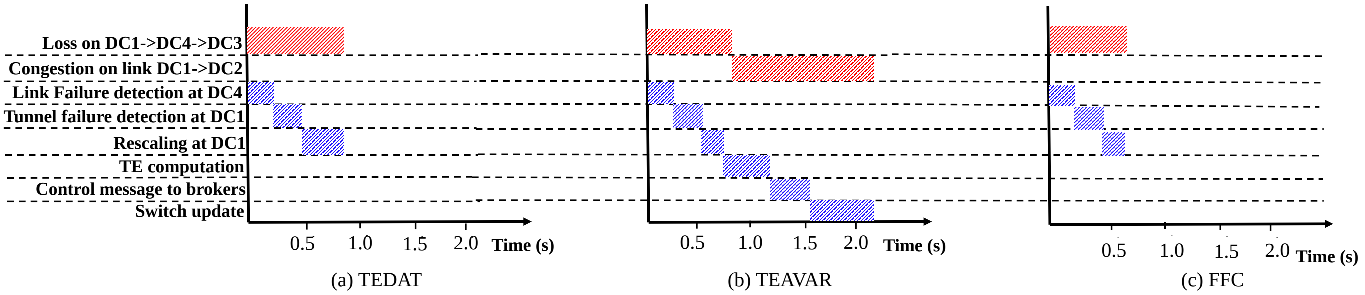


Fig. 11. Data loss ratio comparison.



(a) TEDAT

(b) TEAVAR

(c) FFC

Fig. 12. Events comparison when the link DC4 → DC3 fails, where the x-axis is a time line relative to when the link failure was injected.

TABLE IV
SCHEDULED RESULTS OF DIFFERENT SCHEMES.

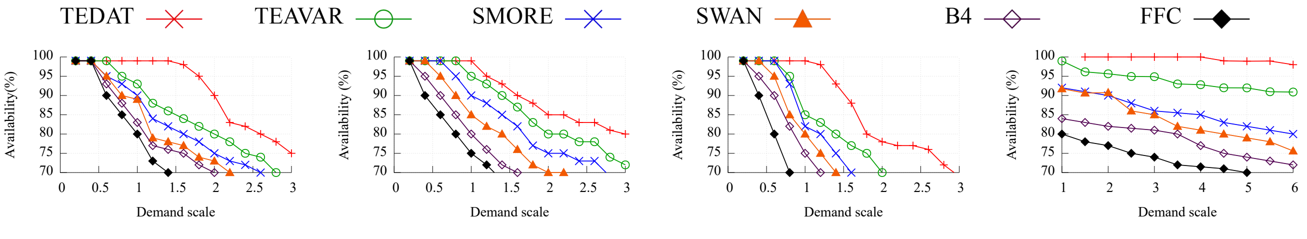
Service	paths	TEDAT	TEAVAR	FFC
demand-1 (99.5%)	DC1→DC2→DC3	0	500	0
	DC1→DC4→DC3	1000	500	250
	DC1→DC2→DC5→DC4→DC3	0	0	0
	DC1→DC4→DC5→DC2→DC3	0	0	0
demand-2 (99.9%)	DC1→DC4	0	250	0
	DC1→DC2→DC5→DC4	0	0	0
	DC1→DC2→DC3→DC4	500	0	250
	DC1→DC6→DC5→DC4	0	250	250
demand-3 (95%)	DC1→DC2→DC5	500	500	750
	DC1→DC4→DC5	0	250	0
	DC1→DC6→DC5	1000	750	750
	DC1→DC2→DC3→DC4→DC5	0	0	0

TEDAT has about 5% less profit loss than BATE.

Default link failure time is 3 seconds in our evaluation. Fig. 8 demonstrates that TEDAT keeps high competitive for BA targets satisfaction when varying failure time from 0.5s to 4.0 seconds.

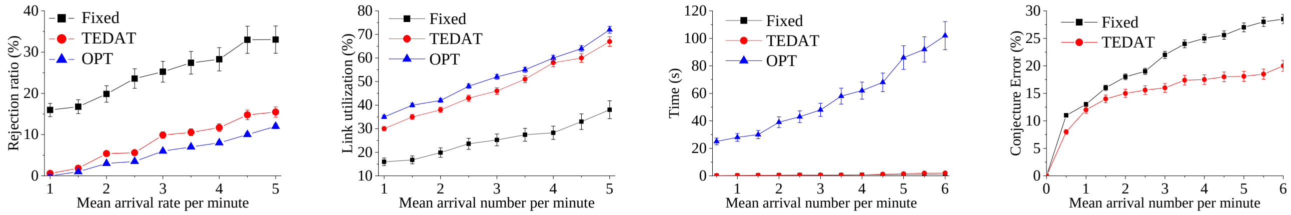
Evaluations on parallel demands. Now we use another example with three parallel user demands to illustrate more details of TEDAT. Demand-1 requires 1000Mbps from DC1 to DC3, demand-2 requires 500Mbps from DC1 to DC4, and demand-3 requires 1500Mbps from DC1 to DC5, with their availability target set as 99.5%, 99.9% and 95%, respectively. We start their traffic simultaneously, assuming all of them have been admitted, and their bandwidth on each path, as shown in TABLE IV, is determined by different TE algorithms. The experiment lasts 100s and is repeated by 100 times. Fig. 9 shows the percentage of time each bandwidth availability demand is satisfied, using the same method as in Fig. 6(a), i.e, for each second, a gap of more than 1% bandwidth downward deviation means the demand is not satisfied in that slot. It shows that all the three demands can reach their availability targets under TEDAT, while TEAVAR and FFC may fail for some users. With an investigation on the bandwidth allocation result in TABLE IV, we can see that, FFC reserves too much bandwidth for failure recovery, so that demand-1 never gets enough bandwidth (250 Mbps allocated v.s. 1500 Mbps demanded), and its achieved bandwidth availability is always 0. Even it allocates enough bandwidth for demand-2, the achieved bandwidth availability (98.2%) is still lower than required (99.9%). On the other hand, TEAVAR does not make a good match between the link failure probability and the availability users ask for. For example, for demand-2, which needs the highest level of availability (99.9%), TEAVAR still allocates 250 Mbps on link L4, which has the highest failure probability (1%) [8]. On the contrary, TEDAT matches demands and links well, and does not use L4 for demand-2. Data loss due to failures is measured according to statistics reported by *iperf* and switches. As shown in Fig. 11, TEDAT and FFC have a slight loss caused by scheduling when failure occurs, while TEAVAR has the highest loss, because it might also have congestion after rescaling besides scheduling data loss. We conduct experiments to show the behaviors of TEDAT and other traffic engineering schemes when the link DC4 → DC3 fails. Fig. 12 shows the results, where the x-axis is a time line relative to when the link failure was injected and the y-axis is the events that might happen when the link fails. The shadowed blocks area denote the start and end of the event. Fig. 12(a) illustrates DC4 and DC1 will take about 0.8s to detect the congestion and DC1 will use about 0.3s to rescale with the surviving tunnels. The packet loss time of tunnel DC1 → DC4 → DC3 is about 0.8s. Fig. 12(b) demonstrates that TEAVAR will have about 0.8s of packet loss on tunnel DC1 → DC4 → DC3 and has about 1.6s congestion on link DC1 → DC2 before the new allocation results are configured successfully.

[8] In Fig. 10, we plot the actual number of failures that occur in the 100 experiments, where L4 fails most frequently.



(a) IBM (b) B4 (c) ATT (d) FITI

Fig. 13. TEDAT to various TE schemes under different topologies, where the performance of TEDAT doesn't depend on particular network.



(a) Rejection ratio (b) Link utilization (c) Admission delay (d) Conjecture error

Fig. 14. Admission control results in simulations.

TABLE V
NETWORK TOPOLOGIES USED IN THE SIMULATIONS.

Topology Name	#Nodes	#Links
IBM	18	48
B4	12	38
ATT	25	112
FITI	14	32

Compared with TEAVAR, TEDAT does not need to compute the allocation results after failures since its failure recovery scheme has already derived the backup allocation results. FFC has the fastest restore rate, as shown in Fig. 12(c), it has only about 0.7s congestion loss. However, compared with TEDAT, FFC is unable to guarantee the bandwidth requirement of demand-1 (as TABLE IV shown).

B. Simulations

Simulation setup. We conduct simulations on four real network topology, including B4 [33], ATT [20], IBM [43] and FITI (a national-level backbone). TABLE V shows the topology [9]. We have collected 200 matrices for each topology. We simulate link failures according to a Weibull distribution with its shape $k = 8$ and scale $\lambda = 0.6$, which matches Fig. 1(b). The availability targets are randomly chosen from $\{0\%, 90\%, 95\%, 99\%, 99.9\%, 99.95\%, 99.99\%\}$, which are similar to the real inter-DC WAN services shown in TABLE II. We generate the demand workload in a similar way to that in the testbed. In default, the required bandwidth in each user demand is randomly drawn from the traffic metrics with a proper scale down factor$s$ [10], so that between each source-destination pair, multiple users can be served simultaneously. The refunding ratio are randomly chosen from 10 Azure cloud services (API Management [9], App Configuration [10], Application Gateway [11], Application Insights [12], Automation [13], Virtual Machines [18], BareMetal Infrastructure [15], Redis [14],CDN [16], Storage Accounts [17]). In our simulations, besides FFC and TEAVAR, we also compare against several other TE algorithms, including SWAN [31], SMORE [43] and B4 [33]. They have not explicitly considered availability, but pay attention to total throughput, link utilization or user fairness. We assume at most one link failure (i.e., no concurrent failures) in FFC, use 99.9% (which is the maximum value in the user demands) as the default availability target in TEAVAR, and let SWAN maximize the total throughput of all users. Each experiment is repeated 20 times by default, and the error bar paints the maximal, average and minimal value.

Evaluation results. Nowadays, some ISP networks are designed with worst-case assumptions about failures, so topologies might be over-provisioned. We firstly assume BA demands arrive simultaneously and then scale up the average matrices by a factor s [20], [43], [46], where the y-axis denotes the total probabilities of qualified scenarios for all demands. Similar to [20], the *availability* is calculated by running a post-processing simulation in which we induce failure scenarios according to their probability of occurrence and attempt to send the entirety of all the BA demand through the network. The sum of the probabilities for scenarios where all the BA demands are fully satisfied reflects the availability in that experiment. Fig. 13 shows the consistent trend under various topologies: TEDAT performs about 20%, 25%, 30%, 40% and 50% better than TEAVAR, SMORE, SWAN, B4 and FFC, respectively. Specially, the advantage of TEDAT is more obvious when there are resource competition (e.g., $s = 3$). This picture also demonstrates that the performance of TEDAT

[9] For B4, ATT and IBM, we get their topology, link capacities and traffic matrices from the authors of TEAVAR [20], and for FITI, we conduct a direct measurement on it but uses just part of it.

[10] We use a factor of 5, and a mean arrival number around 5 in our simulation corresponds to the normal network load.

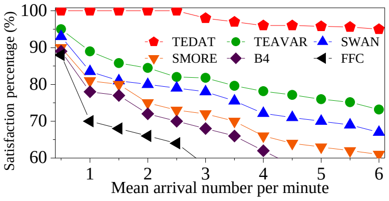


Fig. 15. TEDAT v.s. other TEs.

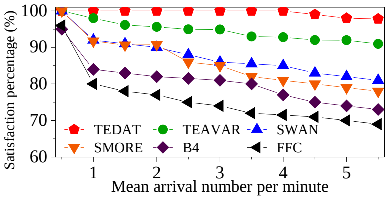


Fig. 16. *fixed* admission control.

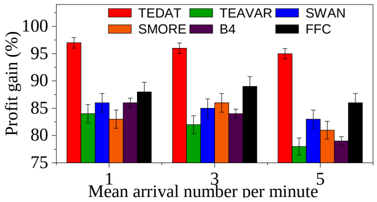


Fig. 17. Profit gain after failures.

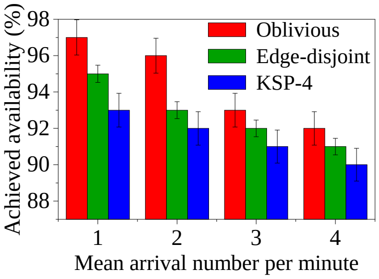


Fig. 18. Different routing schemes.

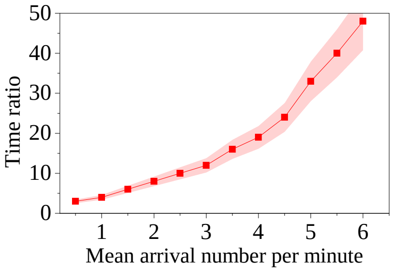


Fig. 19. Acceleration.

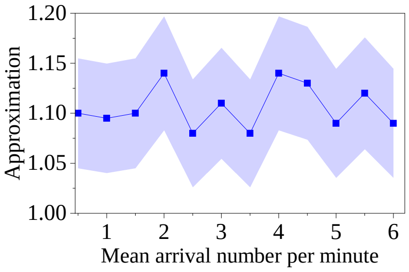


Fig. 20. Approximate ratio.

doesn't depend on particular network topology and traffic. Therefore, we take IBM trace as the example and perform the experiments in the following evaluations.

Next, we assume the arrivals of BA demands follow a Poisson Process, where the mean BA arrival number varies from 1 to 6 in each minute. The duration of each demand follows an exponential distribution, and the mean duration corresponds to 1000 minutes. With the settings, each simulation lasts 150,000 minutes (corresponding to 100 days).

Fig. 14 compares, under different demand arrival rates, the admission results of TEDAT against the optimal strategy and the *fixed* one, i.e., step (1) in TEDAT. Fig. 14(a) shows that, TEDAT rejects at most 4% more demands than the optimal solution, but accepts up to 20% more demands than the Fixed. It can also utilize at least 10% higher bandwidth than the Fixed (when mean arrival number per minute is 1), as shown in Fig. 14(b). We also qualify their efficiency by measuring the admission control delay, and Fig. 14(c) demonstrates that, TEDAT runs at least 30× faster than directly solving the MILP optimization problem, and always finishes within 1 second. Fig. 14(d) shows up to 10% more demands are falsely conjected by *fixed* than TEDAT.

We then compare the traffic scheduling capability of TEDAT against FFC, TEAVAR, SWAN, SMORE and B4. The methodology is similar to the post-processing simulation in TEAVAR [20], where we simulate different failure scenarios according to their probabilities, and in each scenario we record the demands that can be satisfied. If the *achieved availability*, i.e., the total posterior probabilities of *qualified* scenarios where a user's bandwidth target is met, is larger than the user's availability target, then the BA demand is *satisfied*. We plot the overall percentage of satisfied BA demands under each arrival rate (averaged across all simulations) in Fig. 15. TEDAT nearly always achieves a satisfaction ratio around 100%, with a leading margin of at least 23% (with respect to TEAVAR) under a normal arrival rate (mean arrival number per minute is 6 in the figure). To further demonstrates TEDAT's advantage in matching stringent availability requirements with reliable links, we further augment each TE algorithm with the *fixed* admission control scheme. The satisfaction ratios are plotted in Fig. 16, where TEDAT still performs at least 10% better than the others (when mean arrival number per minute is 6). Fig. 17 shows the average profit after failures occur in the network. Due to its consideration of pricing and refunding, TEDAT is able to retain 10%∼20% more profit than the others. Remember that our scaling down factor is 5, so our summarized key results are for a normal network load, where mean arrival number per minute is 5∼6. We note that, under heavier loads, TEDAT performs even better than its competitors, but we regard that as less possible in reality.

Optimality and Robustness. By default, we use the K-shortest paths in the network as tunnels for transmission. To test the robustness of TEDAT's scheduling algorithm, we further replace K-shortest path routing with oblivious routing [43] and edge disjoint path routing [56], which have been used by other TE algorithms. The BA demand satisfaction ratios are plotted in Fig. 18, where there are only minor difference between different tunnel selection algorithms. Scheduling based on oblivious routing works slightly better than the other two, because it finds diverse and low-stretch paths and avoids link over-utilization. In Section III-B, we propose a solution to achieve a good trade-off between optimal and latency. Fig. 19 shows our solution can achieve a speedup by at least 50×. Next, we compute the approximation of our greedy traffic scheduling algorithm. The approximation ratio is defined as the number of the BA demands admitted by optimal solution to the BA demands admitted by Algorithm 1. The performance loss shown in Fig. 20, illustrates that our solution has less than 20 % performance loss, when compared with the optimal solution.

VI. Related work

Optimizing WAN performance is a big challenge. One important topic is on network utilization or fairness. For example, early studies focus more on tuning parameters of widely used routing protocols, such as OSPF [25] and MPLS [23], [38], for given traffic matrices. Recently, Software defined network (SDN) based technologies, including SWAN [31], B4 [32], [33], Bwe [42] and OWAN [37], rely on a centralized view to optimize bandwidth allocations. Pretium [34] combines dynamic pricing with traffic engineering for inter-DC bandwidth, but it does not provide guarantee on network bandwidth. Network scheduling schemes [39], [58] also use SDN technology to decide the priority of traffic. These work mainly consider aggregated traffic in a macro level, while TEDAT handles traffic demands of users. As more applications are deployed in cloud or data centers, many work study how to provide performance guarantee for intra-DC or inter-DC user traffic, including flow deadline [55], [59], [60], flow rate [35], [45], traffic engineering [31]–[34], [39], etc. However, they do not provide adequate mechanisms to deal with potential or actual failures.

Network failures (or uncertainties) have also been considered in various aspects for large scale network environments, including design data center networks [28] and optical networks [26], stochastic models [19], [49] and failure recovery methods [53], [57]. TEDAT studies both proactive and reactive traffic engineering schemes to take network failures into account, so that violations on service level agreements can be avoided or mitigated. As far as we know, FFC [46] and TEAVAR [20] are two pieces of work that are most close to TEDAT, in the sense that they also try to provide certain performance guarantee for inter-DC WAN, even under failures. However, they have not taken into account the heterogeneity and competitions of user demands, and the economic interests of service providers.

VII. Conclusion

We present TEDAT, a framework that attempts to satisify the heterogeneous bandwidth demands of different users or applications under network failures. TEDAT is composed of traffic scheduling and failure recovery. They explicitly take failure probabilities into account, while the last component also deals with real failures, all in an efficient way. Our extensive evaluations show that, it can achieve close to optimal performance guarantee and economic profit.

References

[1] Aliababa. Data transmission service level agreement. https://www.alibabacloud.com/help/zh/doc-detail/50079.htm, 2020.

[2] Aliababa. Short message service (sms) service level agreement. https://www.alibabacloud.com/help/zh/doc-detail/155130.htm, 2020.

[3] O. Alipourfard, J. Gao, J. Koenig, C. Harshaw, A. Vahdat, and M. Yu. Risk based planning of network changes in evolving data centers. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, pages 414–429, New York, NY, USA, 2019. ACM.

[4] Amazon. Amazon appflow service level agreement. https://aws.amazon.com/cn/appflow/sla/, 2020.

[5] Amazon. Aws database migration service (aws dms) service level agreement. https://aws.amazon.com/cn/dms/sla/, 2020.

[6] Amazon. Amazon compute service level agreement. https://aws.amazon.com/compute/sla/?nc1=h_ls, 2021.

[7] aryaka. Aryaka private wan. https://www.aryaka.com, 2020.

[8] Azure. Azure active directory domain services. https://azure.microsoft.com/en-us/support/legal/sla/active-directory-ds/v1_0/, 2021.

[9] Azure. Sla for api management. https://azure.microsoft.com/en-us/support/legal/sla/api-management/v1_5/, 2021.

[10] Azure. Sla for app configuration. https://azure.microsoft.com/en-us/support/legal/sla/app-configuration/v1_0/, 2021.

[11] Azure. Sla for application gateway. https://azure.microsoft.com/en-us/support/legal/sla/application-gateway/v1_2/, 2021.

[12] Azure. Sla for application insights. https://azure.microsoft.com/en-us/support/legal/sla/application-insights/v1_2/, 2021.

[13] Azure. Sla for automation. https://azure.microsoft.com/en-us/support/legal/sla/automation/v1_1/, 2021.

[14] Azure. Sla for azure cache for redis. https://azure.microsoft.com/en-us/support/legal/sla/cache/v1_1/, 2021.

[15] Azure. Sla for baremetal infrastructure. https://azure.microsoft.com/en-us/support/legal/sla/baremetal-infrastructure/v1_0/, 2021.

[16] Azure. Sla for content delivery network. https://azure.microsoft.com/en-us/support/legal/sla/cdn/v1_0/, 2021.

[17] Azure. Sla for storage accounts. https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_5/, 2021.

[18] Azure. Sla for virtual machines. https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/, 2021.

[19] Y. Bi and A. Tang. Uncertainty-aware optimization for network provisioning and routing. pages 1–6, 2019.

[20] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira. Teavar: Striking the right utilization-availability balance in wan traffic engineering. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, pages 29–43, New York, NY, USA, 2019. ACM.

[21] cato. Cato managed services. https://www.catonetworks.com, 2020.

[22] C. Chekuri, S. Khanna, and F. Shepherd. The all-or-nothing multicommodity flow problem. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 156–165, Sept. 2004. Proceedings of the 36th Annual ACM Symposium on Theory of Computing ; Conference date: 13-06-2004 Through 15-06-2004.

[23] A. Elwalid, C. Jin, S. Low, and I. Widjaja. Mate: Mpls adaptive traffic engineering. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 3, pages 1300–1309 vol.3, 2001.

[24] floodlight. Floodlight controller. https://github.com/floodlight/floodlight, 2020.

[25] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[26] M. Ghobadi and R. Mahajan. Optical layer failures in a large backbone. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, pages 461–467, New York, NY, USA, 2016. ACM.

[27] P. Gill, N. Jain, and N. Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 350–361, New York, NY, USA, 2011. ACM.

[28] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat. Evolve or die: High-availability design principles drawn from googles network infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, 2016.

[29] Gurobi. Gurobi is a powerful mathematical optimization solver. https://www.gurobi.com, 2020.

[30] Y. Harchol, D. Bergemann, N. Feamster, E. Friedman, A. Krishnamurthy, A. Panda, S. Ratnasamy, M. Schapira, and S. Shenker. A public option for the core. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM' 20, pages 377–389, New York, NY, USA, 2020. Association for Computing Machinery.

[31] C. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*, 2013.

[32] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, K. N. B., C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev, S. Padgett, F. Rabe, S. Ray, M. Tewari, M. Tierney, M. Zahn, J. Zolla, J. Ong, and A. Vahdat. B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan.

In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, pages 74–87, New York, NY, USA, 2018. ACM.

[33] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a globally-deployed software defined wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 3–14, New York, NY, USA, 2013. ACM.

[34] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache. Dynamic pricing and traffic engineering for timely inter-datacenter transfers. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 73–86, New York, NY, USA, 2016. ACM.

[35] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, A. Greenberg, and C. Kim. Eyeq: Practical network performance isolation at the edge. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, pages 297–311, Lombard, IL, 2013. USENIX.

[36] C. Jiang, S. Rao, and M. Tawarmalani. Pcf: Provably resilient flexible routing. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '20, pages 139–153, New York, NY, USA, 2020. ACM.

[37] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, and J. Rexford. Optimizing bulk transfers with software-defined optical wan. In *Proceedings of the 2016 ACM SIGCOMM Conference*, SIGCOMM '16, pages 87–100, New York, NY, USA, 2016. ACM.

[38] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, SIGCOMM '05, pages 253–264, New York, NY, USA, 2005. ACM.

[39] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula. Calendaring for wide area networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 515–526, New York, NY, USA, 2014. ACM.

[40] Kermel. Linux kernel. http://cdn.kernel.org/pub/linux/kernel/v4.x/, 2020.

[41] S. S. Krishnan and R. K. Sitaraman. Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs. In *Proceedings of the 2012 Internet Measurement Conference*, IMC '12, pages 211–224, New York, NY, USA, 2012. ACM.

[42] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, M. Robin, A. Siganporia, S. Stuart, and A. Vahdat. Bwe: Flexible, hierarchical bandwidth allocation for wan distributed computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 1–14, New York, NY, USA, 2015. ACM.

[43] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé. Semi-oblivious traffic engineering: The road not taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 157–170, Renton, WA, Apr. 2018. USENIX Association.

[44] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.

[45] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, and P. Sharma. Application-driven bandwidth guarantees in datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 467–478, New York, NY, USA, 2014. ACM.

[46] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter. Traffic engineering with forward fault correction. In *Proceedings of the 2014 ACM Conference on SIGCOMM*, SIGCOMM '14, pages 527–538, New York, NY, USA, 2014. ACM.

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

[48] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, pages 197–210, New York, NY, USA, 2017. Association for Computing Machinery.

[49] D. Mitra and Q. Wang. Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management. *IEEE/ACM Trans. Netw.*, 13(2):221–233, Apr. 2005.

[50] Openflow. sdn and openflow. https://tools.ietf.org/html/rfc7426\#page-23, 2020.

[51] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, K. Amidon, and M. Casado. The design and implementation of open vswitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 117–130, Oakland, CA, May 2015. USENIX Association.

[52] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[53] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford. Network architecture for joint failure recovery and traffic engineering. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '11, pages 97–108, New York, NY, USA, 2011. ACM.

[54] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage. California fault lines: Understanding the causes and impact of network failures. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 315–326, New York, NY, USA, 2010. ACM.

[55] B. Vamanan, J. Hasan, and T. Vijaykumar. Deadline-aware datacenter tcp (d2tcp). *SIGCOMM Comput. Commun. Rev.*, 42(4):115–126, Aug. 2012.

[56] B. Vidalenc, L. Noirie, L. Ciavaglia, and E. RENAULT. Dynamic risk-aware routing for ospf networks. In *IEEE International Symposium on Integrated Network Management*, 2013.

[57] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang. R3: Resilient routing reconfiguration. In *Proceedings of the ACM SIGCOMM 2010 Conference*, SIGCOMM '10, pages 291–302, New York, NY, USA, 2010. ACM.

[58] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron. Better never than late: Meeting deadlines in datacenter networks. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 50–61, New York, NY, USA, 2011. ACM.

[59] H. Zhang, K. Chen, W. Bai, D. Han, C. Tian, H. Wang, H. Guan, and M. Zhang. Guaranteeing deadlines for inter-datacenter transfers. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery.

[60] H. Zhang, X. Shi, X. Yin, F. Ren, and Z. Wang. More load, more differentiation– a design principle for deadline-aware congestion control. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 127–135, 2015.

[61] H. Zhang, X. Shi, X. Yin, J. Wang, Z. Wang, Y. Guo, and T. Lan. Boosting bandwidth availability over inter-dc wan. In *Proceedings of the 17th International Conference on emerging Networking EXperiments and Technologies*, pages 297–312, 2021.

Han Zhang received the B.S. degree in Computer Science and Technology from JiLin University and Ph.D. in Tsinghua University. He is now working in the Institute for Network Sciences and Cyberspace, Tsinghua university. His research concerns computer networks, network security. He is a member of IEEE. He has published more than 40 papers in his area.

Xingang Shi received the B.S. degree from Tsinghua University and the PhD degree from The Chinese University of Hong Kong. He is now working in the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include network measurement and routing protocols.

Zhiliang Wang received the B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University, China in 2001, 2003 and 2006 respectively. Currently he is an Associate Professor in the Institute for Network Sciences and Cyberspace at Tsinghua University. His research interests include formal methods and protocol testing, next generation Internet, network measurement.

Jilong Wang received the Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University in 2000. He is currently a Professor with Tsinghua University. His research focuses on network measurement, location-oriented network, and SDN systems, network security.

Yingya Guo received the B.S degree in computer science from Xiamen University, China, in 2013 and Ph.D. degree from Tsinghua University. She is currently an assistant professor in the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. Her research interests include traffic engineering, routing optimization and Software Defined Networking.

HaiJun Geng received the B.E, M.E. and Ph.D degrees from Yantai University, Capital Normal University and Tsinghua University, in 2008, 2011 and 2015 respectively. He is now working in the School of Software Engineering, Shanxi University. His research interests include future Internet architecture and largescale Internet routing.

XiaYin received the B.E., M.E. and Ph.D. degrees in computer science from Tsinghua University in 1995, 1997 and 2000 respectively. She is a Full professor in Department of Computer Science and Technology at Tsinghua University. Her research interests include future Internet architecture, formal method, protocol testing and large-scale Internet routing..

Boosting Bandwidth Availability Over Inter-DC WAN

Han Zhang
Tsinghua University
Beijing, China
zhhan@tsinghua.edu.cn

Xingang Shi*
Tsinghua University
Beijing, China
shixg@cernet.edu.cn

Xia Yin
Tsinghua University
Beijing, China
yxia@tsinghua.edu.cn

Jilong Wang
Tsinghua University
Beijing, China
wjl@cernet.edu.cn

Zhiliang Wang
Tsinghua University
Beijing, China
wzl@cernet.edu.cn

Yingya Guo
Fuzhou University
Fuzhou, China
guoyingya90@163.com

Tian Lan
George Washington University
Washington D.C., USA
tlan@gwu.edu

ABSTRACT

Inter-DataCenter Wide Area Network (Inter-DC WAN) that connects geographically distributed data centers is becoming one of the most critical network infrastructures. Due to limited bandwidth and inevitable link failures, it is highly challenging to guarantee network availability for services, especially those with stringent bandwidth demands, over inter-DC WAN. We present BATE, a novel Traffic Engineering (TE) framework for *bandwidth availability* (BA) provision, which aims to ensure that each bandwidth demand must be satisfied with a stipulated probability, when subjected to the network capacity and possible failures of the inter-DC WAN. The three core components of BATE, i.e., admission control, traffic scheduling and failure recovery, are formulated through different mathematical models and theoretically analyzed. They are also extensively compared against state-of-the-art TE schemes, using a testbed as well as real trace driven simulations across different topologies, traffic matrices and failure scenarios. Our evaluations show that, compared with the optimal admission strategy, BATE can speed up the online admission control by 30× at the expense of less than 4% false rejections. On the other hand, compared with the latest TE schemes like FFC and TEAVAR, BATE can meet the bandwidth availability targets for 23%∼60% more demands under normal loads, and when network failure causes BA targets violations.

CCS CONCEPTS

• **Networks** → **Network management**; **Network resources allocation**; **Traffic engineering algorithms**.

KEYWORDS

Availability,Traffic Engineering, WAN, Bandwidth Guarantee, SLA

ACM Reference Format:
Han Zhang, Xingang Shi, Xia Yin, Jilong Wang, Zhiliang Wang, Yingya Guo, and Tian Lan. 2021. **Boosting Bandwidth Availability Over Inter-DC WAN. In *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21), December 7–10, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3485983.3494843**

1 INTRODUCTION

Nowadays, large scale online services such as finance trading, web search, online shopping, online game and video streaming are posing stringent requirements on the availability and flexibility of the network infrastructures, where Inter-DataCenter Wide Area Network (Inter-DC WAN) that connects geographically distributed data centers has been playing a critical role. Many service providers, including Amazon, Google, Microsoft, etc., are providing various optimizations for their global WAN, especially with the help of the emerging software-defined networking techniques [15, 22, 24–27, 30, 32, 35, 36, 39, 51].

Among various optimization targets, high network availability has been, and will continue to be a major focus. On the one hand, it supports critical uninterrupted services and satisfies fastidious users, while on the other hand, it helps to build a good reputation and improves the competitiveness of network providers. However, guaranteeing network availability for services, especially those with stringent bandwidth demands, over inter-DC WAN is very challenging, since failures may arise from various network components, from data plane to control plane, and could happen anytime [21, 22, 47]. For example, Microsoft reports links in their WAN could fail as often as every 30 minutes [39]. Once a link fails, traffic has to be rescaled and rerouted, resulting in transit or long lasting congestions. Such negative impacts on inter-DC WAN services will ultimately translate into monetary loss.

In this paper, we argue that although existing traffic engineering schemes [15, 24, 26, 29, 36, 39, 50] have already factored in network risks and aimed for network availability guarantee, they cannot

[*] Corresponding author.

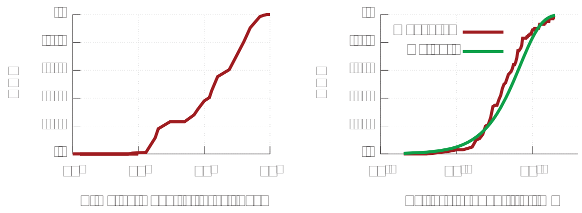
CoNEXT '21, December 7–10, 2021, Virtual Event, Germany
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-9098-9/21/12.
https://doi.org/10.1145/3485983.3494843

meet the above objectives due to three limitations: *First*, most of them [15, 29, 36, 39] typically make a conservative bandwidth allocation, so that even if a failure occurs, surviving paths could be used and the network can still be free from congestion under traffic rerouting. To prevent congestion, links, including those with negligible failure probabilities, must be kept at low utilization, resulting in significant waste of network bandwidth. Although the over provision solution is simple and has been adopted by some existing ISPs, it is highly costly and inefficient. *Second*, existing techniques mainly focus on the availability of the whole network, but ignore the fact that applications' or users' expectations for reliability may vary significantly in practice. Providing reliable bandwidth can be a value-added service for many cloud providers. For example, in B4, the promised bandwidth for its DNS service and Email service should be available 99.99% and 99.95% of the time, due to their stringent availability requirements. If the availability target is violated, user experience will be influenced. A *one-size-fit-all* approach (e.g., TEAVAR [15]) ignoring these heterogeneities cannot support such availability well, and may even hurt critical and uninterruptible applications when there are competitions on bandwidth. *Third*, such heterogeneity and competitions are either not considered by their failure recovery approaches, especially those who allocate bandwidth aggressively [15, 24]. Therefore, without a systematic optimization framework, services may run into congestion when traffic is rerouted under network failures.

To solve these challenges, in this paper we make the following three **contributions**:

Firstly, we advocate traffic engineering with *bandwidth availability* (*BA*) provision: a BA demand $d = (b_d, \beta_d, t_d^s, t_d^e)$ requests bandwidth b_d for a life duration from t_d^s to t_d^e, and should be guaranteed at least β_d% of the duration, subjected to the network capacity and possible failures. Such demands may differ substantially across users and applications (see Table 1 for real world examples in B4). We show that state-of-the-art traffic engineering schemes fail to meet the heterogeneous bandwidth availability demands, especially under diverse link failure probabilities that may vary by several orders of magnitude (see §2). We note that, although the general concept of bandwidth-based availability has been recognized in some recent TE works (e.g., B4 [25, 26, 35], TEAVAR [15]), their methodologies and evaluations are actually achieving *only a soft guarantee*, i.e., a high ratio of the allocated bandwidth to the negotiated one, while we will provide a **hard guarantee**, i.e., the negotiated bandwidth **must** be met.

Secondly, we design BATE, a novel traffic engineering framework that aims for bandwidth availability provision over inter-DC WAN (see §3). BATE is composed of three core components, i.e., admission control, traffic scheduling and failure recovery. The admission control procedure strikes a balance between efficiency and optimality, so that new demands can be admitted and guaranteed as much as possible with negligible delay. Then based on a Linear Programming (LP) model, our traffic scheduling algorithm allocates bandwidth for the admitted demands over tunnels to guarantee bandwidth availability. A key innovation of solution is that to cope with exponentially increasing complexity due to network size, we propose a pruning method by ignoring certain failure scenarios that hardly happen, i.e., characterized by sufficiently small probabilities.



(a) Failure intervals (b) Empirical link failure

Figure 1: A commercial inter-DC WAN empirical data.

Table 1: Bandwidth Availability targets in B4 [25].

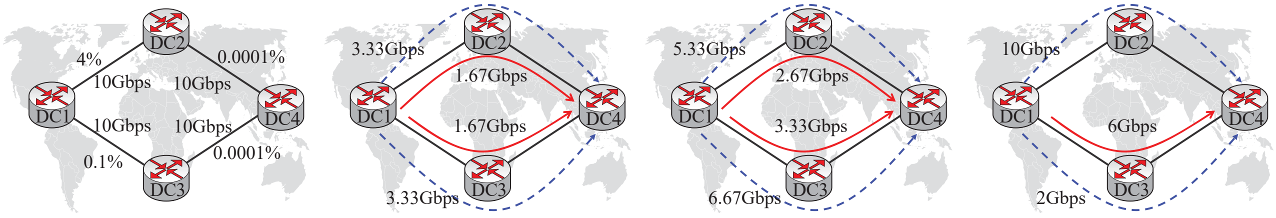
Service	Availability
Search ads, DNS, WWW	99.99%
Photo service, backend, Email	99.95%
Ads database replication	99.9%
Search index copies, logs	99%
Bulk transfer	N/A

At last, we advocate to incorporate the economic interests into rerouting under failures. Based on a Mixed-Integer Linear Programming (MILP) model, the failure recovery procedure pre-computes backup bandwidth allocations and reroutes traffic to minimize the revenue loss due to BA target violations, which is proved to be 2-optimal.

Thirdly, we implement BATE as a real system, including a centralized controller and multiple brokers (one for each DC) (see §4). We conduct extensive experiments using a network testbed as well as trace driven large scale simulations (see §5). We compare BATE with state-of-the-art WAN TE schemes such as TEAVAR [15], SMORE [36], SWAN [24], B4 [26] and FFC [39], across different topologies, traffic matrices and failure scenarios. Our evaluations on real network topologies and traces demonstrate that, BATE can (1) speed up the online admission control by 30× at the expense of a false rejection ratio that is less than 4%; (2) meet the bandwidth availability targets for 23%~60% more demands under normal loads. Under a pricing and SLA violation refunding model that borrows the basic idea from the cloud services (e.g., Amazon Compute [2], Azure Active Directory Domain Service [3]), BATE can retain 10%~20% more profit, when network failure causes BA violations using real data from Azure cloud [3]. To our knowledge, BATE is the first to tackle bandwidth availability provision over inter-DC WAN, where heterogeneities of demands and link failures are systematically taken into account for profit maximization.

2 BACKGROUND AND MOTIVATION

In this section, we first briefly introduce network failures and common availability requirements in inter-DC WAN, then we use an example to show state-of-the-art traffic engineering schemes' limitations in fulfilling such requirements.



(a) Capacity and failure probability (b) FFC (with one failure) (c) TEAVAR (d) BATE

Figure 2: A simple global wan example, where user1 (solid) requires 6Gbps bandwidth for at least 99% of the time and user2 (dash) requires 12Gbps bandwidth for at least 90% of the time, both from DC1 to DC4.

2.1 Network failures and availability

WAN failures are frequent and follow a heavy-tailed distribution. Failures could occur anywhere, from control plane to data plane across the network [15]. They could also last for long durations, as Google reports, more than 80% of the failures last between 10 mins and 100 mins over their B4 network [1, 22], leading to severe performance degradation and revenue loss. Our failure intervals measurement of a commercial inter-DC WAN shown in Figure 1(a) indicates failures are common cases (e.g., more than 80% failure intervals are in less than 6 hours). The empirical failure probability demonstrated in Figure 1(b) shows link failures often follow a *heavy-tailed distribution*, where a small portion of links contribute to most of the failures and the failure rate of a single link can differ by even more than two orders of magnitude. Our measurements also match previous works [20, 21, 47]. Therefore, *network failures, especially their uneven distribution, should be explicitly taken into account by network operators.*

Bandwidth availability guarantee may be beneficial. Availability has attracted major attention both in the industry and research community. A Service Level Objective (SLO) of β% connectivity-based availability specifies that a certain quality of connectivity (i.e., packet loss is below a certain threshold) should be available β% of the time [25]. However, only connectivity-based availability is insufficient. In recent years, there has been a rapid increase in deploying online services (e.g., online videos, online game, online shopping, live broadcast) over clouds. Concurrent with this trend has been a steady rise in bandwidth demand. Many studies have shown that users will quickly abandon sessions if their minimal bandwidth cannot be guaranteed, leading to significant losses in revenue for content providers [34, 41, 45]. Therefore, for a BA demand $d = (b_d, \beta_d, t_d^s, t_d^e)$, formulating the *hard* guarantee such as "demand d's bandwidth b_d for a life duration from t_d^s to t_d^e is guaranteed at least β_d% of the duration" may be beneficial.

Providing a one-size-fit-all network availability is not enough. In recent years, there has been a surging increase in rapid and agile deployment of services over clouds. Many studies have shown that users will quickly abandon sessions if the qualify of service is not guaranteed, leading to significant losses in revenue for content providers [34, 41, 45]. Multiple services might be simultaneously launched over the global infrastructure operated by the same content provider or cloud provider. They might also pose different availability requirements, and will contend for the inter-DC WAN bandwidth. As B4's availability targets [25, 26] shown in Table 1, the minimal availability demands of DNS and logs are 99.99% and 99%, respectively. *Such heterogeneous availability demands cannot be well captured and handled by a one-size-fit-all approach*, where all users get the same level of availability guarantee (e.g, TEAVAR [15] only considers guaranteeing all users' bandwidth at least β% of the time).

2.2 A motivating example for BATE

Now we use a simple example to illustrate why existing traffic engineering algorithms cannot meet the heterogeneous bandwidth availability demands. The toy topology we use is depicted in Figure 2(a), where there are 4 data centers. The links connecting them are annotated with their corresponding capacities as well as failure probabilities. Suppose we have two bandwidth demands for inter-DC transmission from DC1 to DC4, i.e., user1 (solid) requires 6Gbps bandwidth with at least 99% availability, and user2 (dash) requires 12Gbps bandwidth with at least 90% availability. There are two paths from DC1 to DC4, i.e., DC1→ DC2 → DC4, and DC1 → DC3 → DC4, whose available probabilities are $(1 - 4\%) \times (1 - 0.0001\%) = 95.999904\%$ and $(1-0.1\%)\times(1-0.0001\%) = 99.8999001\%$, respectively. We apply FFC [39] and TEAVAR [15], two latest WAN traffic engineering schemes that take network failures into account, to this scenario.

FFC [39] guarantees a total bandwidth from DC1 to DC4 under at most l concurrent node/link failures, and here we simply use $l = 1$. Figure 2(b) shows FFC can support 10Gbps bandwidth from DC1 to DC4 in 99.996% uptime even with one failure (the probability that the two paths fail simultaneously is $(1 - 95.999904\%) \times (1 - 99.8999001\%) = 0.004004092096\%$). User1 and user2 can respectively get 3.34Gbps and 6.66Gbps, which are evenly distributed on the two paths from DC1 to DC4, and neither of their bandwidth demands can be satisfied. This shows *FFC makes a conservative allocation and does not differentiate between paths with different availabilities.* Path (2) has a much smaller failure probability, and lowering its utilization is wasteful.

On the other hand, TEAVAR [15] exploits the different link failure probabilities and maximizes the network utilization, subject to meeting a *single* desired availability. Figure 2(c) illustrates the bandwidth allocation result of TEAVAR, where user1 and user2 can get their demanded 6Gbps and 12Gbps bandwidth, both in about 95.9% of the time. However, this falls below user1's availability demand, i.e., 99%, and will cause BA target violation. This shows *TEAVAR does not consider the heterogeneous user demands on availability.* Since user1 requires a higher availability, it is better to use a path with a lower failure probability.

Our approach: Taking into account the diverse link failure probabilities and user bandwidth availability demands, Figure 2(d) shows a better allocation, where user1 can get 6Gbps over 99.8999001% of the time (via the path that has a lower failure probability) and user2 can get 12Gbps over 95.999904% of the time (via both paths). Therefore, both users' bandwidth availability demands are satisfied.

Table 2: Key Notations for BATE**.**

Input Variables	
$G(V,E)$	inter-DC WAN with nodes V and Links E
$\mathbf{z} \in Z$	a network failure scenario in the scenario set
$p_{\mathbf{z}}$	the probability that a failure scenario $\mathbf{z}$ occurs
$k \in K$	a s(ource)-d(est) pair in the set of all s-d pairs
T_k	the set of tunnels for a s-d pair k
$d = (\mathbf{b}_d, \beta_d)$	a BA demand d, requiring bandwidth $\mathbf{b}_d$ with availability β_d, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, \mathbf{b}_d^2, \ldots >$ of bandwidth demands over all s-d pairs
$D, \hat{D}$	the set of arrived and admitted demands[1]
t	a tunnel for transmitting traffic[2]
u_t^e	whether tunnel t passes link $e \in E$
c_e, c_t	the remaining capacity on link e or a tunnel t
$v_t^{\mathbf{z}}$	whether tunnel t is available under scenario $\mathbf{z}$
$w_e^{\mathbf{z}}$	whether link e is available under scenario $\mathbf{z}$
g_d	the charge for serving demand d
Output Variables	
f_d^t	bandwidth allocated for demand d over tunnel t
r_d	profit (after refunding) for demand d

3 BATE **FRAMEWORK**

In this section, we discuss the details of BATE, which contains three parts, i.e., admission control, traffic scheduling and failure recovery. Main notations are summarized in Table 2. The framework intends to achieve the following objectives:

- **High admission ratio and low admission latency:** Bandwidth availability demands might arrive at anytime. The system should be able to efficiently accommodate as many BA demands as possible under the constraint of network capacity and failure probabilities, as this would increase service agility and might bring more revenue.
- **Guarantee availability for allocated bandwidth:** The system should be able to guarantee the availability of demands according to link failure probabilities, as this would reduce potential penalties and retain a good reputation in the long term. This can be achieved by making a good match between demands on higher availability and paths which fail less probably.
- **Automatic and economical failure recovery:** If any link failure really happens, the system should reroute traffic away from that link, while minimizing any possible collateral damage and revenue loss, i.e., congestion due to contention caused by the rerouted traffic.

3.1 Abstraction of bandwidth availability

In reality, a bandwidth availability demand asking for inter-DC WAN bandwidth resources could from any application spanning multiple data centers in a private cloud. Our abstractions on network failure scenarios and bandwidth availability demands are as follows.

Network failure scenario model: The inter-DC WAN is modeled as a directed graph $G(V,E)$, where the set of nodes V represent the data centers, and the set of links E represent directed links between them. A network scenario $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{|E|}\}$ is a vector of link states, where each element $\mathbf{z}_i \in \{0,1\}$ denotes whether the i-th link is up ($\mathbf{z}_i = 1$) or down ($\mathbf{z}_i = 0$). We assume network operators can use historical data to estimate the failure probability x_i for this link, which are statistically independent [3]. Let Z denote the network scenario set, then the expected probability that a network scenario $\mathbf{z} \in Z$ will happen is given by [15], i.e.,

$$p_{\mathbf{z}} \quad = \quad \prod_{i=1}^{|E|} \left(\mathbf{z}_i \times (1 - x_i) + (1 - \mathbf{z}_i) \times x_i\right)$$

Take the simple inter-DC WAN topology in Figure 2 as an example, where $E = \{e_1, e_2, e_3, e_4\}$. Network scenario $\mathbf{z} = \{1,1,0,1\}$ means e_1 ,e_2, e_4 are working fine and e_3 is down. The expected availabilities of e_1, e_2, e_3, e_4 are 96%, 99.9999%, 99.9%, 99.9999%, respectively. Then the probability of $\mathbf{z}$ is $p_{\mathbf{z}} = p_{\{1,1,0,1\}} = 0.96 \times 0.999999 \times 0.001 \times 0.999999 \simeq 0.000959998$.

BA demand model: Let K denote the set of all source-destination (s-d) DC pairs. A bandwidth availability demand d is in the form of $(\mathbf{b}_d, \beta_d)$, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, ..., \mathbf{b}_d^k, ... >$ of bandwidth demands on each s-d pair $k \in K$ [4] and β_d is its bandwidth availability target.

BA provision model: Similar to [15, 24, 39], BATE also adopts tunnel-based forwarding . For each source-destination node pair $k \in K$ of the inter-DC WAN, we pre-compute a set of tunnels T_k with different routing schemes (e.g., k-shortest paths, edge disjoint paths [49], oblivious routing [36], etc.). Each tunnel $t \in T_k$ contains a sequence of links and u_t^e denotes whether tunnel t passes a specific link $e \in E$ or not. Let D and $\hat{D}$ represent *arrived* demands and *admitted* demands, respectively. Given a new demand, the admission control scheme (see § 3.2) will decide whether to admit it and makes a first-time bandwidth allocation for it. Then the traffic scheduling scheme (see § 3.3) will allocate bandwidth f_d^t for each admitted demand $d \in \hat{D}$ over tunnel t periodically.

We use $v_t^{\mathbf{z}}$ to denote whether tunnel t is available (i.e., $v_t^{\mathbf{z}} = 1$) or not (i.e., $v_t^{\mathbf{z}} = 0$) under network scenario $\mathbf{z}$. Given a BA demand $d = (\mathbf{b}_d, \beta_d)$, an allocation result $\{f_d^t\}$ and a network scenario $\mathbf{z}$, for *every* s-d pair k, if the total allocated bandwidth on available tunnels under $\mathbf{z}$, i.e., $\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}$, is no less than the bandwidth demand $\mathbf{b}_d^k$, then we call $\mathbf{z}$ a *qualified scenario* for allocation $\{f_d^t\}$ with respect to demand d, and denote this by $\mathbf{z} \propto < d, \{f_d^t\} >$. The sum of the probabilities of all such qualified scenarios, i.e., $\sum_{\mathbf{z} \propto < d, \{f_d^t\} >} p_{\mathbf{z}}$, is the expected probability that the bandwidth target $\mathbf{b}_d$ will be satisfied. Now we can formally define when a bandwidth availability

[3] The strong assumption does not affect the network scenario model.

[4] Here we omit the start and end time of this demand, but they will be implicitly considered in our online admission and traffic scheduling.

demand is satisfied: a demand d is satisfied by an allocation $\{f_d^t\}$, if and only if

$$\sum_{\mathbf{z} \propto < d, \{f_d^t\} >} p_{\mathbf{z}} \geq \beta_d$$

.

If a failure indeed occurs, our failure recovery scheme (see § 3.4) will try to reroute traffic that is affected by this failure. If any availability target is violated, a refund will be given back to the customer according to our recommending model, and we use r_d to denote the profit (after refunding) for serving demand d.

Algorithm 1: Admission Conjecture

```
Input: Input parameters shown in Table 2
Output: Whether the new demand can be admitted.
1  while true do
2      $d = \arg_{d' \in D} min\{\sum_{k \in K} b_{d'}^k \times \beta_{d'}\}$;
3      for $k \in K$ do
4          if $b_d^k >$ remaining capacity of s-d pair $k$ then
5              return False;
6          $T_k' = T_k$;
7          while $b_d^k > 0$ do
8              $t = \arg_{t \in T_k'} \min\{c_t * p_t\}$;
9              $f_d^t = \min\{c_t, b_d^k\}$;
10             $T_k' = T_k' \setminus t$;
11             $s_d = s_d * p_t$;
12             $b_d^k = b_d^k - f_d^t$;
13             update the remaining capacities of links and
                 tunnels;
14     if $s_d < \beta_d$ then
15         return False;
16     $D = D \setminus d$;
17 return True;
```

3.2 Admission control

User demands are served in a first-come-first-service (FCFS) manner without preemption. When a new demand d arrives, we have $D = \hat{D} \cup d$. The optimal admission strategy would try to accommodate as many demands as possible: if every demand in D can meet its availability target, then d should be admitted, otherwise, it should be rejected. This can be modeled as a 0-1 Mixed-Integer Linear Programming (MILP) problem which maximizes the number of demands whose availability targets can be satisfied. Appendix A shows the formulation of this problem and it can be proved to be NP-hard by reducing the all-or-nothing multi-commodity flow problem [16] to a special case of it. However, in order to support agile deployment of new applications and services, user demands should be admitted as fast as possible, while the time needed to exactly solve this NP-hard problem may be prohibitive. Therefore, we need a better tradeoff between efficiency and optimality. The final admission control strategy we use is as follows:

(1) When a new demand d arrives, we *fix* the bandwidth allocation for all admitted demands in $\hat{D}$, then we check whether d can be satisfied by the remaining network capacity and failure probability. If the answer is positive, then admit d and make a first-time bandwidth allocation for it.

(2) Otherwise, run a greedy algorithm (Algorithm 1) to *conjecture* whether the admitted demands can potentially be rescheduled to accommodate d. If the answer is positive, then admit d and make a temporary bandwidth allocation for it, using the remaining network capacity as far as needed [5].

(3) If d still cannot be accommodated, reject the demand.

The greedy algorithm tries to conjecture, in an efficient way, whether an allocation strategy satisfying all demands (i.e., including d) exists. It works iteratively as follows. In each iteration, it finds the demand which has the smallest product of bandwidth target and availability target (i.e., $\sum_{k \in K} \mathbf{b}_d^k \times \beta_d$) at first (line 2), and tries to allocate bandwidth for each of its s-d pairs one by one. If the remaining network capacity cannot satisfy this demand, we will give up (line 4-5). Otherwise, it allocates tunnel bandwidth for this demand, where a tunnel with a smaller product of remaining capacity and availability has a higher priority (line 7-13). After this, if the availability target cannot be roughly satisfied, it will give up (line 14-15), otherwise, it will go for the next iteration.

The time complexity of Algorithm 1 is $O(|D| * |K| * max(|T_k|))$. *It is also worth to note that, there is no false positive in conjectures made by Algorithm 1*, as indicated by the following theorem, whose proof can be found in Appendix B:

Theorem 1. *If a new demand d can be admitted by Algorithm 1, then there must exist an allocation result $\{f_d^t\}$ to satisfy the bandwidth availability targets of all demands $D = \hat{D} \cup d$.*

3.3 Traffic scheduling

For *admitted* demands (including the newly admitted ones), we carry out traffic scheduling to further optimize the bandwidth allocation periodically (e.g., every 10 minutes). We aim to guarantee all bandwidth targets with least network resource. Specifically,

$$\sum_{t \in T_k} f_d^t \geq \mathbf{b}_d^k, \quad \forall d \in \hat{D}, k \in K \tag{1}$$

For an s-d pair k of BA demand d, we use $R_{dk}^{\mathbf{z}}$ to denote the ratio of the effective bandwidth under network scenario $\mathbf{z}$ to the demanded bandwidth, which is defined as:

$$R_{dk}^{\mathbf{z}} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in \hat{D}, k \in K, \mathbf{z} \in Z \tag{2}$$

Here, our consideration is tunnel t might be unavailable (i.e., $v_t^{\mathbf{z}} = 0$) under network scenario $\mathbf{z}$, but if $R_{dk}^{\mathbf{z}} \geq 1$ holds, then this scenario is still *qualified*, i.e.,

$$\mathbf{z} \propto < d, \{f_d^t\} > \quad \Leftrightarrow \quad \forall k, R_{dk}^{\mathbf{z}} \geq 1$$

To meet the availability target, we should guarantee the total probability of the qualified scenarios is no less than the availability target, i.e., $\sum_{R_{dk}^{\mathbf{z}} \geq 1} p_{\mathbf{z}} \geq \beta_d$. However, this condition will result in

[5] It's possible that the temporarily allocated bandwidth falls below the demanded bandwidth, but a new allocation strategy satisfying all demands does exist (see Theorem 1), and will be computed later in our periodical traffic scheduling.

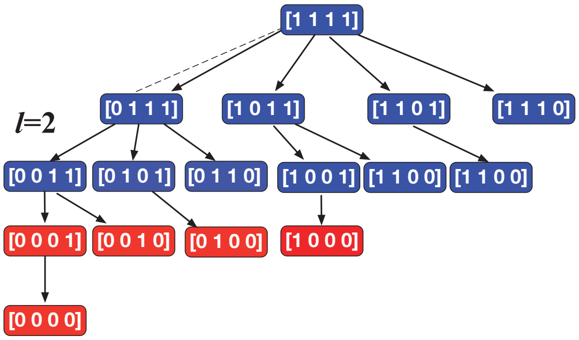
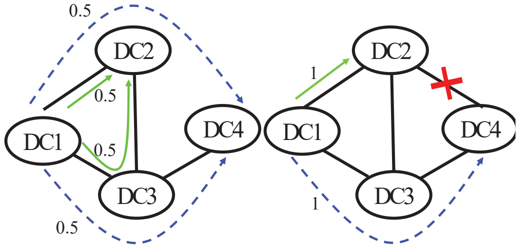


Figure 3: A pruning example.



(a) Original allocation (b) Backup allocation

Figure 4: Failure recovery.

an Mixed Integer Linear Programming (MILP) problem, and we choose to relax it and solve the following Linear Programming (LP) problem.

Let $B^{\mathbf{z}}_d$ denote the lower bound of $R^{\mathbf{z}}_{dk}$ over all s-d pairs, i.e.,

$$B^{\mathbf{z}}_d \le R^{\mathbf{z}}_{dk}, \quad \forall d \in \hat{D}, k \in K, \mathbf{z} \in Z \tag{3}$$

We can use $B^{\mathbf{z}}_d \times p_{\mathbf{z}}$ to roughly represent the availability that can be achieved under network scenario $\mathbf{z}$, which is set to be no smaller than the availability target, i.e.,

$$\sum_{\mathbf{z} \in Z} B^{\mathbf{z}}_d \times p_{\mathbf{z}} \ge \beta_d, \quad \forall d \in \hat{D} \tag{4}$$

Besides, the bandwidth allocation result f^t_d should be non-negative and limited by the network capacity, i.e.,

$$f^t_d \ge 0, \quad \forall d \in D, k \in K, t \in T_k \tag{5}$$

and

$$\sum_{d \in \hat{D}} \sum_{k \in K, t \in T_k} f^t_d u^e_t \le c_e, \quad \forall e \in E \tag{6}$$

Finally, our traffic scheduling will minimize the overall bandwith allocated to all admitted demands under the above constraints, i.e.,

$$\begin{aligned} &minimize \sum_{d \in \hat{D}, k \in K, t \in T_k} f^t_d \\ &s.t. (1), (2), (3), (4), (5), (6) \end{aligned} \tag{7}$$

Solving this LP problem directly is possible, but as it considers every possible network scenario, the complexity will increase exponentially with the network size. For instance, the B4 topology [26] has 12 nodes and 38 links, so there are totally 2^{38} network failure scenarios (when only link failures considered). Therefore, an important question is *how to effectively reduce the problem size without affecting the result significantly*. TEAVAR [15] prunes a scenario if its probability is smaller than a threshold. However, such a threshold is difficult to choose, and an enumeration of all possible scenarios is still needed. Instead, we use a much faster pruning method, where at most y (from 1 to 4 in our experiments) concurrent link failures will be considered, and all the remaining scenarios will be aggregated into one special *unqualified* scenario. In this way, the set of scenarios Z and the corresponding probabilities $\{p_{\mathbf{z}}\}$ can be efficiently computed. Figure 3 depicts an example of a simple network. The root node denotes all links are available (i.e., [1111]). The i-th layer denotes concurrent i link failures could happen. Network scenarios (red) located in layer 3, 4 are pruned (the root node is regarded as layer 0).

3.4 Failure recovery

When failures occur and any tunnel becomes unavailable, traffic can be redistributed across the surviving tunnels. To reduce recovery time, BATE proactively computes backup allocation strategies for potential failure scenarios, so that the surviving tunnels can be used immediately, and packet loss can be mitigated [6].

For example, in Figure 4, there are two users, and the link capacity is 1 everywhere. One user requests a bandwidth of 1 from DC1 to DC2, while the other one requests a bandwidth of 1 from DC1 to DC4. Figure 4(a) shows the original bandwidth allocation when no failures occur, and Figure 4(b) depicts the backup allocation pre-computed for a failure of link DC2⟶DC4.

Bandwidth availability, even well planned, cannot always be guaranteed due to network failures, and this ultimately hurts the reputation of the cloud providers. In reality, many popular cloud services (e.g., Amazon Compute Service [2], Azure Active Directory Domain Service [3]) will refund their customers in case their agreed SLAs are violated. For example, the Amazon Compute Service SLA [2] defines that they will provide 10% refund if the achieved availability (e.g., monthly uptime percentage) is between 99.99% and 99.0%. Although this practice is specified for scenarios other than inter-DC WAN, its principles and policy designs might provide good hints for inter-DC WAN services. We borrow the SLA violation refunding idea from the popular cloud services (e.g., Amazon Compute Service [2], Azure Active Directory Domain Service [3]) and advocate to use economic interests to guide our design of rerouting under failures as follows.

For a specific network scenario $\mathbf{z}$ (where one link failure occurs) in consideration, the ratio of allocated bandwidth to a user's demanded bandwidth is [7]:

$$R_{dk} = \frac{\sum_{t \in T_k} f^t_d v^{\mathbf{z}}_t}{\mathbf{b}^k_d}, \quad \forall d \in \hat{D}, k \in K \tag{8}$$

[6] Here we only consider backup allocations for one link, while this scheme can be easily extended to deal with concurrent failures.

[7] This is the same as equation (2), but we omit the superscript $\mathbf{z}$ of $R^{\mathbf{z}}_{dk}$.

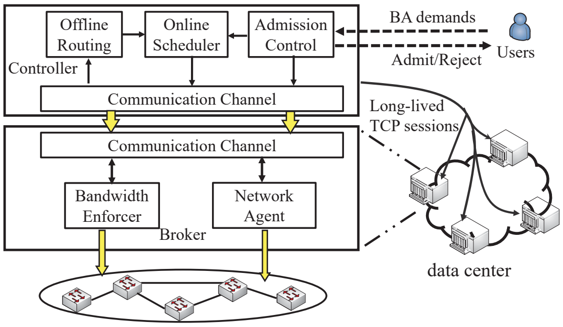


Figure 5: BATE **System.**

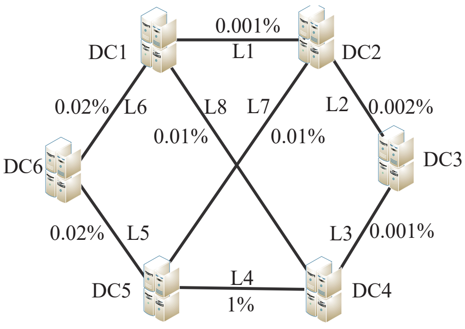


Figure 6: Testbed topology.

If for every k, R_{dk} is larger than its demand (i.e., $R_{dk} \geq 1$), then there is no problem since the demanded availability is still satisfied. However, if any R_{dk} falls below 1, then the corresponding bandwidth availability (BA) target will be violated. For simplicity, here we assume a simple pricing and refunding model, where the charge for serving a user demand d is g_d, and if the bandwidth availability target cannot be guaranteed, a fraction μ_d of g_d will be refunded. We use r_d to denote the profit of demand d with refunding, such that

$$r_d = \begin{cases} g_d & \text{if } R_{dk} \geq 1 \text{ for every } k \in K \\ (1-\mu_d)g_d & \text{Otherwise} \end{cases}$$

We use an auxiliary integer variable $y_d \in \{0,1\}$ to denote the violation condition, where $y_d = 1$ means no violation. Then the profit r_d can be rewritten as

$$\begin{aligned} y_d \in\ & \{0,1\}, && \forall d \in \hat{D} \\ r_d =\ & g_d \times \Big(y_d + (1-\mu_d) \times (1-y_d)\Big), && \forall d \in \hat{D} \\ R_{dk} <\ & M \times y_d + 1 - y_d, && \forall d \in \hat{D}, k \in K \\ R_{dk} \geq\ & y_d, && \forall d \in \hat{D}, k \in K \end{aligned} \tag{9}$$

where M is a constant large enough (e.g., at least larger than the upper bound of R_{dk}).

Besides, the bandwidth allocation result f_t^d should be nonnegative and limited by the available network capacity. Let $w_e^{\mathbf{z}}$ denote whether link e is available under scenario $\mathbf{z}$, then we have

$$f_d^t \geq 0, \quad \forall d \in \hat{D}, k \in K, t \in T_k \tag{10}$$

and

$$\sum_{d \in \hat{D}} \sum_{k \in K, t \in T_k^{\mathbf{z}}} f_d^t u_t^e \leq c_e \times w_e^{\mathbf{z}}, \quad \forall e \in E \tag{11}$$

Finally, the failure recovery scheme tries to maximize the total profit (after refunding) by

$$\begin{aligned} & maximize \sum_{d \in \hat{D}} r_d \\ & s.t. (8), (9), (10), (11) \end{aligned} \tag{12}$$

The above Mixed-Integer Linear Programming (MILP) problem can be proved to be NP-hard, and the proof details can be found in Appendix C. To efficiently solve this problem, we further propose a 2-approximation greedy algorithm. The key idea is to prioritize demands by the ratio of profit to the allocated bandwidth in a non-increasing order. Due to space limitations, the detailed algorithm, its complexity analysis, and the proof on its optimality, are put into Appendix D.

4 SYSTEM IMPLEMENTATION

We have implemented BATE on the Linux platform. Figure 5 shows the whole system architecture, which contains one controller, multiple brokers (one for each DC). The controller is responsible for most decision work of BATE, including admission control, traffic scheduling, and failure recovery. The brokers and switches are responsible for bandwidth enforcement. The system works as follows: When a user submits a demand to the controller, the admission control module determines whether the demand can be admitted or not (see § 3.2). If the demand is admitted, this module will also allocate its demanded bandwidth on appropriate paths for the first time, and notify the brokers for enforcement. The online scheduler module performs traffic scheduling (see § 3.3) periodically to further optimize the availability expectation of all active demands. In addition, for potential link failures, it also pre-computes backup allocation strategies that will be activated if any link failure indeed happens (see § 3.4). These central decisions are distributed to the brokers for bandwidth enforcement. The brokers in each DC monitor link status and bandwidth consumption, report these statistics to the central controller, and ask the switches to enforce rate.

Controller is the brain of the whole system. It is responsible for allocating WAN level bandwidth, and orchestrates all activities with a global view. The four main components in Controller are as follows. (1) Offline Routing. This module maintains the WAN level network topology, and computes TE tunnels between each node pair (i.e., $T_k, \forall$s-d pair $k \in K$), using certain routing algorithms (oblivious routing [36], k-shortest path [24], etc.). These tunnels are used by the admission control module and the online scheduler module as input variables; (2) Admission Control. When a BA demand is submitted, this module uses the admission control algorithm (see § 3.2) to reject it, or accept it and allocate bandwidth over the tunnels in nearly real-time. The results are sent to the corresponding brokers; (3) Online Scheduler. Periodically, this module performs traffic scheduling (see § 3.3) according to the bandwidth availability demands submitted by users, so that the availability can be optimized in a probabilistic sense. It also pre-computes backup allocation (see § 3.4) for some potential link failures. For each user demand, the normal bandwidth and backup bandwidth allocated over each tunnel (i.e., f_d^t) are then sent to the corresponding brokers. In addition, our system also supports several other TE algorithms, e.g., SWAN [24], FFC [39] and TEAVAR [15]; (4) Communication Channel. This module is responsible for communication with brokers, where we use long-lived TCP connections to avoid unnecessary delay. Also, controller failures can be remedied by using multiple replications, where the master controller is elected by the Paxos [37] algorithm.

Broker takes care of the data center it resides in. It consists of three modules: (1) Bandwidth Enforcer. It receives the bandwidth allocation results (i.e., f_d^t) from controller, sends them to the corresponding switches connecting with hosts, and limits the actual traffic rate in each tunnel in case something is wrong on the end hosts; (2) Network Agent. We use commodity SDN switches at data center edges to connect DCs into an inter-DC wan. The network agent runs in a SDN controller (we use floodlight [18]), and uses the OpenFlow [43] protocol to installs and updates forwarding rules on the switches in that DC. To reduce rule complexity, our system uses a label-based forwarding scheme, where the first 12 bits of a VxLAN ID represent different demands, and the last 12 bits represent different tunnels. Therefore, 4096 demands and 4096 tunnels can be supported simultaneously, and this can be further expanded if necessary. In this way, a flow (i.e., traffic corresponding to a BA demand) is marked with a label at the ingress switch, and the succeeding switches use this label for forwarding. Group tables in the switch pipelines are used for flow splitting (i.e., traffic corresponding to a BA demand can be split into multiple sub-flows and transmitted in multiple tunnels). Besides, the network agent also tracks the network topology, reports any change or failure to the central Controller module, and monitors the actual traffic rate; (3) Communication Channel. This component is responsible for communication with the central Controller.

5 EVALUATION

In this section, we use a small testbed and large scale trace driven simulations to evaluate the performance of BATE. On the testbed, we also implement another two state-of-the-art TE algorithms that consider network availability, i.e., FFC [39] and TEAVAR [15]. For simulation, we implement more TE algorithms, including SWAN [24], SMORE [36] and B4 [26]. Our main results are as follows:

(1) BATE consistently outperforms latest TE algorithms under various topologies, traffic matrices and failure scenarios. With BATE, 23%∼60% more BA demands can be successfully fulfilled under normal loads. Using data from the 10 Azure cloud services[8],10%∼20% more profit can be retained when failures occur.

(2) BATE achieves a good tradeoff between efficiency and optimality. Compared with the optimal solutions, (i) our admission control algorithm can speed up the admission procedure by 30× at the expense of less than 4% false rejections, (ii) our pruning-augmented scheduling algorithm runs $10^2 \sim 10^4\times$ faster while wasting only 6% bandwidth, and (iii) our greedy failure recovery algorithm can reduce the reaction time by 50×, where profit loss is only about 10% .

(3) BATE has a stable performance across different network topologies, demand matrices and routing schemes.

5.1 Testbed evaluation

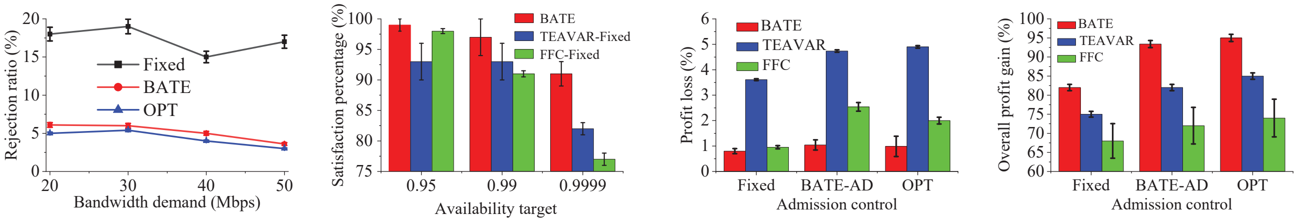
Testbed setup. We build a testbed with 6 servers to emulate a small inter-DC WAN connecting 6 DCs, as shown in Figure 6. The inter-DC WAN links run at 1Gbps, and we add 100ms delay on each link to emulate a WAN environment. Each server is equipped with 4 Intel Xeon E5-2620 CPUs, 64GB memory and 4 Ethernet NICs, and on each server we start 20 VMs, which are all connected to an Open vSwitch [44]. The VMs run CentOS 7 and use Linux v4.15.6 kernel [33]. Every second, we randomly generate an integer p between 0 and 10000 for each link. If $p/10000$ is smaller than the failure probability shown in Figure 6, we disable the network interface to emulate link failure. Then after x seconds, we enable the network interface to emulate link repair, where default value of x is 3 (performance comparison is shown in Appendix E). Each server has enough capacity and there are no negative side effects. We also deploy our controller and brokers on extra VMs. The network agent module in each broker uses Floodlight [18] to control the vSwitch, while the latter monitors link status and reports any failure to the former. If not stated otherwise, we use 4-shortest paths between each source-destination pair as the tunnels in TE algorithms.

Evaluations on continuous demand arrivals. We first conduct experiments where user demands are generated from models used in some latest inter-DC WAN traffic scheduling algorithms [15, 30, 40, 53]. For each source-destination pair, the arrival of user bandwidth demands follows a Poisson Process (mean number is 2 per minute), and the demand duration follows an exponential distribution (mean is 5 minutes). The demanded bandwidth is uniformly generated between 10 Mbps and 50 Mbps. Traffic scheduling is performed each minute. The availability targets are randomly chosen from {95%, 99%, 99.9%, 99.95%, 99.99%}, which are similar to the real inter-DC WAN services shown in Table 1. The refunding ratio are randomly chosen from 3 cloud services (Redis [9],CDN [11], VMs [13]), and we assume a unit price is charged for 1 Mbps. Each experiment lasts 100 minutes and is repeated 50 times, where link failures occur probabilistically.

Admission control. We evaluate how demands can be correctly admitted by BATE. The two baseline algorithms are the optimal admission strategy by solving the optimization problem shown in Appendix A and the step (1) of BATE admission control strategy which assumes a *fixed* bandwidth allocation for admitted demands. Figure 7(a) demonstrates that BATE performs closely to the optimal strategy, i.e., their difference is about 1%, while the difference between *fixed* algorithm and the optimal strategy is at least 10%.

Traffic scheduling. We evaluate, once a user demand is admitted, how often its bandwidth availability target can be met. Since we emulate different link failures according to their probabilities in each second, we can measure the bandwidth a user actually uses deviates from its requirement. If such a downward deviation is less than 1%, we regard the bandwidth availability as *satisfied in that second.* Figure 7(b) shows the overall fraction of satisfaction, under different levels of availability requirements (i.e., 95%, 99% and 99.99%). We note that, FFC-fixed (or TEAVAR-fixed) in the figure represents applying FFC (or TEAVAR) only to demands admitted

[8] API Management [4], App Configuration [5], Application Gateway [6], Application Insights [7], Automation [8], Virtual Machines [13], BareMetal Infrastructure [10], Redis [9],CDN [11], Storage Accounts [12]



(a) Admission control (b) Traffic scheduling (c) Profit loss after failures (d) Overall profit gain

Figure 7: Testbed evaluation with Poisson demand arrivals.

Table 3: Scheduled results of different schemes.

Service	paths	BATE	TEAVAR	FFC
demand-1 (99.5%)	DC1⟶DC2⟶DC3	0	500	0
	DC1⟶DC4⟶DC3	1000	500	250
	DC1⟶DC2⟶DC5⟶DC4⟶DC3	0	0	0
	DC1⟶DC4⟶DC5⟶DC2⟶DC3	0	0	0
demand-2 (99.9%)	DC1⟶DC4	0	250	0
	DC1⟶DC2⟶DC5⟶DC4	0	0	0
	DC1⟶DC2⟶DC3⟶DC4	500	0	250
	DC1⟶DC6⟶DC5⟶DC4	0	250	250
demand-3 (95%)	DC1⟶DC2⟶DC5	500	500	750
	DC1⟶DC4⟶DC5	0	250	0
	DC1⟶DC6⟶DC5	1000	750	750
	DC1⟶DC2⟶DC3⟶DC4⟶DC5	0	0	0

Table 4: Network topologies used in the simulations.

Topology Name	#Nodes	#Links
IBM	18	48
B4	12	38
ATT	25	112
FITI	14	32

by the fixed admission control strategy, where the total bandwidth required for the admitted demands is much lower. BATE always achieves the highest availability, even compared with FFC-fixed and TEAVAR-fixed. In particular, it has a clear advantage for high availability requirements (e.g., ≥ 99.95%).

Failure Recovery. We evaluate when failures do occur and cause BA target violations, how profit loss can be mitigated by our failure recovery scheme. Figure 7(c) depicts the fraction of profit loss caused by BA targets violations under three different admission control strategies, i.e., fixed, BATE-AD (which is the strategy BATE uses) and optimal, where baseline for each algorithm is the profit it can achieve when no failures occur. BATE achieves the lowest loss ratio, while FFC also has a low profit loss ratio due to conservative bandwidth allocation, and TEAVAR causes around 5× higher profit loss.

Overall Profit. Figure 7(d) plots the overall profit of BATE, FFC and TEAVAR. Due to its hard guarantee on bandwidth availability and its profit maximization, BATE can achieve at least 15% more profit than the other two.

We plot in Figure 8, for each algorithm, the ratio of the allocated bandwidth to the demanded bandwidth. The CDF curve shows FFC is too conservative in bandwidth allocation, and fails to allocate proper bandwidth in almost 60% of the time. On the other hand, although TEAVAR provides bandwidth well, it ignores the diverse availability requirements of different users, and achieves a lower satisfaction ratio than BATE.

Evaluations on parallel demands. Now we use another example with three parallel user demands to illustrate more details of BATE. In this evaluation, we also compare with another scheme named BATE-TS, i.e, the traffic scheduling part of BATE, with its fast failure recovery scheme abandoned. Demand-1 requires 1000Mbps from DC1 to DC3, demand-2 requires 500Mbps from DC1 to DC4, and demand-3 requires 1500Mbps from DC1 to DC5, with their availability target set as 99.5%, 99.9% and 95%, respectively. We start their traffic simultaneously, assuming all of them have been admitted, and their bandwidth on each path, as shown in Table 3, is determined by different TE algorithms. The experiment lasts 100s and is repeated by 100 times. Figure 9 shows the percentage of time each bandwidth availability demand is satisfied, using the same method as in Figure 7(a), i.e, for each second, a gap of more than 1% bandwidth downward deviation means the demand is not satisfied in that slot. It shows that all the three demands can reach their availability targets under BATE, while TEAVAR and FFC may fail for some users. With an investigation on the bandwidth allocation result in Table 3, we can see that, FFC reserves too much bandwidth for failure recovery, so that demand-1 never gets enough bandwidth (250 Mbps allocated v.s. 1500 Mbps demanded), and its achieved bandwidth availability is always 0. Even it allocates enough bandwidth for demand-2, the achieved bandwidth availability (98.2%) is still lower than required (99.9%). On the other hand, TEAVAR does not make a good match between the link failure probability and the availability users ask for. For example, for demand-2, which needs the highest level of availability (99.9%), TEAVAR still allocates 250 Mbps on link L4, which has the highest failure probability (1%) [9]. On the contrary, BATE matches demands and links well, and does not use L4 for demand-2. Data loss due to failures is measured according to statistics reported by *iperf* and switches. As shown in Figure 11, BATE and FFC have a slight loss caused by scheduling when failure occurs, while TEAVAR has the highest loss, because it might also have congestion after rescaling besides scheduling data loss.

[9] In Figure 10, we plot the actual number of failures that occur in the 100 experiments, where L4 fails most frequently.

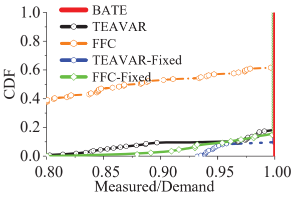


Figure 8: Bw ratio.

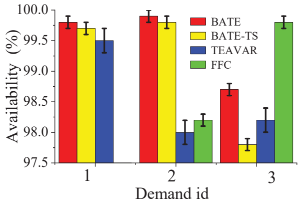


Figure 9: BA availability.

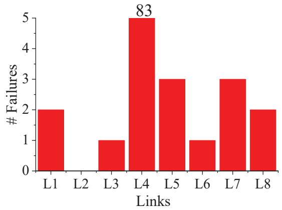


Figure 10: Link failures.

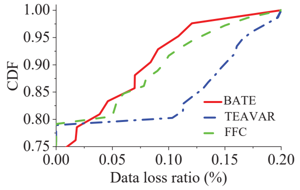
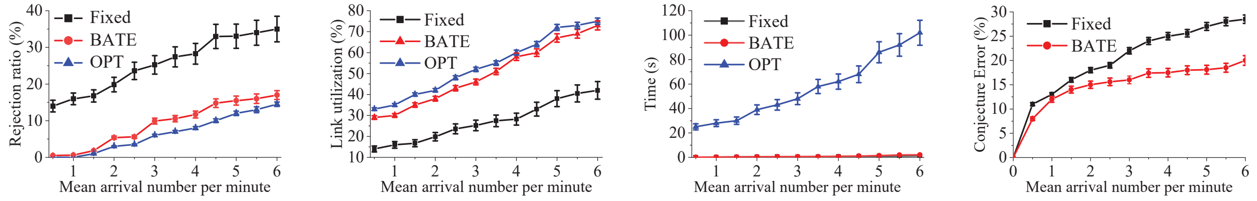


Figure 11: Data loss.



(a) Rejection ratio

(b) Link utilization

(c) Admission delay

(d) Conjecture error

Figure 12: Admission control results in simulations.

5.2 Simulations

Simulation setup. We also conduct simulations on four real network topology, including B4 [26], ATT [15], IBM [36] and FITI [10]. Table 4 shows the topology [11]. We simulate link failures according to a Weibull distribution with its shape $k = 8$ and scale $\lambda = 0.6$, which matches Figure 1(b). We generate the demand workload in a similar way to that in the testbed. The arrivals of BA demands follow a Poisson Process, where the mean BA arrival number varies from 1 to 6 in each minute. The duration of each demand follows an exponential distribution, and the mean duration corresponds to 1000 minutes. The required bandwidth in each user demand is randomly drawn from the traffic metrics (we have collected 200 matrices for each topology) with a proper scale down factor[12], so that between each source-destination pair, multiple users can be served simultaneously. The availability targets are randomly chosen from {0%, 90%, 95%, 99%, 99.9%, 99.95%, 99.99%}, which are similar to the real inter-DC WAN services shown in Table 1. The refunding ratio are randomly chosen from 10 Azure cloud services (API Management [4], App Configuration [5], Application Gateway [6], Application Insights [7], Automation [8], Virtual Machines [13], BareMetal Infrastructure [10], Redis [9],CDN [11], Storage Accounts [12]). In our simulations, besides FFC and TEAVAR, we also compare against several other TE algorithms, including SWAN [24], SMORE [36] and B4 [26]. They have not explicitly considered availability, but pay attention to total throughput, link utilization or user fairness. These TE algorithms will be activated every 10 minutes. We assume at most one link failure (i.e., no concurrent failures) in FFC, use 99.9% (which is the maximum value in the user demands) as the default availability target in TEAVAR, and let SWAN maximize the total throughput of all users. With the above settings, each simulation lasts 150,000 minutes (corresponding to 100 days), and the results achieved by each algorithm on each topology are calculated on 5 independent simulations with different workload traces. Each experiment is repeated 20 times by default, and the error bar paints the maximal, average and minimal value.

Evaluation results. Figure 12 compares, under different demand arrival rates, the admission results of BATE against the optimal strategy and the *fixed* one, i.e., step (1) in BATE. Figure 12(a) shows that, BATE rejects at most 4% more demands than the optimal solution, but accepts up to 20% more demands than the Fixed. It can also utilize at least 10% higher bandwidth than the Fixed (when mean arrival number per minute is 1), as shown in Figure 12(b). We also qualify their efficiency by measuring the admission control delay, and Figure 12(c) demonstrates that, BATE runs at least 30× faster than directly solving the MILP optimization problem, and always finishes within 1 second. Figure 12(d) shows up to 10% more demands are falsely conjected by *fixed* than BATE.

We then compare the traffic scheduling capability of BATE against FFC, TEAVAR, SWAN, SMORE and B4. The methodology is similar to the post-processing simulation in TEAVAR [15], where we simulate different failure scenarios according to their probabilities, and in each scenario we record the demands that can be satisfied. If the *achieved availability*, i.e., the total posterior probabilities of *qualified* scenarios where a user's bandwidth target is met, is larger than the user's availability target, then the BA demand is *satisfied*. We plot the overall percentage of satisfied BA demands under each arrival rate (averaged across all simulations) in Figure 13. BATE nearly always achieves a satisfaction ratio around 100%, with a leading margin of at least 23% (with respect to TEAVAR) under a normal

[10] Future Internet Technology Infrastructure.

[11] For B4, ATT and IBM, we get their topology, link capacities and traffic matrices from the authors of TEAVAR [15], and for FITI, we conduct a direct measurement on it.

[12] We use a factor of 5, and a mean arrival number around 5 in our simulation corresponds to the normal network load.

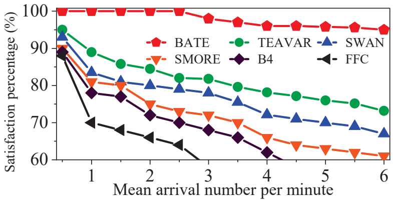


Figure 13: BATE v.s. other TEs.

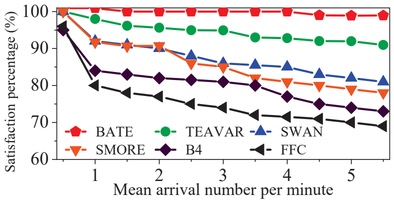


Figure 14: *fixed* admission control.

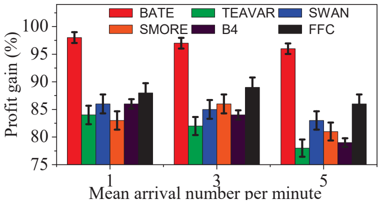


Figure 15: Profit gain after failures.

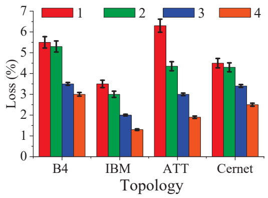


Figure 16: Bandwidth loss.

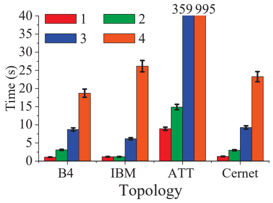


Figure 17: Time.

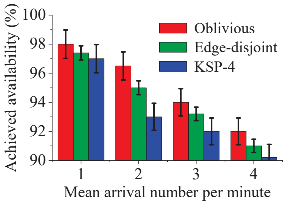


Figure 18: Routing.

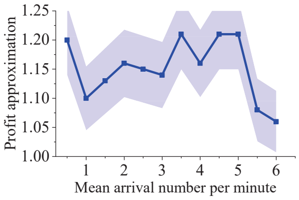


Figure 19: Approx ratio.

arrival rate (mean arrival number per minute is 6 in the figure) [13]. To further demonstrates BATE's advantage in matching stringent availability requirements with reliable links, we further augment each TE algorithm with the *fixed* admission control scheme. The satisfaction ratios are plotted in Figure 14, where BATE still performs at least 10% better than the others (when mean arrival number per minute is 6). Figure 15 shows the average profit after failures occur in the network. Due to its consideration of pricing and refunding, BATE is able to retain 10%~20% more profit than the others. Remember that our scaling down factor is 5, so our summarized key results are for a normal network load, where mean arrival number per minute is 5~6. We note that, under heavier loads, BATE performs even better than its competitors, but we regard that as less possible in reality.

Optimality and Robustness. In traffic scheduling, BATE prune scenarios that are unlikely to happen. We compare the bandwidth allocated by BATE with that allocated by the optimal strategy, i.e., not pruning any scenarios. We calculate the bandwidth loss ratio, as well as the running time of BATE, due to such an optimization, under each topology. Figure 16 plots the relative bandwidth loss ratio, where the highest number of concurrent link failures varies from 1 to 4. This indicates to what extent BATE will trade accuracy for efficiency. We can see the loss ratio is less than 8% even when no current link failures are considered. The corresponding computation time is plotted in Figure 17, where we use Gurobi [23] to solve the pruned LP problem. We can see that even on a large network (e.g., ATT), at most 15 seconds are needed when we consider at most 2 current failures.

By default, we use the K-shortest paths in the network as tunnels for transmission. To test the robustness of BATE's scheduling algorithm, we further replace K-shortest path routing with oblivious routing [36] and edge disjoint path routing [49], which have been used by other TE algorithms. The BA demand satisfaction ratios are plotted in Figure 18, where there are only minor difference between different tunnel selection algorithms. Scheduling based on oblivious routing works slightly better than the other two, because it finds diverse and low-stretch paths and avoids link over-utilization. Finally, we compute the approximation ratio of our greedy failure recovery algorithm, which is defined as dividing the optimal profit by the profit achieved. Figure 19 shows the 2-approximation algorithm achieves a ratio between 1 and 1.25, and the average profit loss is around 10% with a speedup by at least 50× (Appendix E).

6 RELATED WORK

Optimizing WAN performance is a big challenge. One important topic is on network utilization or fairness. For example, early studies focus more on tuning parameters of widely used routing protocols, such as OSPF [19] and MPLS [17, 31], for given traffic matrices. Recently, Software defined network (SDN) based technologies, including SWAN[24], B4[25, 26], Bwe[35] and OWAN[30], rely on a centralized view to optimize bandwidth allocations. Pretium [27] combines dynamic pricing with traffic engineering for inter-DC bandwidth, but it does not provide guarantee on network bandwidth. Network scheduling schemes [32, 52] also use SDN technology to decide the priority of traffic. These work mainly consider aggregated traffic in a macro level, while BATE handles traffic demands of users. As more applications are deployed in cloud or data centers, many work study how to provide performance guarantee for intra-DC or inter-DC user traffic, including flow deadline [48, 53, 54], flow rate [28, 38], traffic engineering [24–27, 32], etc. However, they do not provide adequate mechanisms to deal with potential or actual failures.

Network failures (or uncertainties) have also been considered in various aspects for large scale network environments, including design data center networks [22] and optical networks [20], stochastic models [42] [14] and failure recovery methods [46, 50]. BATE studies both proactive and reactive traffic engineering schemes to take network failures into account, so that violations on service level agreements can be avoided or mitigated. As far as we know, FFC [39] and TEAVAR [15] are two pieces of work that are most close to

[13] BATE leads FFC by around 60%, which is not shown in the figure.

BATE, in the sense that they also try to provide certain performance guarantee for inter-DC WAN, even under failures. However, they have not taken into account the heterogeneity and competitions of user demands, and the economic interests of service providers.

7 CONCLUSION

We present BATE, a framework that attempts to satisify the heterogeneous bandwidth demands of different users or applications under network failures. BATE is composed of three core components, i.e., admission control, traffic scheduling and failure recovery. They explicitly take failure probabilities into account, while the last component also deals with real failures, all in an efficient way. Our extensive evaluations show that, it can achieve close to optimal performance guarantee and economic profit.

8 ACKNOWLEDGEMENT

This work was supported in part by the National Key R&D Program of China under Grant 2018YFB1800401 and in part by National Natural Science Foundation of China under Grants 62002009. This work is part of Future Internet Technology Infrastructure (FITI). We also thank the shepherd Prof. Soudeh as well as all the ACM CoNEXT reviewers who give comprehensive advices to make our paper better.

A THE ADMISSION CONTROL PROBLEM

For an source-destination pair k of BA demand d, let $R^{\mathbf{z}}_{dk}$ denote the ratio of the effective bandwidth under network scenario $\mathbf{z}$ to the demanded bandwidth:

$$R^{\mathbf{z}}_{dk} = \frac{\sum_{t \in T_k} f^t_d v^{\mathbf{z}}_t}{\mathbf{b}^k_d}, \quad \forall d \in D, \mathbf{z} \in \mathbf{z}, k \in K. \tag{13}$$

where $v^{\mathbf{z}}_t$ represents whether tunnel t is available under network scenario $\mathbf{z}$.

For every source-destination pair k, if the total effective bandwidth on all the available tunnels is larger than $\mathbf{b}^k_d$, then the bandwidth target can be met under $\mathbf{z}$, even some tunnels fail. In this situation, network scenario $\mathbf{z}$ can be regarded as *qualified*.

Let $q^{\mathbf{z}}_d$ denote whether scenario $\mathbf{z}$ is qualified (i.e., $q^{\mathbf{z}}_d = 1$) or not (i.e., $q^{\mathbf{z}}_d = 0$) for a BA demand d:

$$q^{\mathbf{z}}_d = \begin{cases} 1 & \text{if } R_{dk} \geq 1 \text{ for every } k \in K \\ 0 & \text{Otherwise} \end{cases}$$

It can be rewritten as

$$\begin{array}{lll} q^{\mathbf{z}}_d \in & \{0, 1\}, & \forall d \in \hat{D}, \mathbf{z} \in Z \\ R^{\mathbf{z}}_{dk} < & M \times q^{\mathbf{z}}_d + 1 - q^{\mathbf{z}}_d, & \forall d \in \hat{D}, k \in K \\ R^{\mathbf{z}}_{dk} \geq & q^{\mathbf{z}}_d, & \forall d \in \hat{D}, k \in K \end{array} \tag{14}$$

where M is a constant larger than the upper bound of $R^{\mathbf{z}}_{dk}$.

The achieved bandwidth availability of demand d is the total probabilities of all *qualified* network scenarios, i.e.,

$$s_d = \sum_{\mathbf{z} \in \mathbf{z}} q^{\mathbf{z}}_d \times p_{\mathbf{z}}, \quad \forall d \in D. \tag{15}$$

Use a_d to represent whether the BA target of d can be satisfied, which also means a_d can be admitted, then we have

$$a_d = \begin{cases} 1 & \text{if } \beta_d \leq s_d \leq 1 \\ 0 & \text{if } 0 \leq s_d < \beta_d \end{cases}$$

which can further written as

$$\begin{array}{lll} a_d \in & \{0, 1\}, & \forall d \in D \\ s_d < & \beta_d \times (1 - a_d) + a_d, & \forall d \in D \\ s_d \geq & \beta_d \times a_d, & \forall d \in D \end{array} \tag{16}$$

In addition, the bandwidth allocation result f^t_d for BA demand d over tunnel t should be non-negative and limited by link capacities, i.e.,

$$f^t_d \geq 0, \quad \forall d \in D, k \in K, t \in T_k. \tag{17}$$

and

$$\sum_{d \in D} \sum_{k \in K, t \in T_k} f^t_d u^e_t \leq c_e, \quad \forall e \in E. \tag{18}$$

Finally, the admission control intends to maximize the total number of accepted demands with the above constraints, i.e.,

$$\begin{array}{l} maximize \sum_{d \in D} a_d \\ s.t. (13), (14), (15), (16), (17), (18) \end{array} \tag{19}$$

B PROOF OF THEOREM 1

Proof. We prove by contradiction. Suppose there is a BA demand that is admitted by Algorithm 1 but the network is unable to satisfy its bandwidth availability. There are two possible cases: (i) network bandwidth is insufficient; (ii) The availability provided by the network is not enough. Case (i) is impossible, because if bandwidth is insufficient (i.e., $\mathbf{b}^k_d$ is larger than the remaining network capacity for s-d pair k) , Algorithm 1 won't admit the demand (Line 4-5). Case (ii) is also impossible, because if the bandwidth availability is smaller than its target (i.e., $s_d < \beta_d$) , Algorithm 1 will reject the demand (Line 14-15). This completes the proof. □

C PROOF OF NP-HARDNESS IN FAILURE RECOVERY

Proof. The all-or-nothing multi-commodity flow problem, which is known to be NP-hard[16], can be regarded as a special case of our failure recovery problem shown in (12). Consider an undirected graph $G = (V, E)$ and a set of k source-destination pairs: $s_1t_1, s_2t_2, ..., s_kt_k$, where each pair s_it_i corresponds to a commodity flow to be sent from the source node s_i to the destination node t_i with demand d_i. Let $\mathcal{P}_i$ denote the path set for pair s_it_i. L_{pe} denotes whether path p goes through link e and f_{ip} is the allocation result of commodity i over path p. The all-or-nothing multi-commodity flow problem tries to find a maximum weight routable set:

$$\begin{array}{ll} maximize \sum_{i=1}^{k} w_i \times y_i & \\ s.t. & \forall e \in E : \sum_{i=1}^{k} \sum_{p \in \mathcal{P}_i} f_{ip} L_{pe} \leq c_e \\ & \forall 1 \leq i \leq k : y_i = \begin{cases} 1 & \sum_{p \in \mathcal{P}_i} f_{ip} \geq d_i \\ 0 & \sum_{p \in \mathcal{P}_i} f_{ip} < d_i \end{cases} \end{array} \tag{20}$$

Where y_i denotes whether commodity flow i is routable. Consider a special case of the failure recovery problem, where $\mu_d = 0$ for every d. This means, if the allocated bandwidth is no less than the demand, then the profit is 1, or the profit is 0 otherwise. We can transform the all-or-nothing multi-commodity flow problem to a special case of our failure recovery problem by regarding the commodities as the BA demands. Therefore, the failure recovery problem is at least as hard as the all-or-nothing multi-commodity flow problem, which is known to be NP-hard. This completes the proof. □

D GREEDY ALGORITHM FOR FAILURE RECOVERY

Our greedy algorithm to solve the MILP failure recovery problem (12) is shown in Algorithm 2, which works as follows. Let F denote the BA demands set that derive full profit (i.e. $h_d = 1$). Firstly, it sorts all the accepted demands $d \in \hat{D}$ in non-increasing order according to the ratio of demand profit to aggregate bandwidth demands, where the aggregate bandwidth demands are derived as $\sum_{k \in K} \mathbf{b}^k_d$ (Line 1). The ordered sequence prefers demands that have large profit and small bandwidth. The algorithm then loops all the ordered admitted demands and tries to allocate resources with remaining network capacity (Line 5-9). If the network is able to support current demand, then add to F (Line 7). If the network is

Algorithm 2: Greedy algorithm for failure recovery

```
Input: Input parameters shown in Table 2, a failure
       scenario $z$
Output: $\{f_d^t\}, F$
1  Sort $d \in \hat{D}$ in non-decreasing order with $\frac{g_d}{\sum_{k \in K} \mathbf{b}_d^k}$;
2  $h_d = 0, \forall d \in \hat{D}$;
3  $F = \{\}$;
4  for $d \in \hat{D}$ do
5      if $z$' s remaining capacity can support $d$ then
6          $h_d = 1$;
7          $F = F \cup d$;
8          Update $\{f_d^t\}$;
9          Update $z$' s remaining network capacity;
10     else
11         if $\sum_{d' \in F} g_{d'} < g_d$ then
12             if network resource allocated to demands in $F$ is
                 able to support $d$ then
13                 release network resource allocated to
                     demands in $F$;
14                 $F = \{d\}$;
15                 Update $\{f_d^t\}, \forall d \in \hat{D}$;
16                 Update z's remaining network capacity;
17                 break;
18         else
19             break;
20 return $\{f_d^t\}, F$
```

unable to support current demand but it has larger profit than the total profit of previous ones in F, the algorithm will try to recycle total resources that are allocated to F and test that if allocating total network resources can support current demand (Line 12-17). If this is true, then algorithm will prefer current demand, otherwise, the algorithm finishes the iteration (Line 18-19). Compared with the bruce force algorithm, Algorithm 2 can derive solution in $O(|\hat{D}||T_k||E|)$, which is Polynomial time. However, it achieves this at the cost of performance loss, which is proven as follows.

Lemma 2. *Algorithm 2 achieves 2-approximation for the MILP failure recovery problem.*

Proof. Algorithm 2 prefers accepted demands according to the following sequence:

$$\frac{g_1}{\sum_{k \in K} \mathbf{b}_1^k} \geq \frac{g_2}{\sum_{k \in K} \mathbf{b}_2^k} \geq .... \tag{21}$$

(21) means the priority of flow pair is decided by the unit value. Withou loss of generality, assume that the network can't transfer the $n+1$ demand, Algorithm 2 will choose $max\{g_{n+1}, \sum_{i=1}^{n} g_i\}$ as the value. Let OPT denote the optimal solution and it is obvious that $\sum_{i=1}^{n} g_i \leq OPT$. Also, we have $\sum_{i=1}^{n+1} g_i \geq OPT$. This holds, since we've already made the density of network as high as possible by the greedy method. If we violate the link capacity constraint and put the $n+1$ demand into links, then links are fulfilled. There is no other way that the density of links are greater than this, that is, the value is greater than OPT. $\sum_{i=1}^{n+1} g_i/2 \leq max\{\sum_{i=1}^{n} g_i, g_{n+1}\}$. Therefore, $OPT/2 \leq max\{\sum_{i=1}^{n} g_i, g_{n+1}\}$. This completes the proof. □

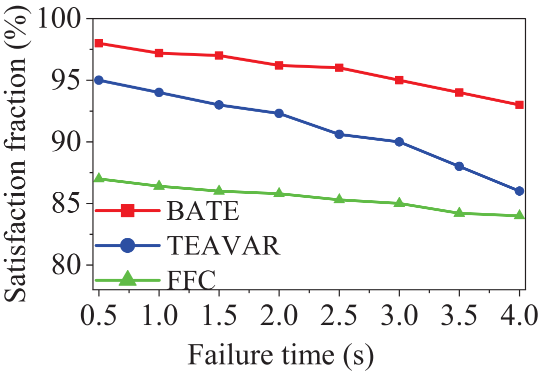


Figure 20: failure time.

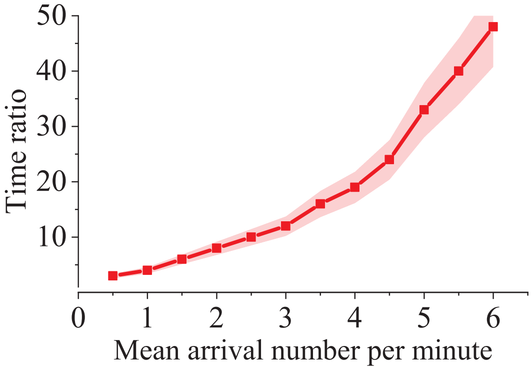


Figure 21: Acceleration.

E MORE EVALUATION RESULTS

Default link failure time is 3 seconds in our evaluation. Figure 20 demonstrates that BATE keeps high competitive for demand BA targets satisfaction when varying failure time from 0.5s to 4.0 seconds.

We compute the time ratio of the optimal solution and our greedy failure recovery algorithm for each scenario. Figure 21 shows that, under normal load (mean arrival number per minute is 5~6), driving the optimal solution by bruce force is at least 50× slower than our greedy algorithm.

REFERENCES

[1] Omid Alipourfard, Jiaqi Gao, Jeremie Koenig, Chris Harshaw, Amin Vahdat, and Minlan Yu. 2019. Risk Based Planning of Network Changes in Evolving Data Centers. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles* (Huntsville, Ontario, Canada) *(SOSP '19)*. ACM, New York, NY, USA, 414–429. https://doi.org/10.1145/3341301.3359664

[2] Amazon. 2021. Amazon Compute Service Level Agreement. https://aws.amazon.com/compute/sla/?nc1=h_ls.

[3] Azure. 2021. Azure Active Directory Domain Services. https://azure.microsoft.com/en-us/support/legal/sla/active-directory-ds/v1_0/.

[4] Azure. 2021. SLA for API Management. https://azure.microsoft.com/en-us/support/legal/sla/api-management/v1_5/.

[5] Azure. 2021. SLA for App Configuration. https://azure.microsoft.com/en-us/support/legal/sla/app-configuration/v1_0/.

[6] Azure. 2021. SLA for Application Gateway. https://azure.microsoft.com/en-us/support/legal/sla/application-gateway/v1_2/.

[7] Azure. 2021. SLA for Application Insights. https://azure.microsoft.com/en-us/support/legal/sla/application-insights/v1_2/.

[8] Azure. 2021. SLA for Automation. https://azure.microsoft.com/en-us/support/legal/sla/automation/v1_1/.

[9] Azure. 2021. SLA for Azure Cache for Redis. https://azure.microsoft.com/en-us/support/legal/sla/cache/v1_1/.

[10] Azure. 2021. SLA for BareMetal Infrastructure. https://azure.microsoft.com/en-us/support/legal/sla/baremetal-infrastructure/v1_0/.

[11] Azure. 2021. SLA for Content Delivery Network. https://azure.microsoft.com/en-us/support/legal/sla/cdn/v1_0/.

[12] Azure. 2021. SLA for Storage Accounts. https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_5/.

[13] Azure. 2021. SLA for Virtual Machines. https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/.

[14] Yingjie Bi and Ao Tang. 2019. Uncertainty-Aware optimization for Network Provisioning and Routing. (2019), 1–6.

[15] Jeremy Bogle, Nikhil Bhatia, Manya Ghobadi, Ishai Menache, Nikolaj Bjørner, Asaf Valadarsky, and Michael Schapira. 2019. TEAVAR: Striking the Right Utilization-Availability Balance in WAN Traffic Engineering. In *Proceedings of the ACM Special Interest Group on Data Communication* (Beijing, China) *(SIGCOMM '19)*. ACM, New York, NY, USA, 29–43. https://doi.org/10.1145/3341302.3342069

[16] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. 2004. The all-or-nothing multicommodity flow problem. *Conference Proceedings of the Annual ACM Symposium on Theory of Computing* (29 Sept. 2004), 156–165. Proceedings of the 36th Annual ACM Symposium on Theory of Computing ; Conference date: 13-06-2004 Through 15-06-2004.

[17] A. Elwalid, C. Jin, S. Low, and I. Widjaja. 2001. MATE: MPLS adaptive traffic engineering. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, Vol. 3. 1300–1309 vol.3.

[18] floodlight. 2020. Floodlight controller. https://github.com/floodlight/floodlight.

[19] B. Fortz and M. Thorup. 2002. Optimizing OSPF/IS-IS weights in a changing world. *IEEE Journal on Selected Areas in Communications* 20, 4 (2002), 756–767.

[20] Monia Ghobadi and Ratul Mahajan. 2016. Optical Layer Failures in a Large Backbone. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica, California, USA) *(IMC '16)*. ACM, New York, NY, USA, 461–467. https://doi.org/10.1145/2987443.2987483

[21] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. 2011. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In *Proceedings of the ACM SIGCOMM 2011 Conference* (Toronto, Ontario, Canada) *(SIGCOMM '11)*. ACM, New York, NY, USA, 350–361. https://doi.org/10.1145/2018436.2018477

[22] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. 2016. Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *Proceedings of the 2016 ACM SIGCOMM Conference (SIGCOMM '16)*.

[23] Gurobi. 2020. Gurobi is a powerful mathematical optimization solver. https://www.gurobi.com.

[24] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. 2013. Achieving high utilization with software-driven WAN. In *ACM SIGCOMM 2013 Conference, SIGCOMM'13, Hong Kong, China, August 12-16, 2013*.

[25] Chi-Yao Hong, Subhasree Mandal, Mohammad Al-Fares, Min Zhu, Richard Alimi, Kondapa Naidu B., Chandan Bhagat, Sourabh Jain, Jay Kaimal, Shiyu Liang, Kirill Mendelev, Steve Padgett, Faro Rabe, Saikat Ray, Malveeka Tewari, Matt Tierney, Monika Zahn, Jonathan Zolla, Joon Ong, and Amin Vahdat. 2018. B4 and after: Managing Hierarchy, Partitioning, and Asymmetry for Availability and Scale in Google's Software-Defined WAN. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (Budapest, Hungary) *(SIGCOMM '18)*. ACM, New York, NY, USA, 74–87. https://doi.org/10.1145/3230543.3230545

[26] Sushant Jain, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata, Jim Wanderer, Junlan Zhou, Min Zhu, Jon Zolla, Urs Hölzle, Stephen Stuart, and Amin Vahdat. 2013. B4: Experience with a Globally-Deployed Software Defined Wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM* (Hong Kong, China) *(SIGCOMM '13)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2486001.2486019

[27] Virajith Jalaparti, Ivan Bliznets, Srikanth Kandula, Brendan Lucier, and Ishai Menache. 2016. Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. ACM, New York, NY, USA, 73–86. https://doi.org/10.1145/2934872.2934893

[28] Vimalkumar Jeyakumar, Mohammad Alizadeh, David Mazières, Balaji Prabhakar, Albert Greenberg, and Changhoon Kim. 2013. EyeQ: Practical Network Performance Isolation at the Edge. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. USENIX, Lombard, IL, 297–311. https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/jeyakumar

[29] Chuan Jiang, Sanjay Rao, and Mohit Tawarmalani. 2020. PCF: Provably Resilient Flexible Routing. In *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication* (Virtual Event, USA) *(SIGCOMM '20)*. ACM, New York, NY, USA, 139–153. https://doi.org/10.1145/3387514.3405858

[30] Xin Jin, Yiran Li, Da Wei, Siming Li, Jie Gao, Lei Xu, Guangzhi Li, Wei Xu, and Jennifer Rexford. 2016. Optimizing Bulk Transfers with Software-Defined Optical WAN. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. ACM, New York, NY, USA, 87–100. https://doi.org/10.1145/2934872.2934904

[31] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. 2005. Walking the Tightrope: Responsive yet Stable Traffic Engineering. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (Philadelphia, Pennsylvania, USA) *(SIGCOMM '05)*. ACM, New York, NY, USA, 253–264. https://doi.org/10.1145/1080091.1080122

[32] Srikanth Kandula, Ishai Menache, Roy Schwartz, and Spandana Raj Babbula. 2014. Calendaring for Wide Area Networks. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. ACM, New York, NY, USA, 515–526. https://doi.org/10.1145/2619239.2626336

[33] Kermel. 2020. Linux Kernel. http://cdn.kernel.org/pub/linux/kernel/v4.x/.

[34] S. Shunmuga Krishnan and Ramesh K. Sitaraman. 2012. Video Stream Quality Impacts Viewer Behavior: Inferring Causality Using Quasi-Experimental Designs. In *Proceedings of the 2012 Internet Measurement Conference* (Boston, Massachusetts, USA) *(IMC '12)*. ACM, New York, NY, USA, 211–224. https://doi.org/10.1145/2398776.2398799

[35] Alok Kumar, Sushant Jain, Uday Naik, Anand Raghuraman, Nikhil Kasinadhuni, Enrique Cauich Zermeno, C. Stephen Gunn, Jing Ai, Björn Carlin, Mihai Amarandei-Stavila, Mathieu Robin, Aspi Siganporia, Stephen Stuart, and Amin Vahdat. 2015. BwE: Flexible, Hierarchical Bandwidth Allocation for WAN Distributed Computing. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/2785956.2787478

[36] Praveen Kumar, Yang Yuan, Chris Yu, Nate Foster, Robert Kleinberg, Petr Lapukhov, Chiun Lin Lim, and Robert Soulé. 2018. Semi-Oblivious Traffic Engineering: The Road Not Taken. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 157–170. https://www.usenix.org/conference/nsdi18/presentation/kumar

[37] Leslie Lamport. 1998. The part-time parliament. *ACM Transactions on Computer Systems* 16, 2 (1998), 133–169.

[38] Jeongkeun Lee, Yoshio Turner, Myungjin Lee, Lucian Popa, Sujata Banerjee, Joon-Myung Kang, and Puneet Sharma. 2014. Application-Driven Bandwidth Guarantees in Datacenters. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. ACM, New York, NY, USA, 467–478. https://doi.org/10.1145/2619239.2626326

[39] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. 2014. Traffic Engineering with Forward Fault Correction. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. ACM, New York, NY, USA, 527–538. https://doi.org/10.1145/2619239.2626314

[40] L. Luo, H. Yu, Z. Ye, and X. Du. 2018. Online Deadline-Aware Bulk Transfer Over Inter-Datacenter WANs. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 630–638. https://doi.org/10.1109/INFOCOM.2018.8485828

[41] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[42] Debasis Mitra and Qiong Wang. 2005. Stochastic Traffic Engineering for Demand Uncertainty and Risk-Aware Network Revenue Management. *IEEE/ACM Trans. Netw.* 13, 2 (April 2005), 221–233. https://doi.org/10.1109/TNET.2005.845527

[43] Openflow. 2020. sdn and openflow. https://tools.ietf.org/html/rfc7426#page-23.

[44] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, Keith Amidon, and Martin Casado. 2015. The Design and Implementation of Open vSwitch. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. USENIX Association, Oakland, CA, 117–130. https://www.usenix.org/conference/nsdi15/technical-sessions/presentation/pfaff

[45] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*. 1–9.

[46] Martin Suchara, Dahai Xu, Robert Doverspike, David Johnson, and Jennifer Rexford. 2011. Network Architecture for Joint Failure Recovery and Traffic Engineering. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems* (San Jose, California, USA) *(SIGMETRICS '11)*. ACM, New York, NY, USA, 97–108. https://doi.org/10.1145/1993744.1993756

[47] Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage. 2010. California Fault Lines: Understanding the Causes and Impact of Network Failures. In *Proceedings of the ACM SIGCOMM 2010 Conference* (New Delhi, India) *(SIGCOMM '10)*. ACM, New York, NY, USA, 315–326. https://doi.org/10.1145/1851182.1851220

[48] Balajee Vamanan, Jahangir Hasan, and T.N. Vijaykumar. 2012. Deadline-Aware Datacenter Tcp (D2TCP). *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012), 115–126. https://doi.org/10.1145/2377677.2377709

[49] Bruno Vidalenc, Ludovic Noirie, Laurent Ciavaglia, and Eric RENAULT. 2013. Dynamic risk-aware routing for OSPF networks. In *IEEE International Symposium on Integrated Network Management*.

[50] Ye Wang, Hao Wang, Ajay Mahimkar, Richard Alimi, Yin Zhang, Lili Qiu, and Yang Richard Yang. 2010. R3: Resilient Routing Reconfiguration. In *Proceedings of the ACM SIGCOMM 2010 Conference* (New Delhi, India) *(SIGCOMM '10)*. ACM, New York, NY, USA, 291–302. https://doi.org/10.1145/1851182.1851218

[51] Zhiliang Wang, Han Zhang, Xingang Shi, Xia Yin, Yahui Li, Haijun Geng, Qianhong Wu, and Jianwei Liu. 2019. Efficient Scheduling of Weighted Coflows in Data Centers. *IEEE Transactions on Parallel and Distributed Systems* 30, 9 (2019), 2003–2017. https://doi.org/10.1109/TPDS.2019.2905560

[52] Christo Wilson, Hitesh Ballani, Thomas Karagiannis, and Ant Rowtron. 2011. Better Never than Late: Meeting Deadlines in Datacenter Networks. In *Proceedings of the ACM SIGCOMM 2011 Conference* (Toronto, Ontario, Canada) *(SIGCOMM '11)*. ACM, New York, NY, USA, 50–61. https://doi.org/10.1145/2018436.2018443

[53] Hong Zhang, Kai Chen, Wei Bai, Dongsu Han, Chen Tian, Hao Wang, Haibing Guan, and Ming Zhang. 2015. Guaranteeing Deadlines for Inter-Datacenter Transfers. In *Proceedings of the Tenth European Conference on Computer Systems* (Bordeaux, France) *(EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 20, 14 pages. https://doi.org/10.1145/2741948.2741957

[54] H. Zhang, X. Shi, X. Yin, F. Ren, and Z. Wang. 2015. More load, more differentiation– A design principle for deadline-aware congestion control. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 127–135.