Achieving High Availability in Inter-DC WAN Traffic Engineering

Han Zhang, *Member, IEEE*, Xia Yin, Xingang Shi, *Member, IEEE*, Jilong Wang, Zhiliang Wang, Yingya Guo, Tian Lan, *Senior Member, IEEE*, Yahui Li, Yongqing Zhu, Ke Ruan, and Haijun Geng

***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 *Diverse Availability Targets* (DAT), 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.**

Manuscript received 17 December 2021; revised 10 August 2022; accepted 23 September 2022; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor K. C.-J. Lin. Date of publication 12 December 2022; date of current version 19 December 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62002009 and in part by the National Key Research and Development Program of China under Grant 2018YFB1800401. *(Corresponding author: Xingang Shi.)*

Han Zhang, Xingang Shi, Jilong Wang, Zhiliang Wang, and Yahui Li are with the INSC&BNRist, Tsinghua University, Beijing 100084, China, and also with the Zhongguancun Laboratory, Beijing 100000, China (e-mail: shixg@cernet.edu.cn).

Xia Yin is with the DCST, Tsinghua University, Beijing 100084, China, and also with the Zhongguancun Laboratory, Beijing 100000, China.

Yingya Guo is with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350025, China.

Tian Lan is with the Department of Electrical and Computer Engineering, George Washington University, Washington, DC 20052 USA.

Yongqing Zhu and Ke Ruan are with the Research Institute of China Telecom, Beijing 100033, China.

Haijun Geng is with the School of Automation and Software Engineering, Shanxi University, Taiyuan 030006, China.

Digital Object Identifier 10.1109/TNET.2022.3216592

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], [29], [32], [33], [34], [35], [38], [40], [43], [44], [47].

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 [28], [29], [57]. For example, Microsoft reports links in their WAN could fail as often as every 30 minutes [47]. 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 [32], [40], [62]. 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], [32], [34], [37], [44], [47], [60] 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], [37], [44], [47], [60] 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

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%

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], [29], [32], [34] or leasing bandwidth from ISPs [7], [21]), this is quite uneconomic [31]. *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 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 not considered by their failure recovery approaches, especially those who allocate bandwidth aggressively [20], [32]. 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 $\beta_d\%$ 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 [33], [34], [43], 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. In reality, some time-critical applications (e.g., real-time data analysis [50], bulk transfer [48], high-definition video streaming [23], search indexes synchronization [34], industry control system [55]) might need low latency, therefore, we need to efficiently solve the 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.

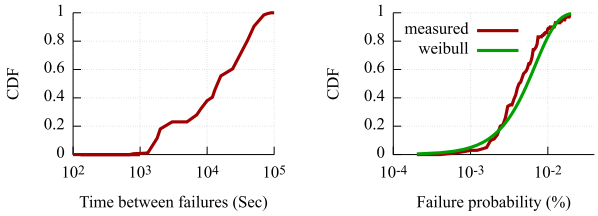
Thirdly, we implement TEDAT as a real system, including a centralized controller and multiple brokers (one for each DC) (see §IV). We conduct extensive experiments using a network testbed as well as trace driven large scale simulations (see §V). We compare TEDAT with state-of-the-art WAN TE schemes such as TEAVAR [20], SMORE [44], SWAN [32], B4 [34] and FFC [47], 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 $30\times$ 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 Cloud Services

1) WAN Failures are Frequent and Follow a Heavy-Tailed Distribution: Failures could occur anywhere, from control



(a) Failure intervals (b) Empirical link failure

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

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], [29], leading to severe performance degradation and revenue loss. Our failure intervals measurement of a commercial inter-DC WAN shown in Fig. 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 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 [27], [28], [57]. Therefore, *network failures, especially their uneven distribution, should be explicitly taken into account by network operators*.

2) 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 $\beta\%$ of the time [33]. 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 [42], [49], [54]. Therefore, for a BA demand d, formulating the *hard* guarantee such as "bandwidth requirement should be guaranteed at least $\beta_d\%$ of its life duration" may be beneficial.

3) 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*.

TABLE II
BANDWIDTH AVAILABILITY TARGETS IN B4 [33]

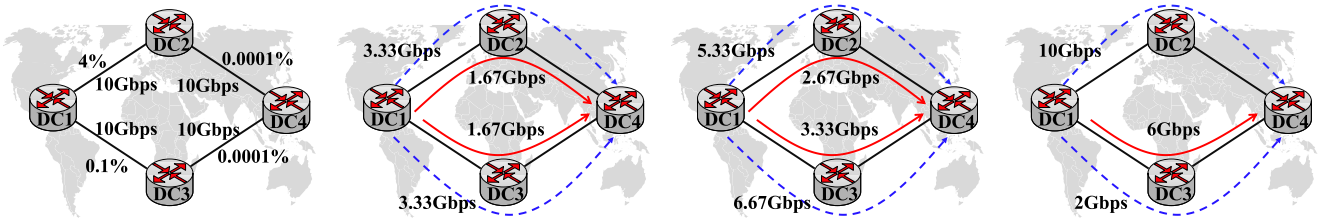
Service	Availability
Search ads, DNS, WWW	99.99%
Photo service, backend, Email	99.95%
Ads database replication	99.9%
Search index copies, logs	99%
Bulk transfer	N/A

4) 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 [42], [49], [54]. 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 [33], [34] 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).

5) Low Admission Processing Latency is Necessary for Some Time-Critical Applications: More service provides are willing to move their applications to distributed data centers to serve their worldwide users [48]. Among the applications, some time-critical ones (e.g., real-time data analysis [50], bulk transfer [48], high-definition video streaming [23], search indexes synchronization [34], industry control system [55]) need low admission processing latency since they might have deadline requirement, and *any deadline missing will lead to data worthless*. The deadline is always defined in their SLA and varies from minutes to hours [32], [40], [48]. In the future, more delay-sensitive services (e.g., autonomous driving, remote surgery) might be moved to distributed clouds. To guarantee the revenue, existing traffic engineering controller should make fast decision to prevent deadline missing.

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,



(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.

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 [47] and TEAVAR [20], two latest WAN traffic engineering schemes that take network failures into account, to this scenario.

FFC [47] 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.

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, especially for the time-critical applications.
- **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.

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
t	a tunnel for transmitting traffic [1]
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

A. Abstraction of Bandwidth Availability

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

1) 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 $\mathbf{z} =< \mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{|E|} >$ 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. 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.,

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

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 $p_{\mathbf{z}} = p_{<1,1,0,1>} = 0.96 \times 0.999999 \times 0.001 \times 0.999999 \simeq 0.000959998$.

2) 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 $< \mathbf{b}_d^1, \ldots, \mathbf{b}_d^k, \ldots >$ of bandwidth demands on each s-d pair $k \in K$[2] and β_d is its bandwidth availability target.

3) BA Provision Model: Similar to [20], [32], [47], 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 [59], oblivious routing [44], 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 the 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, and it can present the *achieved availability* under an allocation result $\{f_d^t\}$. Given the *availability target* β_d, we now formally define a demand d is satisfied by an allocation $\{f_d^t\}$, if and only if

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

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

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

We assume user demands are served in a first-come-first-serve (FCFS) manner without preemption. Let D and $\hat{D}$ denote the set of arrived and admitted demands.[3] When a new demand d arrives, we have $D = \hat{D} \cup d$. TEDAT tries to *accommodate as many demands as possible*: if every demand in D can meet its availability target, then d should be admitted and our traffic scheduling procedure will allocate bandwidth to it, otherwise, it should be rejected. We now try to formulate the traffic scheduling in Appendix A. 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 $d_1, d_2, \ldots$, 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:

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

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. ■

[3]When an admitted demand finishes, it will be removed from $\hat{D}$.

Algorithm 1 Heuristic for Solving Traffic Scheduling

Input: Input parameters shown in TABLE III.
Output: Bandwidth allocation results.

```
1  b̂_d^k = b_d^k, ∀d ∈ D, k ∈ K;
3  f̂_d^t = f_d^t, ∀d ∈ D, k ∈ K, t ∈ T_k;
5  while true do
6      d = arg_{d'∈D} min{Σ_{k∈K} b̂_{d'}^k × β_{d'}};
8      for k ∈ K do
9          if b_d^k > remaining capacity of s-d pair k then
10             return False, {f_d^t};
11         T'_k = T_k;
13         while b̂_d^k > 0 do
14             t = arg_{t∈T'_k} min{c_t * p_t};
16             f̂_d^t = min{c_t, b̂_d^k};
18             T'_k = T'_k \ t;
20             s_d = s_d * p_t;
22             b̂_d^k = b̂_d^k − f̂_d^t;
24             update the remaining capacities of links
25             and tunnels;
26     if s_d < β_d then
27         return False, {f_d^t};
28     D = D \ d;
29 return True, {f̂_d^t};
```

However, in order to support agile deployment of time-critical applications, 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 (i.e., $\{f_d^t\}$) 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, then admit d and make bandwidth allocation for it (i.e., $\{f_d^t\}$).
3) If d still cannot be accommodated, reject the demand without rescheduling the whole network.

The greedy method shown in Algorithm 1, 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 gives priority to the demand which has the smallest product of bandwidth target and availability target (i.e., $\sum_{k\in K} \hat{b}_d^k \times \beta_d$) to make room for more demands (line 4), then it 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 (line 9-15). In this process, it first finds the tunnel which has the smallest product of remaining capacity and availability probability (i.e., $c_t * p_t$) (line 10), and then determines the bandwidth through the tunnel (line 11). Then it will update some intermediate variables (Line 12-15). Finally, 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. ■

Analyzing the performance loss of general TEDAT is not the focus of this paper. Therefore, we take the simplest dumbbell 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.

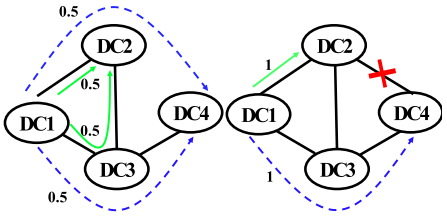
Proof: Algorithm 1 will prior BA demands according to the following sequence (Line 4):

$$\sum_{k\in K} \hat{b}_{d1}^k \times \beta_{d1} \le \sum_{k\in K} \hat{b}_{d2}^k \times \beta_{d2} \le \dots. \tag{2}$$

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 $\sum_{i=1}^{n} a_i \le OPT$. Also, we have $\sum_{i=1}^{n+1} a_i \ge OPT$. 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. $\sum_{i=1}^{n+1} a_i \le 2\sum_{i=1}^{n} a_i$. Therefore, $OPT \le 2\sum_{i=1}^{n} a_i$. 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



(a) Original allocation (b) Backup allocation

Fig. 3. A failure recovery example.

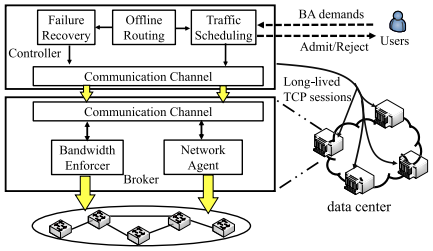


Fig. 4. TEDAT system design.

packet loss can be mitigated.[4] 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.

Let Z^* denote the scenario set (e.g., one link failure scenarios) that we want to configure fast failure recovery. For a specific network scenario $\mathbf{z} \in Z^*$ in consideration, the ratio of allocated bandwidth to a user's demanded bandwidth is:[5]

$$R_{dk} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in \hat{D}, k \in K \tag{3}$$

For the specific network scenario $\mathbf{z} \in Z^*$, 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

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

For the specific network scenario $\mathbf{z} \in Z^*$, 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:

$$\begin{aligned} & 0 \leq y_d \leq 1, \quad \forall d \in \hat{D} \\ & r_d = g_d \times \Big(y_d + (1-\mu_d) \times (1-y_d) \Big), \quad \forall d \in \hat{D} \\ & R_{dk} < M \times y_d + 1 - y_d, \quad \forall d \in \hat{D}, k \in K \\ & R_{dk} \geq y_d, \quad \forall d \in \hat{D}, k \in K \end{aligned} \tag{4}$$

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 piecewize function linearized.

For the specific network scenario $\mathbf{z} \in Z^*$, the bandwidth allocation result f_d^t 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

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

and

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

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

$$\begin{aligned} & maximize \sum_{d \in \hat{D}} r_d \\ & s.t. \quad (3), (4), (5), (6) \end{aligned} \tag{7}$$

The above failure recovery problem is a LP problem and it is easy to solve it using the optimization problem solver such as Gurobi [30]. In reality, the total scenario set Z^* can be decided by the network operators on demand and we just consider one link failure in our evaluation part.

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,

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

[5]This is the same as equation (8), but we omit the superscript $\mathbf{z}$ of for this equation. Equations (3)-(7) also omit the superscript $\mathbf{z}$ in our formulation.

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.

A. 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 [44], k-shortest path [32], 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 [32], FFC [47] 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 [45] algorithm.

B. 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 [25]), and uses the OpenFlow [52] 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.

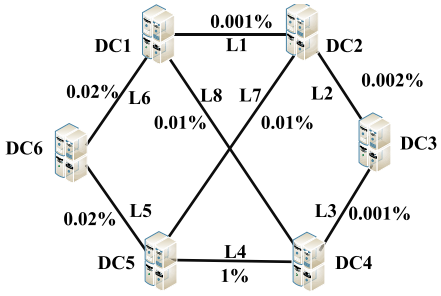


Fig. 5. Testbed topology.

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 [47] and TEAVAR [20]. For simulation, we implement more TE algorithms, including SWAN [32], SMORE [44] and B4 [34]. 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,[6] 10%∼20% more profit can be retained when failures occur.

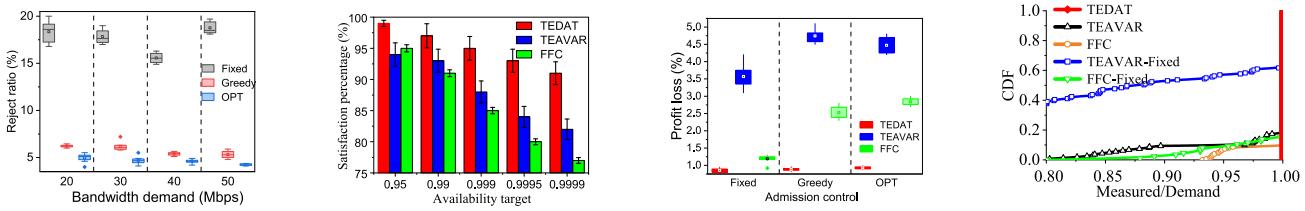
(2) TEDAT achieves a good tradeoff between efficiency and optimality. Compared with the optimal solutions, (i) our admission control algorithm can speed up the admission procedure by 30× at the expense of less than 4% false rejections, (ii) our pruning-augmented scheduling algorithm runs $10^2 \sim 10^4\times$ 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

1) Testbed Setup: We build a testbed with 6 servers to emulate a small inter-DC WAN connecting 6 DCs, as shown

[6]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.

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 [53]. The VMs run CentOS 7 and use Linux v4.15.6 kernel [41]. 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 [25] 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.

2) 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], [38], [48], [62]. 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.

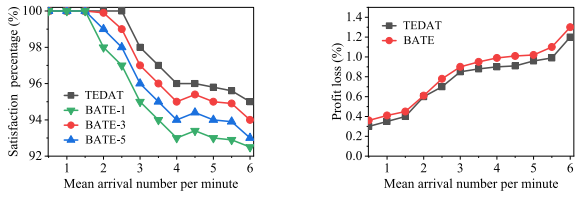
a) Traffic Scheduling: We evaluate how demands can be correctly admitted by TEDAT. The two baseline algorithms are the optimal admission strategy by solving an optimization problem shown in Section III-B and the step (1) of TEDAT admission control strategy which assumes a *fixed* bandwidth allocation 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%).

b) 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 [64] 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 [64]. 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 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



(a) Availability (b) Profit loss

Fig. 7. TEDAT and BATE comparison.

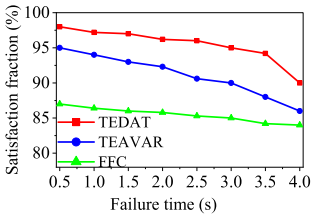


Fig. 8. Different failure time.

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

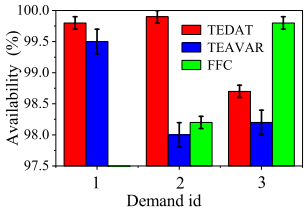


Fig. 9. Bandwidth availability.

BA targets satisfaction when varying failure time from 0.5s to 4.0 seconds.

3) 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

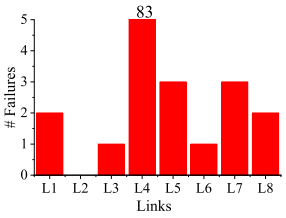


Fig. 10. Total link failures.

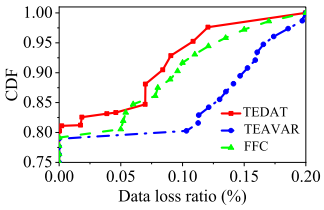


Fig. 11. Data loss ratio comparison.

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%).[7] 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. Compared with TEAVAR, TEDAT does not need

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

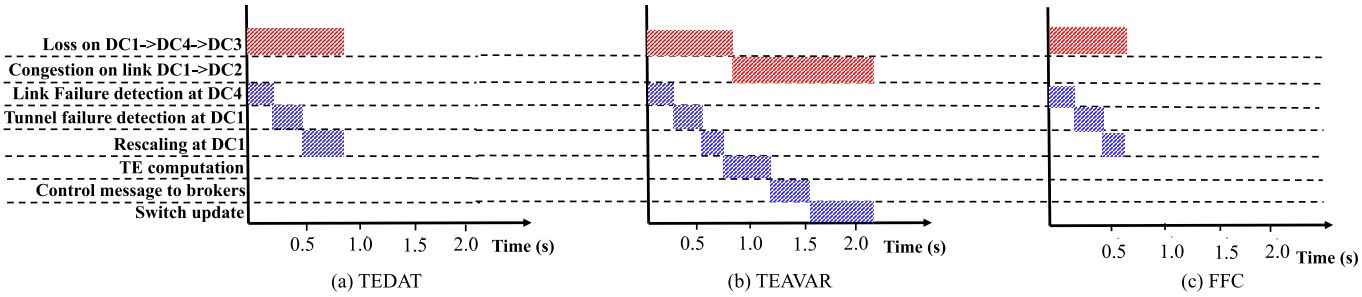


Fig. 12. Events comparison when the link DC4 → DC3 fails, where the x-axis is a time line relative to when the link failure was injected.

TABLE V
NETWORK TOPOLOGIES USED IN THE SIMULATIONS

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

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

1) Simulation Setup: We conduct simulations on four real network topology, including B4 [34], ATT [20], IBM [44] and FITI (a national-level backbone). TABLE V shows the topology.[8] 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$s$,[9] 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 [32], SMORE [44] and B4 [34]. 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.

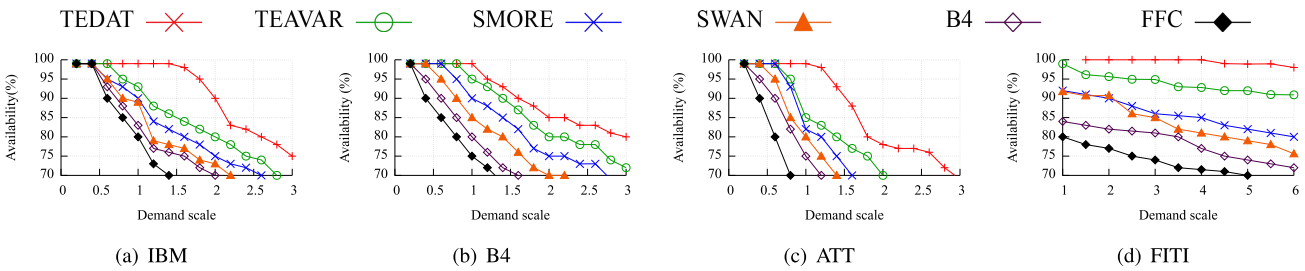
2) Evaluation Results: Nowadays, some ISP networks are designed with worst-case assumptions about failures, so topologies might be over-provisioned. We firstly assume BA demands arrive simultaneously and then scale up the average matrices by a factor s [20], [44], [47], where the y-axis denotes the total 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 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,

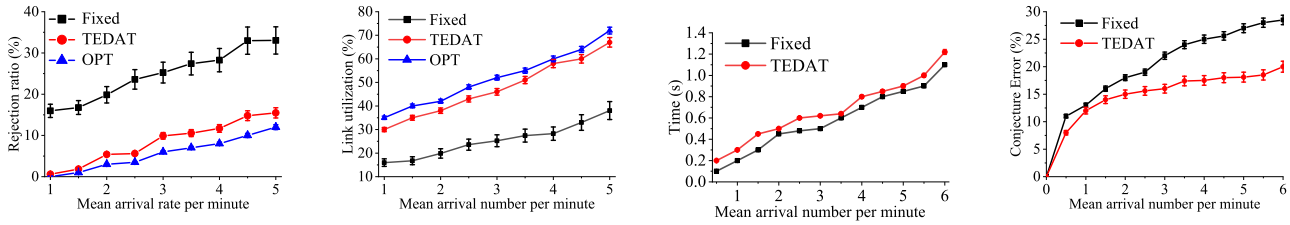
[8]For B4, ATT and IBM, we get their topology, link capacities and traffic matrices from the authors of TEAVAR [20], and for FITI, we conduct a direct measurement on it but uses just part of it.

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



(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.

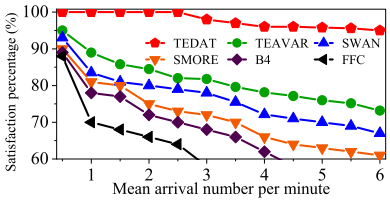


Fig. 15. TEDAT v.s. other TEs.

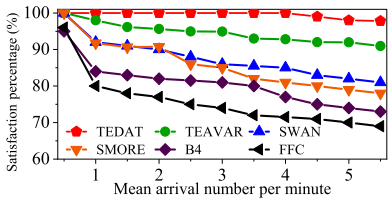


Fig. 16. *fixed* admission control.

TEDAT 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

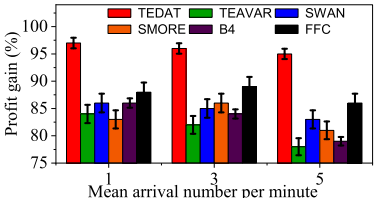


Fig. 17. Profit gain after failures.

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.

3) 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 [44] and edge disjoint path routing [59], 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

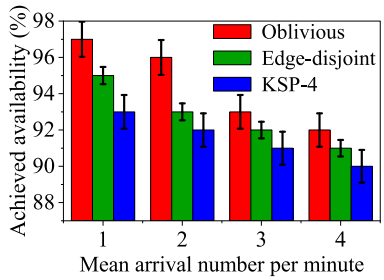


Fig. 18. Different routing schemes.

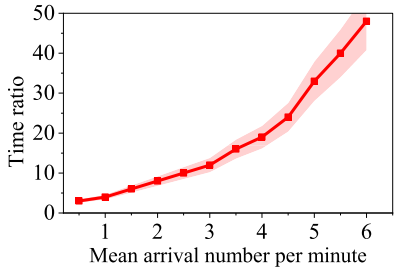


Fig. 19. Acceleration.

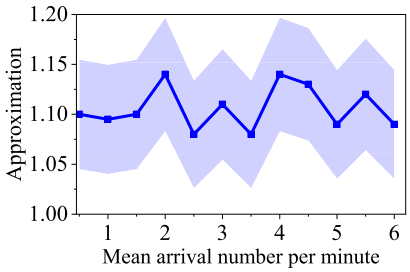


Fig. 20. Approximate ratio.

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 [26] and MPLS [24], [39], for given traffic matrices. Recently, Software defined network (SDN) based technologies, including SWAN [32], B4 [33], [34], Bwe [43] and OWAN [38], rely on a centralized view to optimize bandwidth allocations. Pretium [35] combines dynamic pricing with traffic engineering for inter-DC bandwidth, but it does not provide guarantee on network bandwidth. Network scheduling schemes [40], [61] 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 [58], [62], [63], flow rate [36], [46], traffic engineering [32], [33], [34], [35], [40], 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 [29] and optical networks [27], stochastic models [19], [51] and failure recovery methods [56], [60]. 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 [47] 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.

Appendix A
TEDAT Problem Formulation

A. Input

Let c_e denote the capacity of link $e \in E$ over inter-DC WAN denoted as $G(V, E)$. We use Z to present the scenario set. $p_{\mathbf{z}}$ is the probability of scenario $\mathbf{z} \in Z$. Let T_k denote the tunnel set for node pair $k \in K$. In addition to D and $\hat{D}$, $v_t^{\mathbf{z}}$ and u_t^e are also known by the network operators beforehand.

B. Output

The output of TEDAT contains two parts: (1) f_d^t is the bandwidth allocated for demand $d \in D$ over tunnel t; (2) a_d represents whether the BA target of $d \in D$ can be satisfied.

C. Optimization Problem

We now show the constraints. 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:

$$R_{dk}^{\mathbf{z}} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in D, \mathbf{z} \in Z, k \in K. \tag{8}$$

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

$$\begin{aligned} q_d^{\mathbf{z}} &\in \{0,1\}, \forall d \in \hat{D}, \mathbf{z} \in Z \\ R_{dk}^{\mathbf{z}} &< M \times q_d^{\mathbf{z}} + 1 - q_d^{\mathbf{z}}, \forall d \in \hat{D}, k \in K \\ R_{dk}^{\mathbf{z}} &\geq q_d^{\mathbf{z}}, \forall d \in \hat{D}, k \in K \end{aligned} \tag{9}$$

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.,

$$s_d = \sum_{\mathbf{z} \in \mathbf{z}} q_d^{\mathbf{z}} \times p_{\mathbf{z}}, \quad \forall d \in D. \tag{10}$$

If s_d is not smaller than the BA target β_d, then the availability target can be satisfied. We 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

$$\begin{aligned} a_d &\in \{0,1\}, \forall d \in D \\ s_d &< \beta_d \times (1 - a_d) + a_d, \forall d \in D \\ s_d &\geq \beta_d \times a_d, \forall d \in D \end{aligned} \tag{11}$$

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.,

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

and

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

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

$$\begin{aligned} &maximize \sum_{d \in D} a_d \\ &s.t. \qquad (8), (9), (10), (11), (12), (13) \end{aligned} \tag{14}$$

REFERENCES

[1] Aliababa. (2020). *Data Transmission Service Level Agreement*. [Online]. Available: https://www.alibabacloud.com/help/zh/doc-detail/50079.htm

[2] Aliababa. (2020). *Short Message Service (SMS) Service Level Agreement*. [Online]. Available: https://www.alibabacloud.com/help/zh/doc-detail/155130.htm

[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 *Proc. 27th ACM Symp. Operating Syst. Principles*, New York, NY, USA, 2019, pp. 414–429.

[4] Amazon. (2020). *Amazon Appflow Service Level Agreement*. [Online]. Available: https://aws.amazon.com/cn/appflow/sla/

[5] Amazon. (2020). *Aws Database Migration Service (AWS DMS) Service Level Agreement*. [Online]. Available: https://aws.amazon.com/cn/dms/sla/

[6] Amazon. (2021). *Amazon Compute Service Level Agreement*. [Online]. Available: https://aws.amazon.com/compute/sla/?nc1=h_ls

[7] Aryaka. (2020). *Aryaka Private Wan*. [Online]. Available: https://www.aryaka.com

[8] Azure. (2021). *Azure Active Directory Domain Services*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/active-directory-ds/v1_0/

[9] Azure. (2021). *SLA for API Management*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/api-management/v1_5/

[10] Azure. (2021). *SLA for APP Configuration*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/app-configuration/v1_0/

[11] Azure. (2021). *SLA for Application Gateway*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/application-gateway/v1_2/

[12] Azure. (2021). *SLA for Application Insights*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/application-insights/v1_2/

[13] Azure. (2021). *SLA for Automation*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/automation/v1_1/

[14] Azure. (2021). *SLA for Azure Cache for Redis*. [Online]. Available: https://azure.microsoft.com/en-us/ support/legal/sla/cache/v11/

[15] Azure. (2021). *SLA for Baremetal Infrastructure*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/baremetal-infrastructure/v1_0/

[16] Azure. (2021). *SLA for Content Delivery Network*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/cdn/v1_0/

[17] Azure. (2021). *SLA for Storage Accounts*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_5/

[18] Azure. (2021). *SLA for Virtual Machines*. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

[19] Y. Bi and A. Tang, "Uncertainty-aware optimization for network provisioning and routing," in *Proc. 53rd Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2019, pp. 1–6.

[20] J. Bogle et al., "Teavar: Striking the right utilization-availability balance in WAN traffic engineering," in *Proc. ACM Special Interest Group Data Commun.*, New York, NY, USA, 2019, pp. 29–43.

[21] Cato. (2020). *Cato Managed Services*. [Online]. Available: https://www.catonetworks.com

[22] C. Chekuri, S. Khanna, and F. B. Shepherd, "The all-or-nothing multicommodity flow problem," in *Proc. 36th Annu. ACM Symp. Theory Comput. (STOC)*, 2004, pp. 156–165.

[23] Dacast. (2022). *High-Definition Video Streaming*. [Online]. Available: https://www.dacast.com/blog/hd-live-streaming/

[24] A. Elwalid, C. Jin, S. Low, and I. Widjaja, "MATE: MPLS adaptive traffic engineering," in *Proc. IEEE INFOCOM. Conf. Comput. Commun. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Aug. 2001, pp. 1300–1309.

[25] Floodlight. (2020). *Floodlight Controller*. [Online]. Available: https://github.com/floodlight/floodlight

[26] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS weights in a changing world," *IEEE J. Sel. Areas Commun.*, vol. 20, no. 4, pp. 756–767, May 2002.

[27] M. Ghobadi and R. Mahajan, "Optical layer failures in a large backbone," in *Proc. Internet Meas. Conf.*, New York, NY, USA, Nov. 2016, pp. 461–467.

[28] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: Measurement, analysis, and implications," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)*, New York, NY, USA, 2011, pp. 350–361.
[29] R. Govindan, I. Minei, M. Kallahalla, B. Koley, and A. Vahdat, "Evolve or die: High-availability design principles drawn from Googles network infrastructure," in *Proc. ACM SIGCOMM Conf.*, Aug. 2016, pp. 1–95.
[30] Gurobi. (2020). *Gurobi is a Powerful Mathematical Optimization Solver*. [Online]. Available: https://www.gurobi.com
[31] Y. Harchol et al., "A public option for the core," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, New York, NY, USA: Association for Computing Machinery, 2020, pp. 377–389.
[32] C. Hong et al., "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM*, Hong Kong, Aug. 2013, pp. 1–12.
[33] C.-Y. Hong et al., "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. Conf. ACM Special Interest Group Data Commun.*, New York, NY, USA, 2018, pp. 74–87.
[34] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," in *Proc. ACM SIGCOMM*, New York, NY, USA, 2013, pp. 3–14.
[35] V. Jalaparti, I. Bliznets, S. Kandula, B. Lucier, and I. Menache, "Dynamic pricing and traffic engineering for timely inter-datacenter transfers," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, Aug. 2016, pp. 73–86.
[36] V. Jeyakumar, M. Alizadeh, D. Mazières, B. Prabhakar, A. Greenberg, and C. Kim, "EyeQ: Practical network performance isolation at the edge," in *Proc. 10th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Lombard, IL, USA, 2013, pp. 297–311.
[37] C. Jiang, S. Rao, and M. Tawarmalani, "PCF: Provably resilient flexible routing," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, New York, NY, USA, Jul. 2020, pp. 139–153.
[38] X. Jin et al., "Optimizing bulk transfers with software-defined optical WAN," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, Aug. 2016, pp. 87–100.
[39] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, New York, NY, USA, 2005, pp. 253–264.
[40] S. Kandula, I. Menache, R. Schwartz, and S. R. Babbula, "Calendaring for wide area networks," in *Proc. ACM Conf. SIGCOMM*, New York, NY, USA, Aug. 2014, pp. 515–526.
[41] Kermel. (2020). *Linux Kernel*. [Online]. Available: http://cdn.kernel.org/pub/linux/kernel/v4.x/
[42] S. S. Krishnan and R. K. Sitaraman, "Video stream quality impacts viewer behavior: Inferring causality using quasi-experimental designs," in *Proc. Internet Meas. Conf.*, New York, NY, USA, 2012, pp. 211–224.
[43] A. Kumar et al., "BWE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proc. ACM Conf. Special Interest Group Data Commun.*, New York, NY, USA, 2015, pp. 1–14.
[44] P. Kumar et al., "Semi-oblivious traffic engineering: The road not taken," in *Proc. USENIX NSDI*, Renton, WA, USA, Apr. 2018, pp. 157–170.
[45] L. Lamport, "The part-time parliament," *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.
[46] J. Lee et al., "Application-driven bandwidth guarantees in datacenters," in *Proc. ACM Conf. SIGCOMM*, New York, NY, USA, 2014, pp. 467–478.
[47] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM Conf. SIGCOMM*, New York, NY, USA, Aug. 2014, pp. 527–538.
[48] L. Luo, H. Yu, Z. Ye, and X. Du, "Online deadline-aware bulk transfer over inter-datacenter WANs," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2018, pp. 630–638.
[49] H. Mao, R. Netravali, and M. Alizadeh, "Neural adaptive video streaming with pensieve," in *Proc. Conf. ACM Special Interest Group Data Commun.* New York, NY, USA: Association for Computing Machinery, Aug. 2017, pp. 197–210.
[50] Microsoft. (2022). *Real-Time Analytics on Big Data Architecture*. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/solution-ideas/articles/real-time-analytics
[51] D. Mitra and Q. Wang, "Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management," *IEEE/ACM Trans. Netw.*, vol. 13, no. 2, pp. 221–233, Apr. 2005.
[52] Openflow. (2020). *SDN and OpenFlow*. [Online]. Available: https://tools.ietf.org/html/rfc7426#page-23
[53] B. Pfaff et al., "The design and implementation of open vSwitch," in *Proc. 12th USENIX Symp. Networked Syst. Design Implement. (NSDI)*, Oakland, CA, May 2015, pp. 117–130.
[54] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. IEEE 35th Annu. IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.
[55] K. Stouffer, J. Falco, and K. Scarfone, "Guide to industrial control systems (ICS) security," *NIST Special Publication*, vol. 800, no. 82, p. 16, 2011.
[56] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Model. Comput. Syst. (SIGMETRICS)*, New York, NY, USA, 2011, pp. 97–108.
[57] D. Turner, K. Levchenko, A. C. Snoeren, and S. Savage, "California fault lines: Understanding the causes and impact of network failures," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2010, pp. 315–326.
[58] B. Vamanan, J. Hasan, and T. N. Vijaykumar, "Deadline-aware datacenter TCP (D2TCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 115–126, 2012.
[59] B. Vidalenc, L. Noirie, L. Ciavaglia, and E. Renault, "Dynamic risk-aware routing for OSPF networks," in *Proc. IFIP/IEEE Int. Symp. Integr. Network Manag. (IM)*, 2013, pp. 226–234.
[60] Y. Wang et al., "R3: Resilient routing reconfiguration," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2010, pp. 291–302.
[61] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowtron, "Better never than late: Meeting deadlines in datacenter networks," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2011, pp. 50–61.
[62] H. Zhang et al., "Guaranteeing deadlines for inter-datacenter transfers," in *Proc. 10th Eur. Conf. Comput. Syst.* New York, NY, USA: Association for Computing Machinery, 2015, pp. 1–14.
[63] H. Zhang, X. Shi, X. Yin, F. Ren, and Z. Wang, "More load, more differentiation—A design principle for deadline-aware congestion control," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2015, pp. 127–135.
[64] H. Zhang et al., "Boosting bandwidth availability over inter-DC WAN," in *Proc. 17th Int. Conf. Emerg. Netw. EXperiments Technol.*, 2021, pp. 297–312.

Han Zhang (Member, IEEE) received the B.S. degree in computer science and technology from Jilin University and the Ph.D. degree from Tsinghua University. He is currently working with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include computer networks and network security. He has published more than 40 articles in his area.

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

Xingang Shi (Member, IEEE) received the B.S. degree from Tsinghua University and the Ph.D. degree from The Chinese University of Hong Kong. He is currently working with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include network measurement and routing protocols.

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 interests include network measurement, location-oriented networks, and SDN systems, and network security.

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. He is currently an Associate Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interests include formal methods and protocol testing, next generation internet, and network measurement.

Yingya Guo received the B.S. degree in computer science from Xiamen University, China, in 2013, and the Ph.D. degree from Tsinghua University. She is currently an Assistant Professor with the College of Mathematics and Computer Science, Fuzhou University, Fuzhou, China. Her research interests include traffic engineering, routing optimization, and software-defined networking.

Tian Lan (Senior Member, IEEE) received the B.A.Sc. degree in electrical engineering from Tsinghua University in 2003, the M.S. degree from the Department of Electrical and Computer Engineering, University of Toronto, in 2005, and the Ph.D. degree from the Department of Electrical Engineering, Princeton University, in 2010. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, George Washington University, where he joined in 2010. His research interests include network optimization and algorithms, cyber security, network protocols, cloud and edge (fog) computing, distributed storage, and wireless networks.

Yahui Li, photograph and biography not available at the time of publication.

Yongqing Zhu received the B.E. degree from the Shandong University of Science and Technology in 1997, and the M.E. degree from Beihang University in 2000. He is currently working with the Institute of Network Technology, Research Institute of China Telecom. His research interests include IP networks and cloud computing.

Ke Ruan received the B.E. and M.E. degrees from Wuhan University in 1997 and 2004, respectively. He is currently working with the Institute of Network Technology, Research Institute of China Telecom. His research interests include IP networks and cloud computing.

Haijun Geng received the B.E. degree from Yantai University in 2008, the M.E. degree from Capital Normal University in 2011, and the Ph.D. degree from Tsinghua University in 2015. He is currently working with the School of Software Engineering, Shanxi University. His research interests include future internet architecture and large scale internet routing.