TEDAT: Guaranteeing Bandwidth-based Availability for Services Over Inter-DC WAN

Paper # 56, 16 pages

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 the limited bandwidth resources 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 that aims for *bandwidth-based availability* provision, where a service level agreement (SLA) defines that a demand on certain bandwidth should 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 TEDAT, i.e., admission control, traffic scheduling and failure recovery, are built on different mathematical models and theoretically analyzed. They are also extensively compared against state-of-the-art TE schemes, using trace driven simulations as well as real testbed across different topologies, traffic matrices and failure scenarios. Our evaluation demonstrates that compared with the optimal admission strategy, TEDAT can speed up the online admission control by 30 × at the expense of 3% higher rejection ratio. On the other hand, compared with the latest TE schemes like FFC and TEAVAR, TEDAT can meet the availability SLAs of 40% more bandwidth demands, and when network failure causes SLA violations, it can retain 30% more profit under a simple pricing and refunding model.

1 INTRODUCTION

Nowadays, large scale online services such as finance trading, web search, online game and video streaming are posing stringent requirements on the availability and agility of the underlying network infrastructure, 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 and etc., are providing various optimizations for their global WAN, especially with the help of the emerging software-defined networking techniques [10, 18, 20–23, 26, 28, 31, 32, 36].

Among various optimization targets, high network availability has always been, and continues 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 [17, 18, 44]. For example, Microsoft reports links in their WAN could fail as often as every 30 minutes [36]. Once a link fails, traffic has to be rescaled and rerouted, resulting in transit or long lasting congestions. Such negative impact on inter-DC WAN services will ultimately translate into monetary loss (e.g., refund to customers). At the same time, as more business moves to cloud, and there are inevitable competitions over the scarce inter-DC WAN resources [20, 28, 49]. Therefore, the design and optimization of inter-DC WANs have to take heterogeneities and competitions into consideration.

In this paper, we argue that although existing traffic engineering schemes [10, 15, 25, 36, 39, 43, 47] have already factored in network risks and targets for network availability guarantee, they still have three limitations when facing heterogeneous and competing bandwidth demands over inter-DC WAN: *Firstly*, some proactive schemes (e.g., FFC [36]) need too much bandwidth headroom to guarantee network is free from congestion when there are failures, and consequently their bandwidth allocation can often be too conservative [25, 36, 47] and result in inefficient resource usage as well as significant performance gap [10, 16]. To prevent congestion during rerouting, links, including those with negligible failure probabilities must be kept at low utilization, and this method is suited for ISP networks which are designed with worst-case assumptions about failures. However, content providers can't invest so much in their private backbone (e.g., inter-DC WAN) as public backbone to reduce the risk of congestion [19], therefore, leaving too much capacity unused most of the time when no link fails over inter-DC WAN implies resource wastage. *Secondly*, existing techniques (e.g., TEAVAR [10]) usually emphasize on network service availability rather than the availability targets of applications deployed over networks, i.e., they normally take a *one-size-fit-all* approach and do not differentiate availability requirements of different classes of traffic, which may impair high availability traffic when there are competitions. Availability is one of the main items of Service Level Agreements (SLAs) [5, 8] and cloud providers have to refund customers for any violation of SLAs. For example, Microsoft Azure should guarantee at least 99.9% availability of the backup service and a minimum of 99.95% availability of ExpressRoute service [8]. If the availability agreement is violated, a 10% or 25% refund will return to customers. TEAVAR [10] can't guarantee the availability of high priority traffic when resource is inadequate. *Thirdly*, their failure recovery schemes might cause much monetary loss when network fails. In tunnel-based traffic engineering schemes (e.g., TEAVAR [10], FFC [36]), traffic can be redistributed across the surviving tunnels via rerouting when any tunnel becomes unavailable. Their rerouting mechanisms via surviving tunnels could lead to much revenue loss when services attempt to restore rate after failures.

Facing these challenges, in this paper we make the following **contributions**:

Firstly, we advocate traffic engineering with *bandwidth-based availability guarantee* [21]: a tenant's *promised bandwidth of service j should be available at least β_j% time.* It tries to guarantee the bandwidth demands from tenants under network uncertainty. We show that state-of-the-art traffic engineering schemes fail to meet services' bandwidth-based availability targets under arbitrarily link failures (see §2). This is due to the fact that these schemes either ignore the difference in link failure probabilities that can vary by several orders of magnitude in practice or fail to consider the diverse availability targets of services.

Secondly, we propose TEDAT (Traffic Engineering under Diverse Availability Targets), a novel TE optimization framework that allows network operators to optimize bandwidth-based availability for diverse services over inter-DC WAN (see §4). Compared with other TE schemes (see §6), TEDAT is the first work to formally guarantee bandwidth-based availability demands from tenants in a unifying framework. TEDAT employs a fixed-readmission-combination allocation algorithm for online admission control to strike balance between efficiency and optimality. We prove that for demands already admitted by our algorithm, there must exist an allocation scheme to satisfy their bandwidth-based availability targets. We further propose a linear programming scheduling algorithm to allocate bandwidth for the admitted demands periodicly. The linear programming complexity increases exponentially with network size and we propose a pruning method which cuts off the scenarios that hardly happen to effectively reduce the size. Our online scheduling algorithm is able to guarantee diverse availability targets of services even when network resource is inadequate. TEDAT also precomputes backup allocation across each tunnel to maximize revenue when a link fails. We model the failure recovery problem as a 0-1 Mixed-integer linear programming, prove its NP-hardness and propose a 2-approximation algorithm to solve it.

Thirdly, we design and implement TEDAT system (see §3) and conduct extensive experiments through real trace driven simulations and on a small testbed (see §5). We compare TEDAT with state-of-the-art TE schemes such as TEAVAR [10], SMORE [32], SWAN [20] and FFC [36], across different scenarios, topologies, traffic matrices. Our evaluation demonstrates that TEDAT can (1) achieve 30 × speedup in admission control time at the expense of sacrificing 8% in performance compared to the optimal stragety; (2) increase the percentage of demands that satisfy their availability SLAs by up to 40%; (3) hold 30% more profit for the provider when network fails.

Table 1: Services have different availability targets.

Service	Availability(%)	Credit
Traffic Manager [8]	< 99.99%	10%
VPN Gateway [8]	< 99.95%	10%
VM Instances [5]	< 99.99%	10%
Cosmos DB [8] (Azure)	< 99.999% < 99%	10% 25%
DMS [7] (AWS)	< 99.99% < 99.0% < 95%	10% 30% 100%
AppFlow[6] (Amazon)	< 99.99% < 99.95% < 95%	10% 25% 100%
SMS[2] (Alibaba)	< 95% < 90%	10% 30%
Data Transmission[1] (Alibaba)	< 99.9% < 99.0% < 95%	15% 30% 100%

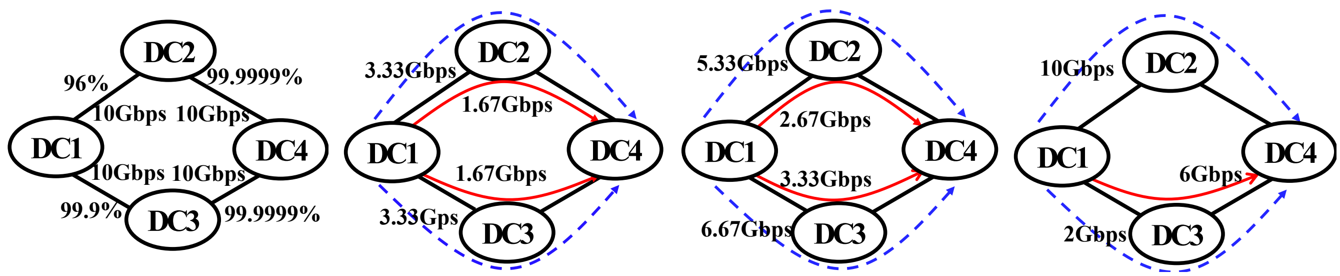
2 BACKGROUND AND MOTIVATION

In this section, we give an introduction of availability and inter-DC WAN failures. Then we use an example to demonstrate the limitations of state-of-the-art traffic engineering schemes.

2.1 Availability and inter-DC WAN failures

How to maintain high performance is a big problem for network operators. Frequently network failures can result in SLA violations and eventually loss of revenue to service providers [3–5, 8].

WAN failures can be frequent and heavy-tailed distribution: Failures could emerge from anywhere across the network, and from control plane to data plane [10]. Besides, failure events could be persistent with long durations. According to the report of Google, more than 80% of the failure duration is between 10 mins and 100 mins over B4 [3, 18]. For some large failure events, before finding the root cause, they will try to drain services away from affected clusters [18]. The service resumption after such failures can even take one day, leading to severe performance degradation and



(a) Link failure probability (b) FFC (c) TEAVAR (d) TEDAT

Figure 1: Different schemes comparison. According to their SLAs: service1's (red) minimal bandwidth demand of 6Gbps from DC1 to DC4 should be guaranteed at least 99% time and service2's (blue) minimal bandwidth demand of 12Gbps from DC1 to DC4 should be guaranteed at least 90% time. Link capacity is 10Gbps everywhere. Neither service gets enough bandwidth or availability guarantees under FFC or TEAVAR. Both services are in line with their SLAs under TEDAT**.**

revenue loss. Therefore, *when planning for resource allocation, network risks should proactively be taken into consideration.* Moreover, according to the measurement of previous literatures [17, 44], most links experience few failures and the failure rate is heavy-tailed distribution. Failure probability of a single link differs by even more than three orders of magnitude [10, 16].

Availability directly translates money: Nowadays, providing high availability is one of the main items in SLAs [5, 7, 8] and customers are eligible for a credit towards a portion of their service fees if there are SLA violations. We survey the SLA claims of some services from different cloud providers, then Table 1 shows their declared availability targets and corresponding refunding policies when violated. We can see that service providers often use a progressive piecewise function to derive refunding credits for violating availability targets. For example, the Microsoft Azure provides 10% refund between 99.99% and 99.0% availability, 30% refund for anything below 99.0% availability for the Traffic Manager service [8]; How to offer availability guarantees to traffic under network failure is becoming a big challenge.

The one-size-fit-all availability target is not enough: In recent years, there has been a rapid increase in deploying online services (online videos, online game, online shopping, live broadcast, etc.) 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 [30, 37, 42]. Tenants might launch multiple service instances over different data centers globally at any time. Inter-DC WAN bandwidth is a valuable resource and can become bottleneck [20] when services *contend* for network resource. In reality, availability targets vary with services, as Table 1 shown, the minimal availability demands of Data Transmission[1] and Short Message Service[2] are 95% and 90%, respectively. Therefore, TEAVAR's guarantee such as "Users' bandwidth should be guaranteed at least β% time " [10] is not enough, since it fails to differentiate availability targets of services.

2.2 Motivating TEDAT

Recently, some traffic engineering algorithms have already proactively considered availability in traffic engineering. FFC[36] seeks to assign a total bandwidth f_k to each node pair k. It achieves this by reserving bandwidth on each tunnel between k and ensuring the total bandwidth of all tunnels is larger than f_k under l concurrent node/link failure scenarios. TEAVAR [10] ensures that all scenarios meeting bandwidth demand d_k has a total probability no less than the required availability target. TEAVAR maximizes the network utilization subject to meeting a *single* desired availability target (e.g., 99.99%). In this part, we demonstrate that reasoning about availability in terms of the number of concurrent failures is often too conservative while meeting the same availability target for diverse services is often not enough.

Figure 1(a) shows a network topology, where DC1 is connected to DC4 via DC3 and DC2. The capacity of each link is 10Gbps and we also show the link failure probability in this figure. There are two paths from DC1 to DC4: (1) DC1$\rightarrow$ DC2 $\rightarrow$ DC4, (2) DC1 $\rightarrow$ DC3 $\rightarrow$ DC4, whose available probabilities are 95.999904% (i.e., 96% × 99.9999%) and 99.8999001% (i.e., 99.9% × 99.9999%). There are two services deployed by a tenant, where service1's (red) minimal bandwidth demand of 6Gbps from DC1 to DC4 should be guaranteed at least 99% time and service2's (blue) minimal bandwidth demand of 12Gbps from DC1 to DC4 should be guaranteed at least 90% time. We demonstrate the bandwidth allocation under FFC-1 scheme (which can tolerate one network failure) in Figure 1(b). We can see that FFC can support total 10Gbps bandwidth almost 99.9999% time (unless path (1) and (2) fail simultaneously), where service1 and service2 can gain 3.34Gbps and 6.66Gbps, respectively. Neither service is able to get enough bandwidth, because FFC treats the two paths indifferently without their individual failure rates. However, path (2) has much smaller failure probability and lowing its utilization does not help improve availability and is wasteful. TEAVAR exploits the different failure probability of network and advocates to fully utilize links under a loss-based availability target constraint (e.g., 99%). TEAVAR's solution, shown in Figure 1(c), can guarantee service1 and service2 attaining 6Gbps and 12Gbps about 95.9% time. TEAVAR fails to satisfy availability target of service1, since it allocates network resources under a single availability target and ignores the different expectations of the two services. As a result, it can't meet service1's availability demand.

Our approach: Observe that the bandwidth allocation of our method TEDAT is shown in Figure 1(d), where service1 can derive 6Gbps over 99% time (i.e., 99.9% × 99.9999%) and service2 can obtain 12Gbps over 95.9% time (i.e., 99.9% × 99.9999% × 96% × 99.9999%). TEDAT assigns service1 to path (2), while the higher failure probability path (1) is only used by service2. Hence, both service1 and service2 are not only in line with SLAs, but also gain enough bandwidth.

3 TEDAT SYSTEM

In this section, we show the design of TEDAT (Traffic Engineering for Diverse Availability Targets) system, which is able to provide high bandwidth-based availability to services. TEDAT system tries to achieve the following objectives:

- **High acceptance ratio:** Tenants' demands might arrive at anytime. The system should be able to accommodate as many demands as possible under the constraint of availability and bandwidth targets.
- **Maintain high availability:** The system must make services with stringent availability requirements (e.g., 99.999%) pass links with high reliability. Therefore, their bandwidth demands can be guaranteed with high probability.
- **Handle failures:** When there are link failures, the system can reroute traffic that would go over failed links over the remaining active tunnels with small revenue loss instead.
- **Practical & scalability:** The system must be able to admit or reject tenants' new demands in near real-time fashion and the system needs to scale up to dozens of DCs.

3.1 Availability abstraction

Tenants could deploy diverse service instances over Inter-DC WAN. In reality, a tenant could be a customer who launches multiple virtual private clouds over multiple DCs in public clouds (e.g., Amazon AWS) or could be a service team who launches multiple VM instances in private clouds (e.g., Google internal DCs). Services have different bandwidth demands as well as availability targets. However, there exists no interface for tenants to specify them. We now present TEDAT 's availability abstraction, which aims to guarantee the bandwidth-based availability of services.

Demand model: We model the inter-DC WAN as a graph $G = (V, E)$, where V and E are the nodes and link sets. Each link $e \in E$ is associated with a link capacity c_e (e.g., in bps). DC pair set is K and β_j presents the desired availability target of service $j \in J$. Each tenant's demand can be presented as a tuple $\{\alpha, \theta, \delta, \gamma, \mu\}$, where α and δ are the arrival and departure of the demand, γ denotes the service type and $\theta = \{d_1, d_2, ..., d_K\}$ presents bandwidth over each node pair. Service providers will offer refunding for any violating availability targets (see §2) and we use μ to denote the penalty.

Network model: A network scenario $z = \{z_1, z_2, ...\}$ is a vector consisting of each link's state. Each element $z_i \in \{0, 1\}$ denotes whether link i is up ($z_i = 1$) or down ($z_i = 0$). For each link e, operators can examine historical data and track whether e was up or down in a measured time epoch (e.g., 1 min). The up probability p_i of link i can be given by the up epoch percentage and its failure probability is $1 - p_i$. Let Z present the network scenario set and $p(z)$ is the probability of scenario $z \in Z$. Let z'_i denote the value of z_i, and assume link failures are independent. Similar to TEAVAR [10], probability of network scenario z is given by:

$$\begin{aligned} p(z) &= p\left(z_1 = z'_1, z_2 = z'_2, z_3 = z'_3, ..., z_e = z'_e\right) \\ &= \prod_{i=1}^{|E|} (z'_i p_i + (1 - z'_i)(1 - p_i)) \end{aligned} \tag{1}$$

For example, an inter-DC WAN contains three links $E = \{e_1, e_2, e_3\}$. Network scenario $\{1, 1, 0\}$ means e_1 ,e_2 are available and e_3 fails. If the up probability of e_1, e_2, e_3 are 0.9, 0.8, 0.85, respectively. Then its probability $p(\{1, 1, 0\}) = 0.9 \times 0.8 \times 0.15 = 0.108$.

3.2 System design

The system contains one controller, multiple brokers and multiple clients. Figure 2 shows the architecture of TEDAT system. It works as follows: When a tenant submits a demand to the controller, the Admission Control module will check whether the request can be admitted or not (see § 4.1). If the demand is admitted, the system will pre-allocate resource to the tenant and pass it to Online Scheduler module, while the rejected demands can be resubmitted by the tenants latter. The Online Scheduler module performs traffic engineering algorithm (see § 4.2) every T (e.g., 10) minutes to enforce

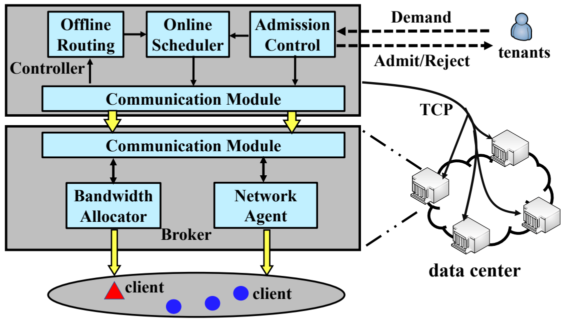


Figure 2: Architecture of TEDAT**.**

bandwidth of each demand. To prevent much revenue loss, the Online Scheduler module also derives the backup allocation under the assumption that some fragile links could fail (see § 4.3) every T. Each data center has a broker that manages the hosts inside. Brokers track topology and apportions the allocated rate to each host. Each host belongs to a service and it maintains a client to enforce bandwidth.

Controller is the brain of the whole system and it has three main modules: (1) Offline Routing. The controller maintains the WAN level network topology. When topology changes (e.g., remove a node) are reported to controller, the TE Routing module will perform routing algorithm (oblivious routing [32], k-shortest path [20], etc.) to derive all the possible tunnels between each node pair (i.e., $T_k, \forall k \in K$) offline. Tunnel set is then sent to TE Scheduler module as the input; (2) Online Scheduler. Every T minutes, this module will perform traffic engineering algorithm (see § 4.2). In addition to TEDAT, the prototype framework can support other TE algorithms (e.g., SWAN [20], FFC [36]). After deriving TE solution, it apportions the bandwidth allocation results over each tunnel (i.e., f_{ikt}) to corresponding brokers; It also some fragile links could fail and derives the backup allocation (see § 4.3) every T. (3) Admission Control. When a demand is submitted, the Admission Control module would perform admission algorithm (see § 4.1) to reply tenants in near real-time fashion; (4) Communication Module. This component is responsible for communication by building long TCP connections with brokers. The controller orchestrates all activity with a global view and is responsible for allocating WAN level bandwidth. To handle controller failure, the controller also replicate controllers and places them in different data centers, then Paxos [33] is used to elect one controller as the master.

Broker manages the hosts in its data center and it has the following key modules: (1) Bandwidth Allocator. After deriving the requests' bandwidth allocation results (i.e., f_{ikt}) from controller, this module will send them to the corresponding hosts; (2) Network Agent. It tracks topology and traffic with the aid of switches. It reports news about topology changes to the controller right away. It also runs OpenFlow controller (e.g., [13]) to work with commodity OpenFlow switches; (3) Communication Module. This component is responsible for communication with controller.

Client is deployed on each host. The client will enforce sending rate by the bandwidth enforcement module. The bandwidth enforcement module consists of an enforcement kernel module and an enforcement daemon. The enforcement daemon interacts with the site brokers to obtain rate limits. The enforcement kernel module is located between TCP/IP stack and the Linux Traffic Control (TC) module[35]. It intercepts all outgoing packets. Then these packets are directed to TC for rate limiting.

TEDAT system adopts OpenFlow[40] to install and update forwarding rules. To reduce forwarding complexity, our prototype uses label-based forwarding. We adopt VxLAN ID which contains 24 bits, where the first 12 bits to present demands and the remaining 12 bits are used to distinguish tunnels between each node. Therefore, $2^{12} = 4096$ tenants' demands and $2^{12} = 4096$ tunnels can be supported simultaneously. Flows are marked with a label at the ingress switch and the remaining switches can just read the labels and forward the packets. To implement unequal bandwidth splitting, our prototype uses group table in its OpenFlow pipeline. Flow table just maps the packet based on the destination and the group tables consist of bandwidth weight for each label.

Table 2: Key Notations for TEDAT **framework.**

	Input Variables
$G(V, E)$	Inter-DC WAN with nodes V and Links E
$k \in K$	A node pair
c_e	Remaining capacity of link e
L_{te}	Whether tunnel t passes link e
$j \in J$	A service
$i \in I_a$	A demand from tenant
d_{ik}	Bandwidth demand through node pair k of i
γ_i	Service type (i.e., j) of demand i
β_j	Bandwidth-based availability target of service j
T_k	Tunnel set through node pair k
$z \in Z$	A network scenario
$x_t(z)$	Whether tunnel t is available under z
$p(z)$	Probability for network scenario z
	Output Variables
f_{ikt}	Bandwidth of demand i over tunnel $t \in T_k$
g_i	Whether demand i can be admitted or not
h_i	Profit of demand i

4 TEDAT

In this section, we make a full description of TEDAT framework, which contains three parts, i.e., admission control, traffic engineering and failure recovery. We consider tunnel-based forwarding [10, 20, 36], where traffic is carried over a set of tunnels. For each node pair $k \in K$ of the inter-DC WAN $G = (V, E)$, we are given a set of pre-selected tunnels T_k, where tunnel set T_k can be derived with different routing schemes (e.g., k-shortest paths, edge disjoint paths [46], oblivious routing [32]). Each tunnel $t \in T_k$ contains a set of links and L_{te} presents whether tunnel t contains link $e \in E$ or not. f_{ikt} is the bandwidth allocated over tunnel $t \in T_k$ for demand i and I_a is the total *admitted* demand set form tenants. Let $x_t(z)$ denote whether tunnel t is available (i.e., $x_t(z) = 1$) or not (i.e., $x_t(z) = 0$) under network scenario z. Table 2 shows the main notations. The framework intends to achieve three goals: (1) Accommodate as many demands as possible subject to network failure model, diverse service availability and bandwidth targets; (2) Guarantee the bandwidth-based availability target for each admitted demand; (3) Minimize revenue loss when network fails.

4.1 Admission control

Tenants' demands come in the first-come-first-service (FCFS) manner and *no preemption* is allowed. The new arrival demand, together with the admitted ones (i.e., I_a), constitute demand set I. If every element in I can meet its availability target, the new arrival demand can be admitted, otherwise, it is rejected. To accommodate as many demands as possible, we can model the admission control as a 0-1 Mixed-integer linear programming and maximize the number of demands that satisfy availability targets. Appendix A shows the optimization formulation. The admission control optimization problem can be proven as NP-hard problem by reducing the all-or-nothing multi-commodity flow problem [11] to a special case of it. We omit the details for brevity.

In reality, tenants' demands can arrive at anytime and admission control should reply to the tenants in near real time. There is a *tradeoff* between efficiency and optimal solution: If we assume all the admitted demands are fixed and can't be rescheduled, then we can derive the solution fast but the new arrival one might be unable to be accommodated; If we derive the optimal solution by solving the optimization problem shown in Appendix A, then we can accommodate more demands, however, this is time-consuming. To achieve a tradeoff, we would like to address the following three steps:

(1) When a new demand i arrives, we assume that all admitted demands are *fixed* and check whether the remaining network capacity and the corresponding links' up probability can support it. If this is true, then go to step (3), otherwise, go to step (2).
(2) Run Algorithm 1 and check whether all admitted demands can be rescheduled to accommodate the new arrival demand i. If this is true, then go to step (3), otherwise, the system will reject the demand, which can be resubmitted by tenants again latter.
(3) Pre-allocate bandwidth to the demand with remaining network capacity and pass all the admitted demands to the traffic engineering algorithm.

To derive the solution when all the admitted demands can be rescheduled, we propose Algorithm 1, which is able to gain the solution in a fast manner. The algorithm will choose demand with small bandwidth and availability targets each iteration (Line 4). Then it checks whether the remaining network capacity can satisfy its bandwidth demand. If this is false, then the network is unable to support the request (Line 6-8), otherwise, it prioritizes tunnels with small remaining capacity and up probability (Line 10). It allocates bandwidth with the remaining capacity of the tunnel until bandwidth demand between the node pair is fulfilled (Line 11-13). The algorithm will return true when all demands' bandwidth-based availability targets in I are satisfied (Line 19). Algorithm 1 is able to derive allocation results in $O(I * K * max(|T_k|))$. Algorithm 1 can derive a tight solution:

Algorithm 1: Readmission control algorithm

Input: Input parameters shown in Table 2
Output: $\{g_1, g_2, .., g_i, ...\}, \{f_{ikt}\}$

1 $g_i = 0, s_i = 1, \forall i \in I$;
2 $f_{ikt} = 0, \forall i \in I, k \in K, t \in T_k$;
3 **while** *true* **do**
4     $i = \arg_{i \in I} min\{\sum_{k \in K} d_{ik} \times \beta_{\gamma_i}\}$;
5   **for** $k \in K$ **do**
6   **if** $d_{ik} > Capacity(T_k)$ **then**
7             $g_i = 0$;
8   **return** $\{g_1, g_2, .., g_i, ...\}, \{f_{ikt}\}, False$;
9   **while** $d_{ik} > 0$ **do**
10             $t = \arg_{t \in T_k} \min\{c_t * p_t\}$;
11             $f_{ikt} = Allocation(t, d_{ik})$;
12             $s_i = s_i * p_t$;
13             $d_{ik} = d_{ik} - f_{ikt}$;
14   **if** $s_i < \beta_{\gamma_i}$ **then**
15         $g_i = 0$;
16   **return** $\{g_1, g_2, .., g_i, ...\}, \{f_{ikt}\}, False$;
17     $g_i = 1$;
18     $I = I \setminus i$;
19 **return** $\{g_1, g_2, .., g_i, ...\}, \{f_{ikt}\}, True$;

LEMMA 4.1. *There must exist an allocation scheme to satisfy the bandwidth-based availability targets of all demands admitted by Algorithm 1.*

The proof details can be found in Appendix B

4.2 Online scheduling

For each *admitted* demand, the online scheduling determines bandwidth allocation over each tunnel every T (e.g., 10 mins). The goal of online scheduling is to satisfy each demand's bandwidth-based availability target. We model the online scheduling as a linear programming. Firstly, the bandwidth allocation results f_{ikt} for demand i over tunnel $t \in T_k$ should be non-negative, i.e.,

$$f_{ikt} \geq 0, \quad \forall i \in I, k \in K, t \in T_k. \tag{2}$$

Then total traffic through link e should not be larger than its capacity c_e:

$$\sum_{i \in I} \sum_{k \in K} \sum_{t \in T_k} f_{ikt} L_{te} \leq c_e, \quad \forall e \in E. \tag{3}$$

To provider high availability with minimum induced overload, we restrict each demand's total allocated bandwidth smaller than its real need, i.e.,

$$\sum_{t \in T_k} f_{ikt} \leq d_{ik}, \quad \forall i \in I_a, k \in K \tag{4}$$

$R_{ik}(z)$ is the ratio of total reserved bandwidth to demand through pair k for i under network scenario z and it is defined as:

$$R_{ik}(z) = \frac{\sum_{t \in T_k} f_{ikt} x_t(z)}{d_{ik}}, \quad \forall i \in I, z \in Z, k \in K. \tag{5}$$

$R_{ik}(z)$ describes the allocation return under network scenario z. In reality, tunnel t might be unavailable (i.e., $x_t(z) = 0$), and if the total reserved bandwidth through all the available tunnels is larger than $d_{ik}, \forall k \in K$, then bandwidth demand can still be satisfied and network scenario z can be regarded as *qualified*, i.e., $R_{ik}(z) \geq 1, \forall k \in K$. To guarantee the availability target of a demand, we should make its total probability of qualified scenario larger than availability target, i.e., $\sum_{\forall k: R_{ik}(z) \geq 1} p(z) \geq \beta_i, \forall i \in I_a$. Let $B_i(z)$ denote the lower bound of $R_{ik}(z)$ over the k pairs for i under network scenario z, i.e.,

$$B_i(z) \leq R_{ik}(z), \quad \forall i \in I_a, z \in Z, k \in K \tag{6}$$

It is obvious $B_i(z) \leq 1$ and $B_i(z) = 1$ presents scenario z is qualified, therefore, we can define $B_i(z) \times p(z)$ as the potential achieved availability of i under network scenario z. To satisfy diverse availability targets, the achieved availability should be larger than availability target, i.e.,

$$\sum_{z \in Z} B_i(z) \times p(z) \geq \beta_{\gamma_i}, \quad \forall i \in I_a \tag{7}$$

We want to maximize the overall achieved availability of all admitted demands, thus, we can finally give the formulation of traffic engineering formulation:

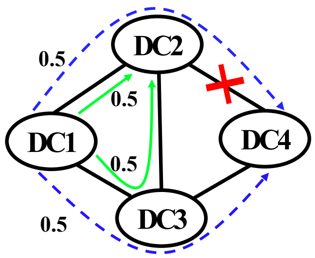
$$\begin{aligned} &maximize \sum_{i \in I_a} \sum_{z \in Z} B_i(z) \times p(z) \\ &s.t. (2), (3), (4), (5), (6), (7) \end{aligned} \tag{8}$$

We can see that although the traffic engineering module is LP problem, it considers each network scenario, which makes problem complexity increase exponentially with network size. For instance, B4 [22] topology has 12 nodes and 38 links, then there are $2^{38} - 1$ network failure scenarios in total. Therefore, an important question is *how to effectively reduce the size of traffic engineering problem?*

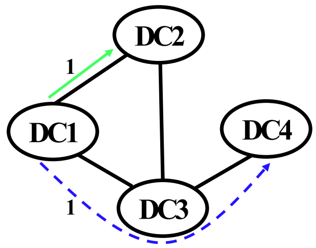
Previous literatures such as TEAVAR [10] advocate to prune scenarios which has smaller probability than a threshold, therefore, problem scale will reduce. This method is easy but the accuracy and efficiency tradeoff threshold is hard to decide. A large threshold could significantly accelerate the algorithm since large amount of network scenarios are cut off, however, the accuracy might decrease. Instead, we propose to cut off network failure scenarios in which *concurrent link failure number* is larger than l. We think this method is much more practical since most scenarios could hardly happen (e.g., more than two links fail simultaneously). Therefore, we can ignore these scenarios. To provide an upper bound of the traffic engineering problem, we collapse all the pruned scenarios into a single scenario with probability equal to the sum of probabilities of the pruned scenarios and regard that scenario as an unsafe one.

4.3 Failure recovery

In TEDAT, traffic can also be redistributed across the surviving tunnels via rerouting when any tunnel becomes unavailable. In reality, service providers have to derive refunding credits for violating availability targets (see §2). TEDAT proactively considers failure scenarios and pre-computes backup allocation to maximize total revenue, and the surviving tunnels can be used immediately to support tenants. For example, Figure 3 shows two services are from DC1 to DC4 and both of them need bandwidth demand of 2. Link capacity is 1 everywhere. Figure 3(a) shows the original traffic engineering allocation and it is used when no failures happen. Figure 3(b) depicts the backup allocation result when we assume the link between DC2 and DC4 is down, and the result will be put into practice when link DC2 $\rightarrow$ DC4 failure is detected. Network scenario number is huge for large-scale network and we assume that at any time, only one link in



(a) Original TE allocation



(b) Backup allocation

Figure 3: There are two services from DC1 to DC4 and link capacity is 1 everywhere. (a) shows the original TE allocation. (b) shows the backup allocation result when assuming link DC2 → DC4 is broken.

the network could fail. If we assume network scenario z could happen, and let f^z_{ikt} denote bandwidth of i over tunnel $t \in T^z_k$, where T^z_k is the surviving tunnel set under scenario z and it should be non-negative, i.e.,

$$f^z_{ikt} \geq 0, \quad \forall i \in I, k \in K, t \in T^z_k. \tag{9}$$

Let A^z_e denote whether link e is available under scenario z. The total bandwidth should be smaller than capacity for each available link, i.e.,

$$\sum_{i \in I_a} \sum_{k \in K} \sum_{t \in T^z_k} f^z_{ikt} L_{te} \leq c_e \times A^z_e \tag{10}$$

The revenue return for i over node pair k can be described:

$$R_{ik} = \frac{\sum_{t \in T^z_k} f^z_{ikt}}{d_{ik}}, \forall i \in I_a, k \in K \tag{11}$$

If the allocated bandwidth of $i \in I_a$ between each node pair is larger than its need (i.e., $R_{ik} \geq 1, \forall i \in I_a$), the profit unit is 1, otherwise, there is a penalty, which can be derived from SLA and we use μ_{γ_i} to denote its penalty. Then, the profit of i is:

$$\forall i \in I_a : h_i = \begin{cases} 1 & \forall k \in K : R_{ik} \geq 1 \\ \mu_{\gamma_i} & \text{Otherwise} \end{cases} \tag{12}$$

It is a step function and we can change it to the following linear format and we can also change it into a linear format:

$$\begin{cases} h_i = y_i + \mu_{\gamma_i} * (1 - y_i), & \forall i \in I_a. \\ R_{ik} < M * y_i + 1 - y_i, & \forall i \in I_a, k \in K. \\ R_{ik} >= y_i, & \forall i \in I_a, k \in K. \\ y_i \in \{0, 1\}, & \forall i \in I_a. \end{cases} \tag{13}$$

Where M is a big integer that is at least larger than the upper bound of R_{ik}. Let w_i denote the profit of demand i

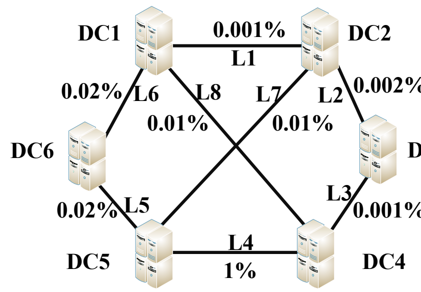


Figure 4: Testbed topology. Each link has the same capacity 1Gbps but different failure probability.

when its availability target can be satisfied. The goal of failure recovery is to maximize profits of all the admitted demands, and we can give the problem formulation finally:

$$\begin{aligned} & maximize \sum_{i \in I_a} w_i \times h_i \\ & s.t. (9), (10), (11), (13) \end{aligned} \tag{14}$$

The auxiliary output variables y_i are chosen from {0,1}, so that the failure recovery problem is a 0-1 Mixed-integer linear programming.

Lemma 4.2. *The failure recovery problem is NP-hard.*

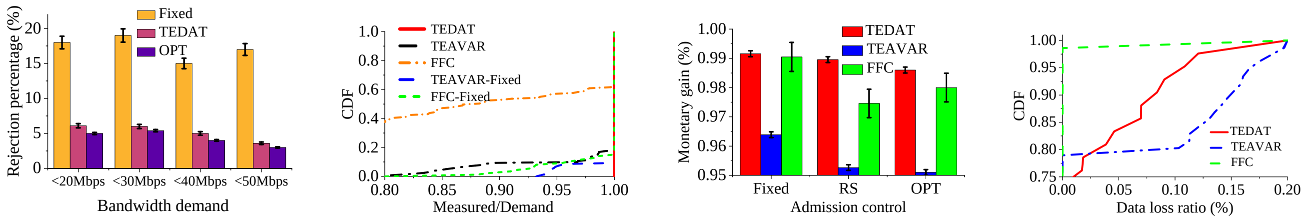
The detailed proof can be found in Appendix C.

As the failure recovery problem is NP-hard and one important question is *can we develop an efficient algorithm to solve it in polynomial time?* In this case, we propose a greedy algorithm to attain an approximate solution whose pseudocode is shown in Appendix D. The key idea of the greedy algorithm is to give priority to demands in non-decreasing order by the ratio of profit to its efficient bandwidth demands. *We also prove the approximation of the greedy algorithm is 2 in Appendix D.*

5 EVALUATION

In this section, we evaluate the performance of TEDAT both in real trace driven simulation and testbed. We show the performance of TEDAT through answering the following specific questions:

- **What is the benefit of** TEDAT **?** In § 5.1.3, we show TEDAT proactively considers link failure probability as well as the diverse demands of services proactively and it increases the percentage of services that satisfy their availability SLAs by up to 40%.
- **How does** TEDAT **compare with the state-of-the-art traffic engineering TE schemes?** In § 5.2.1, we



(a) Admission control (b) Measured and demand ratio (c) Monetary gain after failures (d) Data loss ratio

Figure 5: A public cloud example. Tenants' demands arrive obeying poisson process and the duration of demands are with exponential distribution.

show the consistent performance of TEDAT under various topologies and traffic. TEDAT performs up to 30%, 40%, 50%, 75% better than TEAVAR[10], SMORE[32], SWAN[20] and FFC[36] on bandwidth-availability guarantee, respectively.

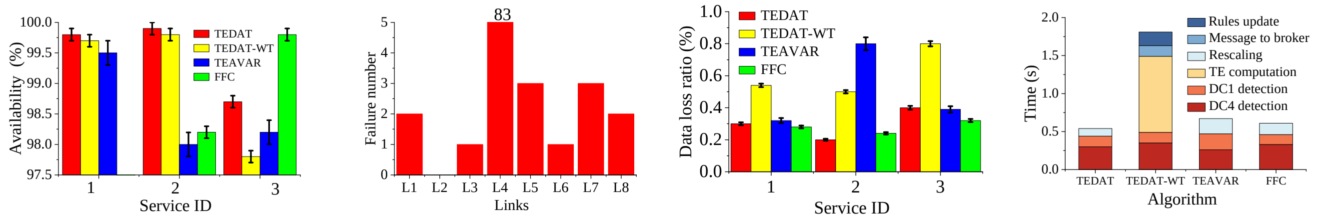
- **How effect is** TEDAT **under different parameter settings ?** In § **??**, we show the stable performance of TEDAT framework. It doesn't depend on specific profit function, routing schemes, etc.
- **How robust is the solution?** In § **??**, we evaluate the accuracy and efficiency of the solutions. The average error between the optimal and approximate solutions is within 8% and we also show the algorithm is fast and accuracy enough to use in large scale network.

5.1 Testbed evaluation

5.1.1 Testbed setup. We build a small testbed with 6 servers to emulate an inter-DC WAN with 6 DCs as in Figure 4. Each server equips with 4 Intel Xeon E5-2620 CPUs, 64GB memory and 4 Ethernet NICs, and runs Centos 7 64-bit version with Linux 4.15.6 kernel [29]. Each inter-DC WAN link is emulated using 1Gbps physical link. We use 4-shortest path to decide tunnel set of each node pair. Controller locates in DC1 and runs the software defined networking framework. We add 100ms delay to emulate the WAN environment. We start 21 VMs (1 vCPU, 2GB memory, 100GB disk) in each server, where 20 VMs are used to emulate the hosts and 1 VM runs as the broker. Each VM belongs to only one customer and will send *iperf* UDP flows with specified rates to other DCs. We measure total bytes loss according to *iperf* server side reports and packet counters in the switches, then finally derive the data loss ratio. The network agent in broker runs Floodlight [13] as the OpenFlow controller. In each data center, all hosts and NICs connect to an Open vSwitch [41]. We start an extra VM in DC1 as the controller. All switches run link liveness detection protocol, and they report any failures to brokers. 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 4, we disable the network interface to emulate link failure. Then after 3 seconds, we enable the network interface to emulate link repair. We also evaluate FFC [36] and TEAVAR [10] to make comparison.

5.1.2 A public cloud example. We firstly evaluate with a public cloud case. Assume tenants arrive with poisson process with arrival rate 2 every minute and the duration of each bandwidth demand is under exponential distribution with a mean of 5 minute. Tenant' bandwidth demands between each node pair are under uniform distribution between 10Mbps and 50Mbps. The profit of each tenant (i.e., w_i) is proportional to his/her total bandwidth demand. Tenants' availability targets and refunding ratio are from the SLAs of Azure [8]. We perform each experiment 100 minutes and repeat the process 20 times, where the error bars paints the maximal, minimal and mean value.

Figure 5(a) demonstrates TEDAT only performs slightly worse than the optimal solution (about 4%) and is at least 2 $\times$ better than the *Fixed* admission control. This is because TEDAT fully considers the availability diversity of tenants. In contrast, *Fixed* admission control can't change tenants' routing during runtime and such inflexibility leads to higher rejection rate. Figure 5(b) shows once a tenant's demand is admitted, TEDAT can guarantee bandwidth. FFC and TEAVAR with *Fixed* admission control perform slightly better than when they are without admission control since many tenants' demands are rejected, but they also perform worse than TEDAT, because FFC is too conservative and and TEAVAR ignores the diverse demands. Figure 5(c) shows TEDAT can retain about 15% more money ratio than FFC and TEAVAR when network fails. This is because TEDAT pre-computes backup allocation with maximal profit and they can be put into practice as far as failures are detected. Figure 5(d) demonstrates data loss ratio after network fails, where TEDAT performs about 8% worse than FFC. FFC performs best since it



(a) Bandwidth-based availability (b) Link failures (c) Data loss ratio (d) Events after L3 failure

Figure 6: A private cloud example. We deploy three services whose details are shown in Table 3.

Table 3: Scheduled results of different schemes.

Service	paths	TEDAT	TEAVAR	FFC
Service-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
Service-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
Service-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
CERNET2	14	32

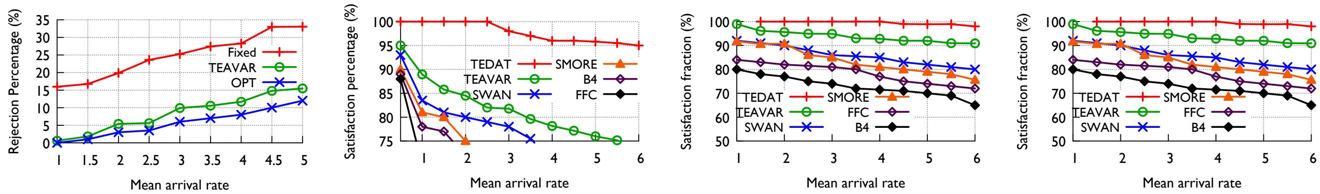
keeps network utilization low and almost no congestion occurs when network fails, TEDAT can decrease performance loss by quick recovery.

5.1.3 A private cloud example. We now test the performance under a simple private cloud example, where three services are deployed and all admitted. Service-1 is from DC1 to DC3, Service-2 is from DC1 to DC4 and Service-3 is from DC1 to DC5. Their bandwidth demands are 1000Mbps, 500Mbps and 1500Mbps, respectively. Assume the availability targets of Service-1 , Service-2 and Service-3 are 99.5%, 99.9% and 95%, respectively. All services last 100s and each experiment repeats 100 times, where the error bars paints the maximal, minimal and mean value. TEDAT-WT refers TEDAT without failure recovery.

Figure 6(a) shows the bandwidth-based availability which is measured as the time qualification ratio. A second is regarded as qualified if the gap between total measured bandwidth and bandwidth demand is less than 1%. We can see that all services can reach their availability targets under TEDAT, while Service-1 fails under FFC and Service-2 fails under both TEAVAR and FFC. To explore the reason, we show the scheduled results of the three schemes in TABLE 3: (1) FFC reserves too much bandwidth to protect routing and service1 is unable to gain enough bandwidth; (2) TEDAT abandons link L4 for Service-2, while TEAVAR adopts it. According to the link failure times shown in Figure 6(b), we can see that most failures are from link L4 in our experiment. Although TEAVAR considers link failure probability, it ignores the availability targets of different services. Therefore, Service-3 gains redundant bandwidth and affects Service-2 finally. Data loss mainly comes from congestion and blackhole, where congestion losses are always link oversubscribed and blackhole losses occur during the time between a link fails and ingress switch rescale. Figure 6(c) demonstrates TEDAT-1 and FFC performs better than TEDAT-2 and TEAVAR for data losses ratio. Data losses of Service-2 under TEAVAR is high due to Link L4. Also, TEDAT-2 (TEDAT without rescaling) performs worse than the other three schemes, because it takes much time to derive new resource allocation when network fails. Figure 6(d) shows the time of events after link L3 fails. When L3 fails, DC1 and DC4 take about 0.4s to detect the failure. TEDAT-1 can rescale traffic with the backup allocation results, therefore, packet losses will stop immediately.

5.2 Simulation evaluation

Simulation setup. We evaluate the performance of TEDAT under four network topologies: B4 [22], ATT [10], IBM[32], CERNET2 (China Education and Research Network), where the first three topologies (including their traffic metrics) are obtained from the authors of TEAVAR[10] and CERNET2 is measured by ourselves. Table 4 shows the topology details. To accommodate the reproducibility of link failures, we make the assumption that link failures fit Weibull distribution, which has been widely used in studying failures over



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

Figure 7: TEDAT **vs. various TE schemes under different network topologies.**

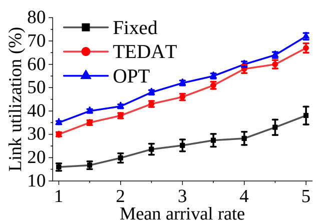


Figure 8: Weight difference.

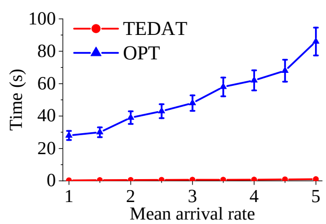


Figure 9: Weight difference.

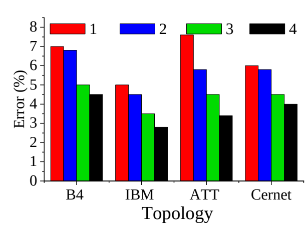


Figure 10: Weight difference.

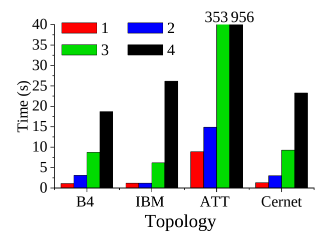


Figure 11: Weight difference.

backbones [38] and inter-DC WAN [10]. Link failure probability varies each time slot with Weibull distribution. Each run simulates 150000 1-minute timeslot (about 100 days). The Weibull distribution probability density function $f(\lambda, k)$ contains two parameters, where λ is shape parameter and k is scale parameter. We choose $f(0.8, 0.00001)$ as our default distribution, which is similar to [10]. Tenants' demands can arrive at anytime. Traffic engineering performs every 10 time slots. For traffic engineering part, we compare the performance of TEDAT against FFC [36], TEAVAR [10], SWAN [20], SMORE [32], B4 [22]. We change them to adapt service level traffic engineering: FFC tries to maximize the link utilization with considering l faults occur and we choose $l = 1$ as its default value. TEAVAR maximizes bandwidth allocation to each request subject to a single operator-specified availability target. The default availability target of TEAVAR is 99.9% in our simulations. SMORE [32] minimizes the maximum link utilization without explicit guarantees on availability. SWAN [20] tries to maximize the total throughput of all requests in the current slot. B4 [22] aims to deliver max-min fair allocation manner to each request. We consider 10 availability targets, whose value are from the SLAs of Azure [8]. Our collected traffic data contains the capacity of all links and the bandwidth demands between each DC pair. We generate requests with a Poisson process, the duration of each request is modeled as an exponential distribution with a mean of 1000 time slots.

5.2.1 TEDAT vs. state-of-the-art TE schemes. Figure 7 shows the consistent trend under various topologies: TEDAT performs about 20%, 25%, 30%, 40% better than TEAVAR, SMORE, SWAN, 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. Next, we take IBM trace as the example and perform the experiments.

Figure **??**(a) shows total profit comparison between TEDAT and other schemes. We can see that TEDAT performs best and its performance is even up to 30%, 40%, 50%, 75% better than TEAVAR, SMORE, SWAN and FFC when traffic scale is 3. Figure **??**(b) shows the percentage of services whose availability targets are satisfied for different importance levels when traffic scale is 3. In the picture, L, U, N, I, S are abbreviations of Lax, Unimportant, Normal, Important and Significant, and the error bar paints the mean and mean ± standard deviation value. We can see TEDAT performs even worse than other TE schemes for the Lax services, but it performs at least 25% better for the Important and Significant services. Specially, we can see that SWAN [20] also performs well for the importance services, but it allocates bandwidth only in strict precedence across three coarse priority classes and TEDAT is fine-grained priority and can describe the five level of importance. TEDAT aims to maximize the sum profit of all services, where a service's profit is defined in (**??**). Figure **??**(c) presents the CDF of achieved availability for different schemes when traffic scale is 3. We can see that about 87%

services's achieved availability are larger than 95% under TEDAT, which is at least 30% better than other schemes.

Solution error mainly comes from scenario pruning and greedy heuristic algorithm. Algorithm **??** cuts off the scenarios whose probability is smaller than the threshold. Algorithm 2 proposes a greedy algorithm to solve TEDAT problem. Let $Greedy$ denote the profit with greedy algorithm and OPT denote the profit of optimal solution. Then the error can be defined as $\frac{|Greedy-OPT|}{|OPT|}$. Figure **??** (a) shows the error of greedy algorithm with different thresholds. We can see the error is less than 8%. Traditionally, TE should update the network state every 5-10 minutes and though the powerful mathematical optimization solver such as Gurobi [?] can solve TEDAT problem, it might be too slow. Figure **??** (b) shows the time comparison between greedy and Gurobi on a server (4-core, 2.60GHz processor with 32GB) when they reach the same approximation. We can see that for large network (e.g., ATT topology) , Gurobi is too slow to converge and the greedy algorithm is necessary. While the bruce force algorithm is more than 1000 minutes for each topology and we don't paint them in the picture. We can see the result is fast enough to deploy in practice.

6 RELATED WORK

Traffic engineering for WAN. Optimizing WAN performance is a big challenge. Prior technologies such as OSPF[14] and MPLS tunnels [12, 27] can improve the backbone traffic transfer under the knowledge of traffic demands. Recently, SDN is widely used in WAN TE. SWAN[20], B4[21, 22], Bwe[31],OWAN[26] leverage the whole network information to allocate network bandwidth. Network scheduling schemes [23, 28, 48] also use SDN technology to decide the priority of traffic. Although these schemes can improve the network utilization, they ignore the *risk* of networks, thus, service availability can't be guaranteed.

NetworkÂărisk based traffic engineering. [18] studies the network failures and proposes the principles to design robust network, but it doesn't contain any TE framework. [15] suggests that backbone traffic engineering strategies should consider current and past optical layer performance. Neither [18] nor [15] contains TE frameworks. [39] models demand and risks as uncertainty, but it can't provide availability at the particular level (e.g., 90%). [9] can adjust traffic according to demands, but it ignores the link failures. Most of the traffic engineering schemes work in the *reactive* manner, such as [43],[47]. The reactive methods could take a long time to recover from faults, while applications have already been hurt[36]. FFC [36] and TEAVAR[10] are proactive routing protection methods, but they can't ensure the resources of services under network failures. TEDAT makes a further step and tries to provide service level bandwidth-based availability. The availability of TEDAT is stronger than them.

Bandwidth guarantee. Up still now, many existing works have already considered to guarantee the bandwidth of applications over inter or intra DCs. Deadline-aware schemes such as D^2TCP[45], LPD [50], Amoeba[49] try to make flows finish before a fixed time constraints, but they can't perform finely grained rate limits. Some cloud bandwidth limit technologies such as CloudMirror [34] and EyeQ [24], can provide bandwidth guarantees to applications or tenants, while they ignore the network risks and service availability requirements. TEDAT is different from in two aspects. Firstly, TEDAT considers the network failures and tries to perform rate limit under the assumption. Secondly, TEDAT can guarantee bandwidth at a particular level (e.g., 99%).

7 CONCLUSION REMARKS

We present TEDAT, a framework that attempts to guarantee the availability demands of applications. TEDAT aims to optimize the total profits of service subject availability targets. We design greedy and network scenario pruning algorithm to derive the solution. We evaluate the performance of TEDAT in real testbed as well as trace-driven simulations. The evaluation results demonstrate that TEDAT can improve bandwidth-based availability performance by up to 50%.

REFERENCES

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

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

[3] 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 (SOSP '19).* ACM, New York, NY, USA, 414–429. https://doi.org/10.1145/3341301.3359664

[4] Ahmed Alquraan, Hatem Takruri, Mohammed Alfatafta, and Samer Al-Kiswany. 2018. An Analysis of Network-Partitioning Failures in Cloud Systems. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation (OSDI'18).* USENIX Association, USA, 51–68.

[5] Amazon. 2019. Amazon Compute Service Level Agreement (2019). https://aws.amazon.com/compute/sla/. (2019).

[6] Amazon. 2020. Amazon AppFlow Service Level Agreement. https://aws.amazon.com/cn/appflow/sla/. (2020).

[7] Amazon. 2020. AWS Database Migration Service (AWS DMS) Service Level Agreement. https://aws.amazon.com/cn/dms/sla/. (2020).

[8] Azure. 2020. Microsoft Azure Service Level Agreements (2020). https://azure.microsoft.com/en-us/support/legal/sla/summary/. (2020).

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

[10] 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 (SIGCOMM '19). ACM, New York, NY, USA, 29–43. https://doi.org/10.1145/3341302.3342069

[11] 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.

[12] 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.

[13] floodlight. 2020. Floodlight controller. https://github.com/floodlight/floodlight. (2020).

[14] 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.

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

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

[17] 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 (SIGCOMM '11)*. ACM, New York, NY, USA, 350–361. https://doi.org/10.1145/2018436.2018477

[18] 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)*.

[19] Yotam Harchol, Dirk Bergemann, Nick Feamster, Eric Friedman, Arvind Krishnamurthy, Aurojit Panda, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. 2020. 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)*. Association for Computing Machinery, New York, NY, USA, 377âĂŞ389. https://doi.org/10.1145/3387514.3405875

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

[21] 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 (SIGCOMM '18)*. ACM, New York, NY, USA, 74–87. https://doi.org/10.1145/3230543.3230545

[22] 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 (SIGCOMM '13)*. ACM, New York, NY, USA, 3–14. https://doi.org/10.1145/2486001.2486019

[23] 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 (SIGCOMM '16)*. ACM, New York, NY, USA, 73–86. https://doi.org/10.1145/2934872.2934893

[24] 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

[25] 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 (SIGCOMM '20)*. ACM, New York, NY, USA, 139–153. https://doi.org/10.1145/3387514.3405858

[26] 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 (SIGCOMM '16)*. ACM, New York, NY, USA, 87–100. https://doi.org/10.1145/2934872.2934904

[27] 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 (SIGCOMM '05)*. ACM, New York, NY, USA, 253–264. https://doi.org/10.1145/1080091.1080122

[28] 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 (SIGCOMM '14)*. ACM, New York, NY, USA, 515–526. https://doi.org/10.1145/2619239.2626336

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

[30] 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 (IMC '12)*. ACM, New York, NY, USA, 211–224. https://doi.org/10.1145/2398776.2398799

[31] 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 (SIGCOMM '15)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/2785956.2787478

[32] 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

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

[34] 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 (SIGCOMM '14)*. ACM, New York, NY, USA, 467–478. https://doi.org/10.1145/2619239.2626326

[35] Linux. 2020. Linux Traffic Control. https://tldp.org/HOWTO/Traffic-Control-HOWTO/intro.html. (2020).

[36] 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 (SIGCOMM '14)*. ACM, New York, NY, USA, 527–538. https://doi.org/10.1145/2619239.2626314

[37] 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 (SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. https://doi.org/10.1145/3098822.3098843

[38] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot. 2008. Characterization of Failures in an Operational IP Backbone Network. *IEEE/ACM Transactions on Networking* 16, 4 (2008), 749–762.

[39] 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

[40] openflow. 2020. sdn and openflow. https://tools.ietf.org/html/rfc7426#page-23. (2020).

[41] 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

[42] 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.

[43] 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 (SIGMETRICS '11)*. ACM, New York, NY, USA, 97–108. https://doi.org/10.1145/1993744.1993756

[44] 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 (SIGCOMM '10)*. ACM, New York, NY, USA, 315–326. https://doi.org/10.1145/1851182.1851220

[45] 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

[46] 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*.

[47] 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 (SIGCOMM '10)*. ACM, New York, NY, USA, 291–302. https://doi.org/10.1145/1851182.1851218

[48] 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 (SIGCOMM '11)*. ACM, New York, NY, USA, 50–61. https://doi.org/10.1145/2018436.2018443

[49] 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 (EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 20, 14 pages. https://doi.org/10.1145/2741948.2741957

[50] 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.

A ADMISSION OPTIMIZATION PROBLEM

Firstly, the bandwidth allocation results f_{ikt} for demand i over tunnel $t \in T_k$ should be non-negative, i.e.,

$$f_{ikt} \geq 0, \quad \forall i \in I, k \in K, t \in T_k. \tag{15}$$

Then total traffic through link e should not be larger than its capacity c_e:

$$\sum_{i \in I} \sum_{k \in K} \sum_{t \in T_k} f_{ikt} L_{te} \leq c_e, \quad \forall e \in E. \tag{16}$$

$R_{ik}(z)$ is the ratio of total reserved bandwidth to demand through pair k for request i under network scenario z and it is defined as:

$$R_{ik}(z) = \frac{\sum_{t \in T_k} f_{ikt} x_t(z)}{d_{ik}}, \quad \forall i \in I, z \in Z, k \in K. \tag{17}$$

Under network scenario z, tunnel t might be unavailable (i.e., $x_t(z) = 0$). If the total reserved bandwidth through all the available tunnels is larger than $d_{ik}, \forall k \in K$, then bandwidth demand can be satisfied even tunnel t fails and network scenario z can be regarded as *safe* . Let $A_i(z)$ denote whether scenario z is safe (i.e., $A_i(z) = 1$) or not (i.e., $A_i(z) = 0$) for request i:

$$A_i(z) = \begin{cases} 1 & \forall k \in K : R_{ik}(z) \geq 1 \\ 0 & \text{Otherwise} \end{cases}, \forall i \in I, z \in Z. \tag{18}$$

It is a step function and we can change it to the following linear format:

$$\begin{cases} R_{ik}(z) < 1 - A_i(z) + M \times A_i(z), & \forall i \in I, z \in Z, k \in K. \\ R_{ik}(z) >= A_i(z), & \forall i \in I, z \in Z, k \in K. \\ A_i(z) \in \{0, 1\}, & \forall i \in I, z \in Z. \end{cases} \tag{19}$$

where M is a big integer that is at least larger than the upper bound of $R_{ik}, \forall i \in I, k \in K$. It is easily to prove (19) equals to (18). The achieved bandwidth-based availability of request i is the sum of all safe network scenarios' probability:

$$S_i = \sum_{z \in Z} A_i(z) \times p(z), \quad \forall i \in I. \tag{20}$$

We use g_i to present whether the availability target of i can be satisfied:

$$g_i = \begin{cases} 1 & 1 > S_i \geq \beta_{Y_i} \\ 0 & \beta_{Y_i} > S_i \geq 0 \end{cases}, \quad \forall i \in I. \tag{21}$$

If the achieved bandwidth-based availability (i.e., S_i) is larger than its desired availability target (i.e., β_{Y_i}), the demand can be admitted. Otherwise, it is rejected, which means the network can't support i. Also, (21) can be changed into the following linear format:

$$\begin{cases} S_i < \beta_{Y_i} \times (1 - g_i) + g_i, & \forall i \in I. \\ S_i \geq \beta_{Y_i} \times g_i, & \forall i \in I. \\ g_i \in \{0, 1\}, & \forall i \in I. \end{cases} \tag{22}$$

We intend to maximize the total accepted demands, i.e., $\sum_{i \in I} g_i$. With the constraints introduced above, we can finally give the formulation of admission problem:

$$\begin{aligned} &maximize \sum_{i \in I} g_i \\ &s.t. (15), (16), (17), (19), (20), (22) \end{aligned} \tag{23}$$

B PROOF OF LEMMA 4.1

Proof. We use the contradiction method to prove, i.e., there is a demand that is admitted by Algorithm 1 but the network is unable to satisfy its bandwidth-based availability. There are two 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., $d_{ik} > Capacity(T_k)$) , Algorithm 1 won't admit the demand (Line 6-8). Case (ii) is also impossible, because if the availability is smaller than its target (i.e., $s_i < \beta_{Y_i}$) , Algorithm 1 will reject the demand (Line 14-16). This completes the proof. □

C PROOF OF NP-HARDNESS

Proof. The failure recovery problem contains the all-or-nothing multi-commodity flow problem as a special case, which is known as an NP-hard problem[11]. Consider an undirected graph $G = (V, E)$ and a set of k 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. The all-or-nothing multi-commodity flow problem tries to find a maximum weight routable set:

$$\begin{aligned} &maximize \sum_{i=1}^{k} w_i \times y_i \\ &s.t. \qquad \forall e \in E : \textstyle\sum_{i=1}^{k} \sum_{p \in \mathcal{P}_i} f_{ip} L_{pe} \leq c_e \\ &\qquad\qquad \forall 1 \leq i \leq k : y_i = \begin{cases} 1 & \sum_{p \in \mathcal{P}_i} f_{pt} \geq d_i \\ 0 & \sum_{p \in \mathcal{P}_i} f_{pt} < d_i \end{cases} \end{aligned} \tag{24}$$

Where g_i denotes whether commodity flow i can be routable. We transyorm the all-or-nothing multi-commodity flow problem to a special instance of failure recovery problem and consider a special case, in which $\mu_i = 0, \forall i \in I_a$. In this case, if allocated bandwidth is larger than the demand, then the profit is 1, otherwise, it is 0. We consider setting the admitted demand from tenants and their bandwidth target in the failure recovery as the multi-commodities and their bandwidth demand in the all-or-nothing multi-commodity flow problem. If we can solve the special case of the admission control 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, 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

Algorithm 2: Greedy algorithm for failure recovery

Input: Input parameters shown in Table 2
Output: $\{f'_{ikt}\}, F$

1 Sort $i \in I_a$ in non-decreasing order with $\frac{w_i}{\sum_{k\in K} d_{ik}}$;
2 $B_e = c_e, \forall e \in E$;
3 $h_i = 0, \forall i \in I_a$;
4 $F = \{\}$;
5 **for** $i \in I_a$ **do**
6 $\quad \{f'_{ikt}\}, h_i, \{D_e\} \leftarrow$Allocation$(F, i, B)$;
7 $\quad$ **if** $h_i == 1$ **then**
8 $\quad\quad F = F \cup i$;
9 $\quad\quad B_e = D_e, \forall e \in E$;
10 $\quad$ **else**
11 $\quad\quad$ **if** $\sum_{j\in F} w_j < w_i$ **then**
12 $\quad\quad\quad D_e = B_e + \sum_{j\in J}\sum_{k\in K}\sum_{t\in T_k} f_{jkt}L_{te}, \forall e \in E$;
13 $\quad\quad\quad \{f''_{ikt}\}, h'_i, \{D'_e\} \leftarrow$Allocation$(F, i, D)$;
14 $\quad\quad\quad$ **if** $h'_i == 1$ **then**
15 $\quad\quad\quad\quad \{f'_{jtk}\} \leftarrow 0, \forall j \in F, k \in K, t \in T_k$;
16 $\quad\quad\quad\quad \{f'_{itk}\} \leftarrow \{f''_{itk}\}, \forall k \in K, t \in T_k$;
17 $\quad\quad\quad\quad F = i$;
18 $\quad\quad\quad\quad h_i = 1$;
19 $\quad\quad\quad\quad$ ***break***;
20 $\quad\quad$ **else**
21 $\quad\quad\quad$ ***break***;
22 **return** $\{f'_{ikt}\}, F$

In this part, we will introduce a greedy algorithm, shown in Algorithm 2, to solve failure recovery problem. Let F denote the set that demand set that the network supports. Firstly, it sorts all the accepted demands $i \in I_a$ in non-decreasing order according to the ratio of weight to aggregate bandwidth demands, where the aggregate bandwidth demands are derived as $\sum_{k\in K} d_{ik}$ (Line 1). The ordered sequence guarantees large profit demand with small bandwidth target have high priority. For each admitted demand, the algorithm tries to allocate resource it with remaining network capacity (Line 6). If the network is able to support it, then add to F. If the network is unable to support current demand's bandwidth-based availability and current demand is more profit , the algorithm will try to recycle total resources and test that if allocating total network resources can support current demand (Line 11-12). If this is true, then algorithm will prefer current demand (Line13-18), otherwise, the algorithm finishes the iteration (Line 21). Compared with the bruce force algorithm, Algorithm 2 can derive solution in $O(|I_a||T_k||E|)$, which is Polynomial time. However, it achieves this at the cost of performance loss.

Lemma D.1. *The approximation of Algorithm 2 is 2.*

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

$$\frac{w_1}{\sum_{k\in K} d_{1k}} \geq \frac{w_2}{\sum_{k\in K} d_{2k}} \geq \ldots. \tag{25}$$

(25) means the priority of flow pair is decided by the unit value. W.l.o.g., assume that the network can't transfer the $n+1$ demand, Algorithm 2 will choose $max\{w_{n+1}, \sum_{i=1}^{n} w_i\}$ as the value. Let OPT denote the optimal solution and it is obvious that $\sum_{i=1}^{n} w_i \leq OPT$. Also, we have $\sum_{i=1}^{n+1} w_i \geq OPT$. 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 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. $\sum_{i=1}^{n+1} w_i/2 \leq max\{\sum_{i=1}^{n} w_i, w_{n+1}\}$. Therefore, $OPT/2 \leq max\{\sum_{i=1}^{n} w_i, w_{n+1}\}$. This completes the proof. □