BATE: Boosting Bandwidth Availability 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 BATE, a novel Traffic Engineering (TE) framework that aims for *bandwidth availability* (BA) 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 BATE, 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 testbed as well as trace driven simulations across different topologies, traffic matrices and failure scenarios. Our evaluation demonstrates that, compared with the optimal admission strategy, BATE can speed up the online admission control by 30 × at the expense of a 5% higher rejection ratio. On the other hand, compared with the latest TE schemes like FFC and TEAVAR, BATE can meet the bandwidth availability SLAs for 40% more demands, and when network failure causes SLA violations, it can retain 25% more profit under a simple pricing and refunding model.

1 INTRODUCTION

Nowadays, large scale online services such as finance trading, web search, online shopping, online game and video streaming are posing stringent requirements on the availability and 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, etc., are providing various optimizations for their global WAN, especially with the help of the emerging software-defined networking techniques [10, 19, 22–25, 28, 30, 33, 34, 38].

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 [18, 19, 46]. For example, Microsoft reports links in their WAN could fail as often as every 30 minutes [38]. Once a link fails, traffic has to be rescaled and rerouted, resulting in transit or long lasting congestions. Such negative impacts on inter-DC WAN services will ultimately translate into monetary loss (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 [22, 30, 51]. Therefore, the design and optimization of inter-DC WANs have to take competitions, heterogeneities and economic interests into consideration.

In this paper, we argue that although existing traffic engineering schemes [10, 22, 24, 27, 34, 38, 49] 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 [10, 27, 34, 38, 49] typically make a conservative bandwidth allocation, so that even if a failure occurs, surviving paths could be used and the network can still be free from congestion under traffic rerouting. To prevent congestion, links, including those with negligible failure probabilities, must be kept at low utilization, resulting in significant waste of network bandwidth (and potentially less accommodated users). Such a solution may be fit for existing ISP networks which use over provision to avoid congestion, but for new players, such as content providers that are building their own backbone network (either physically [4, 19, 22, 24] or leasing bandwidth from ISPs [7, 11]), this is quite uneconomic [21]. *Second*, existing techniques mainly focus on the availability of the whole network, but ignore that users not only have diverse demands on bandwidth, but also ask for different levels of reliability. Providing reliable bandwidth can be a value-added service for many cloud providers, typically in the form of Service Level Agreements (SLAs) [4, 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 [10]) 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 also not considered by current failure recovery approaches, especially those who allocate bandwidth aggressively [10, 22], since they may run into congestions when traffic is rerouted under network failures. Such violations of SLAs will inevitably cause revenue loss, which should be kept as small as possible.

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 $d = (b_d, \beta_d, t_d^s, t_d^e)$ means that d requests bandwidth b_d for a life duration between t_d^s and t_d^e, and should be guaranteed at least β_d% of the duration, subjected to the network capacity and possible failures. Such a demand is typically represented by a Service Level Agreement (see Table 1 for real world examples), and different users or applications may pose different demands. We show that state-of-the-art traffic engineering schemes fail to meet the heterogeneous bandwidth availability demands, especially under diverse link failure probabilities that may vary by several orders of magnitude (see §2). We note that, although the general concept of bandwidth-based availability has been recognized in some recent TE works [23, 24, 33], their methodologies and evaluations are actually achieving only a soft guarantee, i.e., the ratio of the allocated bandwidth to the negotiated one, while we will provide a hard guarantee, i.e., the negotiated bandwidth must be met.

Secondly, we design BATE, a novel traffic engineering framework that aims for bandwidth availability provision over inter-DC WAN (see §3). BATE is composed of three core components, i.e., admission control, traffic scheduling and failure recovery. The admission control step strikes a balance between efficiency and optimality, so that new demands can be admitted and guaranteed as much as possible with negligible delay. Then based on a Linear Programming (LP) model, our traffic scheduling algorithm allocates bandwidth for the admitted demands over tunnels. To cope with the complexity which increases exponentially with the network size, we also propose a pruning method by ignoring certain failure scenarios that hardly happen. At last, based on a Mixed-Integer Linear Programming (MILP) model, the failure recovery procedure pre-computes backup bandwidth allocations and reroutes traffic to minimize the revenue loss due to SLA violations, which is proved to be 2-optimal.

Thirdly, we implement BATE as a real system, including a centralized controller and multiple brokers (one for each DC), together with end-host clients (see §4). We conduct extensive experiments using a small testbed as well as trace driven large scale simulations (see §5). We compare BATE with state-of-the-art WAN TE schemes such as TEAVAR [10], SMORE [34], SWAN [22], B4 [24] and FFC [38], across different topologies, traffic matrices and failure scenarios. Our evaluation demonstrates that BATE can (1) speed up the online admission control by 30 × at the expense of a rejection ratio that is 5% higher than the optimal strategy; (2) meet the bandwidth availability SLAs for 40% more demands; (3) retain 25% more profit when network failure causes SLA violations, under a simple pricing and refunding model. To our knowledge, BATE is the first to tackle bandwidth availability provision over inter-DC WAN, where heterogeneities of demands and link failures are systematically taken into account for profit maximization.

Table 1: Services have different availability targets.

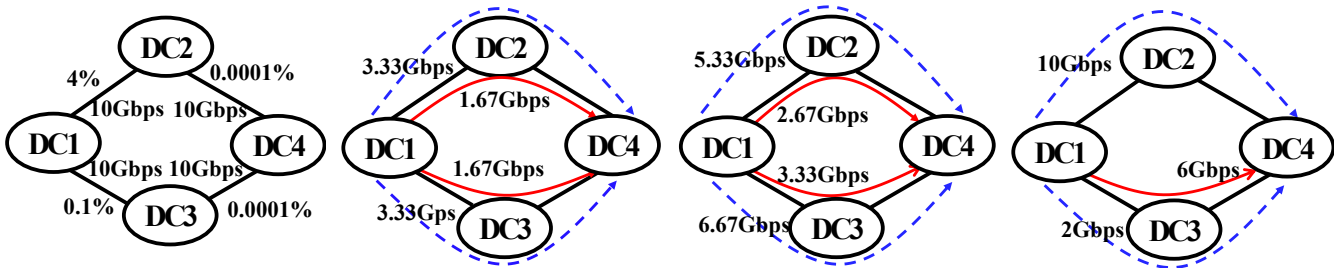
Service	Availability	Refund
Traffic Manager [8]	< 99.99%	10%
VPN Gateway [8]	< 99.95%	10%
VM Instances [4]	< 99.99%	10%
Cosmos DB [8]	< 99.999%	10%
(Azure)	< 99%	25%
DMS [6]	< 99.99%	10%
(AWS)	< 99.0%	30%
	< 95%	100%
AppFlow[5]	< 99.99%	10%
(Amazon)	< 99.95%	25%
	< 95%	100%
SMS[2]	< 95%	10%
(Alibaba)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

2 BACKGROUND AND MOTIVATION

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

2.1 Network failures and availability requirements

WAN failures are frequent and follow a heavy-tailed distribution. Failures could occur anywhere, from control plane to data plane across the network [10]. 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, 19], leading to severe performance degradation and revenue loss. On the other hand, according to the earlier measurements [18, 46], failures often follow a *heavy-tailed distribution*, where a small portion of links contribute to most of the failures, while most links experience only few failures, and the failure rate of a single link can differ by even



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

Figure 1: A simple example where user1 (red) requires 6Gbps bandwidth for at least 99% time and user2 (blue) requires 12Gbps bandwidth for at least 90% time, both from DC1 to DC4.

more than three orders of magnitude [10, 16]. Therefore, *network failures, especially their uneven distribution, should be explicitly taken into account by network operators.*

High availability directly translates into profit. Nowadays, high availability is nearly always one of the main items in SLAs [4, 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 1 shows their declared availability targets and corresponding refunding policies, where the refunding credit 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 anything below 99.0% availability [8]. As more realtime 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 service availability under network failures to retain a good profit is a big challenge.*

A one-size-fit-all network availability target is not enough. In recent years, there has been a rapid 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 [32, 39, 44]. 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 shown in Table 1, the minimal availability demands of the Data Transmission Service [1] and the Short Message Service[2] are 95% and 90%, 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 [10] only considers guaranteeing all users' bandwidth at least β% time) .

2.2 A motivating example for BATE

Now we use a simple example to illustrate why existing traffic engineering algorithms cannot meet the heterogeneous bandwidth availability demands well. The toy topology we use is depicted in Figure 1(a), where there are 4 data centers and the links connecting them are annotated with their corresponding capacities and failure probabilities. Suppose we have two bandwidth demands for inter-DC transmission from DC1 to DC4, i.e., user1 (red) requires 6Gbps bandwidth with at least 99% availability, and user2 (blue) requires 12Gbps bandwidth with at least 90% availability. There are two paths from DC1 to DC4, i.e., DC1→ DC2 → DC4, and DC1 → DC3 → DC4, whose available probabilities are $(1-4\%)\times(1-0.0001\%) = 95.999904\%$ and $(1-0.1\%)\times(1-0.0001\%) = 99.8999001\%$, respectively. We apply FFC [38] and TEAVAR [10], two latest WAN traffic engineering schemes that take network failures into account, to this scenario.

FFC [38] guarantees a total bandwidth from DC1 to DC4 under at most l concurrent node/link failures, and here we simply use $l = 1$. Figure 1(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 $(1-95.999904\%)\times(1-99.8999001\%) = 0.004004092096\%$). Since user1 and user2 can get respectively 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 does not differentiate between paths with different availabilities.* The lower path has a much smaller failure probability, and it is wasteful without utilizing it as much as possible.

On the other hand, TEAVAR [10] exploits the different link failure probabilities and maximizes the network utilization, subject to meeting a *single* desired availability. Figure 1(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 a SLA violation. This shows *TEAVAR does not consider the heterogeneous user demands on availability.* Since user1 requires a higher availability, it is better to use a path with a lower failure probability.

Our approach: Taking into account the diverse link failure probabilities and user bandwidth availability demands, Figure 1(d) shows a better bandwidth allocation, where user1 can get 6Gbps over 99.8999001% time (via the lower path that has a lower failure probability) and user2 can get 12Gbps over 95.999904% time (via both the upper and the lower path), satisfying both of their bandwidth demands.

Table 2: Key Notations for BATE

Input Variables	
$G(V, E)$	inter-DC WAN with nodes V and Links E
c_e	the remaining capacity on link $e \in E$
$k \in K$	a s(ource)-d(estination) pair in the set of all s-d pairs
T_k	the set of tunnels for a s-d pair k
D, D_a	the set of arrived demands and admitted demands[1]
$d = (\mathbf{b}_d, \beta_d)$	for a BA demand d, requiring bandwidth $\mathbf{b}_d$ with availability β_d, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, \mathbf{b}_d^2, \dots >$ of bandwidth demands over all s-d pairs[2]
t	a tunnel for transmitting traffic[3]
u_t^e	whether tunnel t passes link $e \in E$
$z \in Z$	a failure scenario in the scenario set
p_z	the probability that a failure scenario z occurs
v_t^z	whether tunnel t is available under scenario z
Output Variables	
g_d	whether demand d is admitted
f_d^t	bandwidth allocated for demand d over tunnel t
h_d	profit (after refunding) for demand d

3 BATE FRAMEWORK

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

- **High admission ratio and low admission latency:** Bandwidth availability demands might arrive at anytime. The system should be able to efficiently accommodate as many BA demands as possible under the constraint of network capacity and failure probabilities, as this would increase service agility and bring more revenue.
- **High availability for allocated bandwidth:** The system should be able to optimize its achieved availability in a probabilistic manner, as this would, in the long term, reduce potential penalties (i.e., refund due to SLA violations) and retain a good reputation. This can be achieved by making a good match between demands on higher availability and paths with low failure probability.
- **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 to normal traffic, i.e., congestion and lower bandwidth availability due to rerouting.

[1]When an admitted demand finishes, it will be removed from D_a.
[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.
[3]Multiple tunnels may exist for a single s-d pair.

3.1 Abstraction of bandwidth availability demands

In reality, a customer could be a tenant 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). However, there exists no interface for tenants to specify bandwidth availability. We now present BATE 's availability abstraction.

BA demand model: We model the inter-DC WAN as a graph $G = (V, E)$, where V and E are nodes and link sets. Let K denote the a source and destination DC pair. For a bandwidth availability demand d, the BA abstraction can be presented as $(\mathbf{b}_d, \beta_d, t_d^s, t_d^e)$, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, \mathbf{b}_d^2, \dots >$ of bandwidth demands over all s-d pairs. Network providers will offer refunding for any violating availability targets (see §2) and we use h_d to denote the profit (after refunding) for demand d.

Network model: A network scenario $z = \{z_1, z_2, \dots\}$ 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:

$$\begin{aligned} p_z &= p\left(z_1 = z_1', z_2 = z_2', z_3 = z_3', \dots, z_e = z_e'\right) \\ &= \prod_{i=1}^{|E|} (z_i' p_i + (1 - z_i')(1 - p_i)) \end{aligned} \quad (1)$$

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 $p(\{1,1,0\}) = 0.9 \times 0.8 \times 0.15 = 0.108$.

We consider tunnel-based forwarding [10, 22, 38], where traffic is carried over a set of tunnels. For each node pair $k \in K$ of the inter-DC WAN, 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 [48], oblivious routing [34]). Each tunnel $t \in T_k$ contains a set of links and u_t^e presents whether tunnel t contains link $e \in E$ or not. f_d^t is the bandwidth allocated over tunnel t for demand d and D_a is the total *admitted* demand set. Let v_t^z denote whether tunnel t is available (i.e., $v_t^z = 1$) or not (i.e., $v_t^z = 0$) under network scenario z.

3.2 Admission control

Users' demands arrive in the first-come-first-service (FCFS) manner and *no preemption* is allowed. The new arrival demand, together with the admitted ones (i.e., D_a), constitute demand set D. If every element in D 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 (MILP) 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 a NP-hard problem by reducing the all-or-nothing multi-commodity flow problem [12] to a special case of it. We omit the details for brevity.

In reality, 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 d 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 d. 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 smallest bandwidth times 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 availability targets in D are satisfied (Line 19). Algorithm 1 is able to derive allocation results in $O(D * K * max(|T_k|))$. Algorithm 1 can derive a tight solution:

Algorithm 1: Rescheduling algorithm

Input: Input parameters shown in Table 2
Output: $\{g_1, g_2, .., g_d, ...\}, \{f_d^t\}$

1 $g_d = 0, s_d = 1, \forall d \in D$;
2 $f_d^t = 0, \forall d \in D, k \in K : t \in T_k$;
3 **while** *true* **do**
4 $d = \arg_{d' \in D} min\{\sum_{k \in K} b_{d'}^k \times \beta_{d'}\}$;
5 **for** $k \in K$ **do**
6 **if** $b_d^k > Capacity(T_k)$ **then**
7 $g_d = 0$;
8 **return** $\{g_1, g_2, .., g_d, ...\}, \{f_d^t\}, False$;
9 **while** $b_d^k > 0$ **do**
10 $t = \arg_{t \in T_k} \min\{c_t * p_t\}$;
11 $f_d^t = Allocation(t, b_d^k)$;
12 $s_d = s_d * p_t$;
13 $b_d^k = b_d^k - f_d^t$;
14 **if** $s_d < \beta_d$ **then**
15 $g_d = 0$;
16 **return** $\{g_1, g_2, .., g_d, ...\}, \{f_d^t\}, False$;
17 $g_d = 1$;
18 $D = D \setminus d$;
19 **return** $\{g_1, g_2, .., g_d, ...\}, \{f_d^t\}, True$;

Lemma 3.1. *There must exist an allocation scheme to satisfy the bandwidth availability targets of all demands admitted by Algorithm 1.*

The proof details can be found in Appendix B

3.3 Traffic scheduling

For each *admitted* demand, the traffic scheduling determines bandwidth allocation over each tunnel every T (e.g., 10 mins).

Traffic scheduling aims to satisfy each BA demand's availability target and we model the traffic scheduling as a linear programming. Firstly, the bandwidth allocation results f_d^t for BA demand d over tunnel $t \in T_k$ should be non-negative, i.e.,

$$f_d^t \geq 0, \quad \forall d \in D, k \in K, t \in T_k. \tag{2}$$

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

$$\sum_{d \in D} \sum_{k \in K: t \in T_k} f_d^t u_t^e \leq c_e, \quad \forall e \in E. \tag{3}$$

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

$$\sum_{t \in T_k} f_d^t \leq \mathbf{b}_d^k, \quad \forall d \in D_a, k \in K \tag{4}$$

R_{dk}^z is the ratio of total reserved bandwidth to demand through pair k for BA demand d under network scenario z and it is defined as:

$$R_{dk}^z = \frac{\sum_{t \in T_k} f_d^t v_t^z}{\mathbf{b}_d^k}, \quad \forall d \in D, z \in Z, k \in K. \tag{5}$$

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

$$B_d^z \leq R_{dk}^z, \quad \forall d \in D_a, z \in Z, k \in K \tag{6}$$

It is obvious $B_d^z \leq 1$ and $B_d^z = 1$ presents scenario z is qualified, therefore, we can define $B_d^z \times p_z$ 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.,

$$\sum_{z \in Z} B_d^z \times p_z \geq \beta_d, \quad \forall d \in D_a \tag{7}$$

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

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

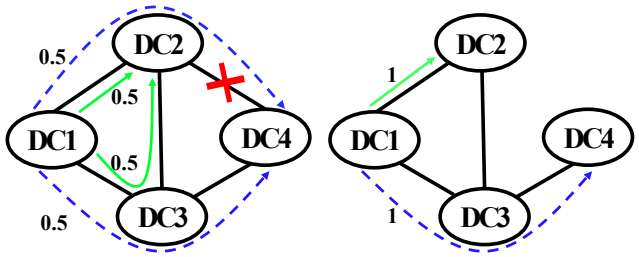
We can see that although the traffic engineering module is LP problem, it considers each network scenario, which makes problem complexity increase exponentially with network size. For instance, B4 [24] topology has 12 nodes and 38 links, then there are $2^{38} - 1$ network failure scenarios in total. 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.

3.4 Failure recovery

In BATE, 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). BATE 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 2 shows two services are from DC1 to DC4 and both of them need bandwidth demand of 2. Link capacity is 1 everywhere. Figure 2(a) shows the original traffic engineering allocation and it is used when no failures happen. Figure 2(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 → DC4 failure is detected. Network scenario number is huge for large-scale network and we assume that at any time, only one link in the network could fail. If we assume network scenario z could happen, and let T_k^z denote the surviving tunnel set under scenario z and the bandwidth allocated for demand d over tunnel t should be non-negative, i.e.,

$$f_d^t \geq 0, \quad \forall d \in D, k \in K : t \in T_k^z. \tag{9}$$



(a) Original TE allocation (b) Backup allocation

Figure 2: 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.

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

$$\sum_{d \in D_a} \sum_{k \in K: t \in T_k^z} f_d^t u_t^e \le c_e \times A_e^z \tag{10}$$

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

$$R_{dk} = \frac{\sum_{t \in T_k^z} f_d^t}{\mathbf{b}_d^k}, \forall d \in D_a, k \in K \tag{11}$$

If the allocated bandwidth of BA demand d between each node pair is larger than its need (i.e., $R_{dk} \ge 1, \forall d \in D_a$), the profit unit is 1, otherwise, there is a refunding, which can be derived from SLA and we use μ_d to denote its profit fraction after refunding. Then, the profit of d is:

$$\forall d \in D_a : h_d = \begin{cases} 1 & \forall k \in K : R_{dk} \ge 1 \\ \mu_d & \text{Otherwise} \end{cases} \tag{12}$$

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:

$$\begin{cases} h_d = y_d + \mu_d * (1 - y_d), & \forall d \in D_a. \\ R_{dk} < M * y_d + 1 - y_d, & \forall d \in D_a, k \in K. \\ R_{dk} >= y_d, & \forall d \in D_a, k \in K. \\ y_d \in \{0, 1\}, & \forall d \in D_a. \end{cases} \tag{13}$$

Where M is a big integer that is at least larger than the upper bound of R_{dk}. Let w_d denote the profit of demand d 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:

$$\begin{aligned} &maximize \sum_{d \in D_a} w_d \times h_d \\ &s.t. (9), (10), (11), (13) \end{aligned} \tag{14}$$

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

Lemma 3.2. *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.*

4 SYSTEM IMPLEMENTATION

We have implemented BATE on the Linux platform. Figure 3 shows the whole system architecture, which contains one controller, multiple brokers (one for each DC) and multiple clients (one for each host). The controller is responsible for most decision work of BATE, including admission control, traffic scheduling, and failure recovery, while the brokers and clients are responsible for bandwidth enforcement. It works as follows: When a user submits a demand to the controller, the admission control module determines whether the demand can be admitted or not (see § 3.2). If the demand is admitted, this module will also allocate its demanded bandwidth on appropriate paths for the first time, and notify the brokers for enforcement. The online scheduler module performs traffic scheduling (see § 3.3) periodically (e.g., every 10 minutes) to further optimize the availability expectation of all active demands. In addition, for potential link failures, it also pre-computes backup allocation strategies that will be activated if any link failure indeed happens (see § 3.4). These central decisions are distributed to the brokers for bandwidth enforcement. The brokers in each DC monitor link status and bandwidth consumption, report these statistics to the central controller, and ask the clients to implement appropriate rate limit on end hosts.

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$k \in K$),

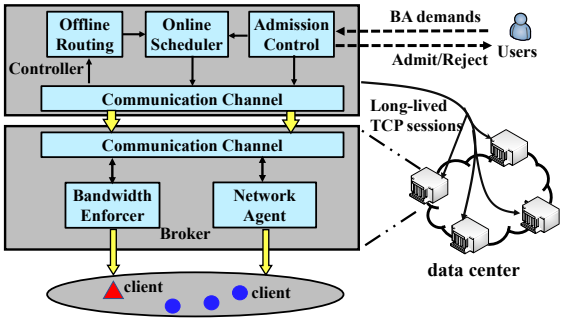


Figure 3: System architecture of BATE.

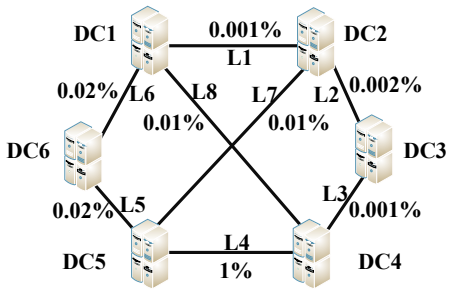


Figure 4: Testbed topology. Each link has the same capacity 1Gbps but different failure probability.

using certain routing algorithms (oblivious routing [34], k-shortest path [22], etc.). These tunnels are used by the admission control module and the online scheduler module as input variables; (2) Admission Control. When a BA demand is submitted, this module uses the admission control algorithm (see § 3.2) to reject it, or accept it and allocate bandwith over the tunnels in nearly real-time. The results are sent to the corresponding brokers. (3) Online Scheduler. Periodically, This module performs traffic scheduling (see § 3.3) according to the bandwidth availability demands submitted by users, so that the availability can be optimized in an expected manner. It also pre-computes backup allocation (see § 3.4) for some potential link failures. For each user demand, the normal bandwidth and backup bandwidth allocated over each tunnel (i.e., f_i^t) are then sent to the corresponding brokers. In addition, our system also supports several other TE algorithms including SWAN [22], FFC [38] and TEAVAR [10]. (4) Communication Channel. This module is responsible for communication with brokers, where we use long-lived TCP connections to avoid unnecessary delay. In addition, controller failures can be remedied by using multiple replications, where the master controller is elected by the Paxos [35] algorithm.

Broker takes care of the data center it resides in. It consists of three modules: (1) Bandwidth Enforcer. It receives the bandwidth allocation results (i.e., f_i^t) from controller, sends them to the corresponding 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 [14]), and uses the OpenFlow [42] protocol to installs and updates forwarding rules on the switches in the same 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, which can be further expanded if necessary. In this way, a flow (i.e., traffic corresponding to a BA demand) is marked with a label at the ingress switch, and the succeeding switches use this label for forwarding. Group tables in the switch pipelines are used for flow splitting (i.e., traffic corresponding to a BA demand can be split into multiple sub-flows and transmitted in multiple tunnels). Besides, the network agent also tracks the network topology, reports any change or failure to the central Controller module, and monitors the actual traffic rate. (3) Communication Channel. This component is responsible for communication with the central Controller.

Client sits on each host to carry out rate limiting. It consists of a daemon and a kernel module. The former recieves bandwidth allocation results from the site broker, while the latter sits between the TCP/IP stack and the Linux Traffic Control (TC) module to control outbound traffic rate.

5 EVALUATION

In this section, we evaluate the performance of BATE in both real trace driven simulation and testbed. Our main results are as follows.

(1) The admission control of BATE can achieve a reasonable tradeoff. BATE can speed up the online admission control by 30 × at the expense of 5% higher rejection ratio.

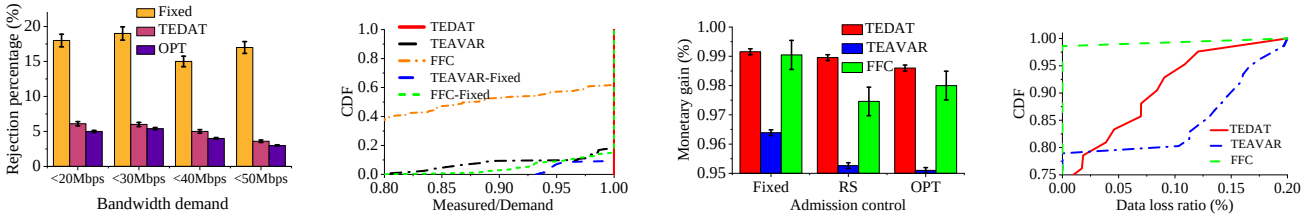
(2) BATE has consistent performance under various topologies and traffic. Compared with the state-of-the-art traffic engineering algorithms, BATE is able to make 40% more bandwidth demands meet availability SLAs.

(3) BATE can retain 30% more profit when network fails.

(4) BATE is fast and accuracy. The average error of traffic scheduling is within 8% and the profit loss of greedy algorithm is less than 10%.

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



(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.

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 [31]. 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. The network agent in broker runs Floodlight [14] as the OpenFlow controller. In each data center, all hosts and NICs connect to an Open vSwitch [43]. 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 threshold 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 [38] and TEAVAR [10] to make comparisons.

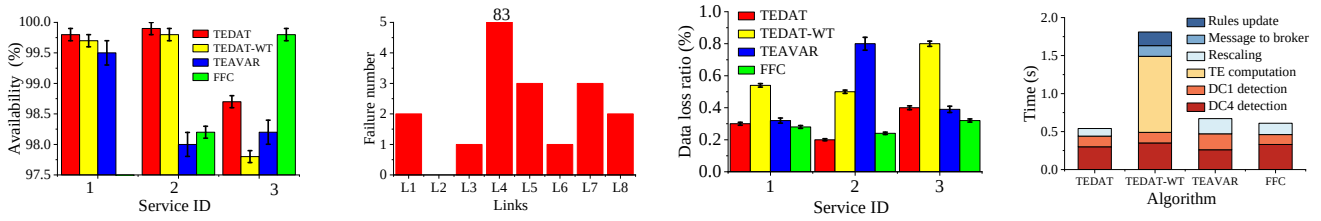
5.1.2 A public cloud example. We firstly evaluate a public cloud case. Assume tenants arrive at the public cloud under poisson process with mean arrival rate 2 every minute and the duration of each bandwidth demand is under exponential distribution with a mean of 5 minutes. Bandwidth demands between each node pair are under uniform distribution between 10Mbps and 50Mbps. Every 1Mbps is sold 1$. Availability targets and refunding ratio are randomly chosen from Azure SLAs [8]. Each experiment lasts 100 minutes and is repeated 20 times, where the error bars paints the maximal, minimal and mean value.

Figure 5(a) demonstrates BATE only performs slightly worse than the optimal solution (about 4%) and is at least 2 × better than the *Fixed* admission control. This is because BATE 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, BATE 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 BATE, because FFC is too conservative and and TEAVAR ignores the diverse demands. Figure 5(c) shows BATE can retain about 15% more money ratio than FFC and TEAVAR when network fails. This is because BATE pre-computes backup allocation with maximal profit and they can be put into practice as far as failures are detected. We measure total bytes loss according to *iperf* server side reports and packet counters in the switches, then finally derive the data loss ratio. Figure 5(d) demonstrates data loss ratio after network fails, where BATE performs about 8% worse than FFC. FFC performs best since it keeps network utilization low and almost no congestion occurs when network fails, while BATE focuses on Monetary gain rather than data loss.

Table 3: Scheduled results of different schemes.

Service	paths	BATE	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

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



(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.

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. BATE-WT refers BATE 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 BATE, 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) BATE 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 BATE and FFC performs better than BATE-WT and TEAVAR for data losses ratio. Data losses of Service-2 under TEAVAR is high due to Link L4 failings. BATE-WT performs worse than the other three schemes, because it takes much time to derive new resource allocation when network fails and the events time of BATE-WT shown in Figure 6(d) just prove this.

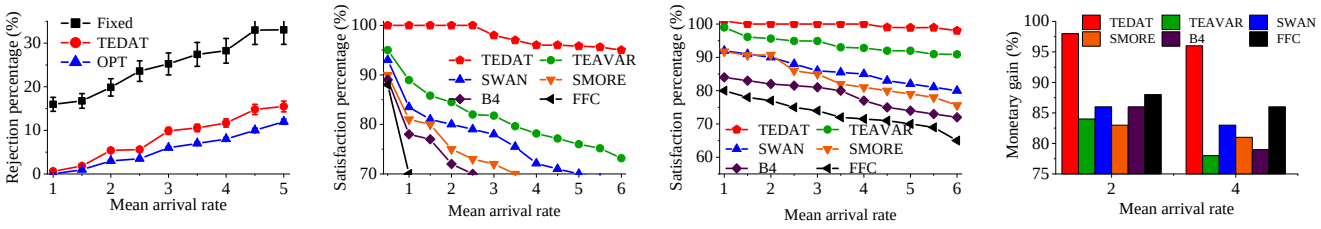
5.2 Simulation evaluation

Simulation setup. We evaluate the performance of BATE under four network topologies: B4 [24], ATT [10], IBM[34], CERNET2 (China Education and Research Network), where

Table 4: Network topologies used in the simulations

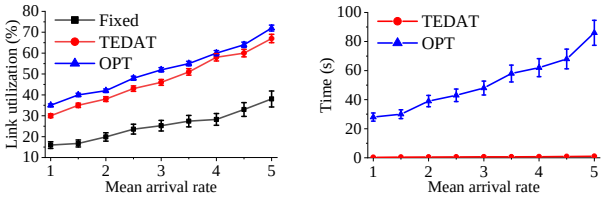
Topology Name	#Nodes	#Links
IBM	18	48
B4	12	38
ATT	25	112
CERNET2	14	32

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 backbones [40] 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 BATE against FFC [38], TEAVAR [10], SWAN [22], SMORE [34], B4 [24]. We change them to adapt service level traffic engineering: FFC tries to maximize the link utilization with considering 1 fault could occur in our evaluation. TEAVAR maximizes bandwidth allocation to each demand subject to a single operator-specified availability target. The default availability target of TEAVAR is 99.9% in our simulations. SMORE [34] minimizes the maximum link utilization without explicit guarantees on availability. SWAN [22] tries to maximize the total throughput of all demands in the current slot. B4 [24] aims to deliver max-min fair allocation manner to each demand. We consider 10 availability targets, whose value are from the SLAs of Azure [8]. Our collected traffic data contains the capacity of all links. We generate tenants' bandwidth demands with a



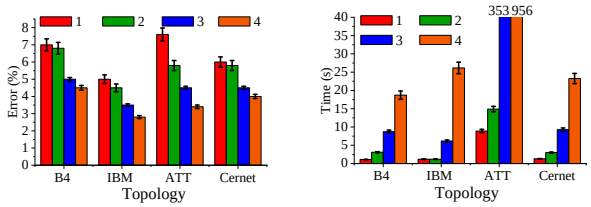
(a) IBM (b) B4 (c) ATT (d) Cernet

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



(a) Link utilization (b) Admission time

Figure 8: Scalability of admission control.



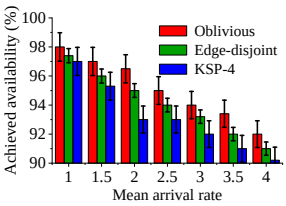
(a) Cutoff error (b) Optimization time

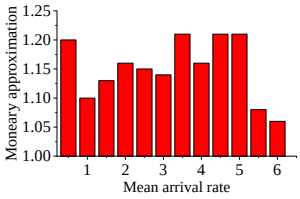
Figure 9: Impact of pruning for traffic scheduling.

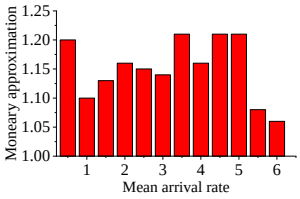
Poisson process, the duration of each demand is modeled as an exponential distribution with a mean of 1000 time slots.

5.2.1 Performance under real network topologies. Figure 7(a) shows BATE rejects only about 3% more demands than the optimal solution and can accept about 50% more demands than the *Fixed* consistently. Similar to [10], we next start a post-processing simulation, in which intends to send entirely bandwidth under each scenario at each time slot. The sum of scenario probability where demand is fully satisfied reflects the *achieved availability*. The term *satisfaction percentage* refers to the fraction of total arrival demands whose achieved availability are higher than their availability targets throughout duration. Figure 7(b) demonstrates that BATE can make 40% more demands satisfy SLA availability. BATE can make demands with stringent availability requirements pass links with high reliability, therefore, their availability targets can be guaranteed, but other schemes are unable differentiate demands. To show this benefit, we adopt *Fixed* admission control for each traffic scheduling algorithm, then Figure 7(c) shows BATE performs at least 10% better than other algorithms under the *Fixed* admission control method. Figure 7(d) shows the average monetary gain percentage for the scenario in which one failure could occur in the network. We can see BATE is able to retain 30% more profit than other algorithms when network fails. We qualify the scalability by measuring the admission control time for each demand. The simulation is performed on a Linux server (4-core, 2.60GHz processor with 32GB). As shown in Figure 8, the admission control time of BATE is less than 1s with about less than 5% error, which is at least 30× faster the global optimal solution under ATT topology. Algorithm **??** cuts off the scenarios that can hardly happen. Let BATE denote the overall achieved availability of all admitted demands with scenario pruning and *OPT* denote the achieved availability of optimal solution without scenario pruning. Then the error can be defined as $\frac{|\text{BATE}-OPT|}{|OPT|}$. Figure 9 (a) shows the error of pruning algorithm with different current link failure numbers. We can see the error is less than 8% even the current link failure number is 1. Traditionally, TE should update the network state every 5-10 minutes. Figure 9 (b) shows the time when solving it with Gurobi [20] on the server (4-core, 2.60GHz processor with 32GB). We can see that even on a large network (e.g., ATT topology), at most 15 seconds are needed when current link failure numbers are 2, which is fast enough in reality.

So far, we use K-shortest path as the default tunnel selection scheme. There are also other tunnel selection schemes, e.g., edge disjoint paths[48], oblivious routing[34]. Figure 10 shows the comparison with different tunnel selection methods. We can see that (1) BATE performs well regardless the tunnel selection algorithm. This indicates that the success of BATE doesn't reply on the particular routing scheme; (2) Oblivious routing is still slighter better than KSP and edge disjoint paths. This is because it is intended to avoid link







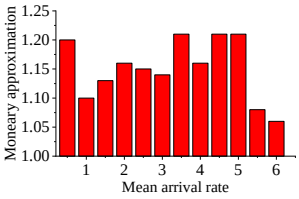


Figure 10: Routings. **Figure 11: Approximation.** **Figure 12: Approximation.** **Figure 13: Approximation.**

over-utilization through diverse and low-stretch path selection.

We propose a greedy for failure recovery optimization problem. Figure 13 shows the approximation of the greedy algorithm, where the approximation can be defined as the profit ratio of optimal solution and greedy solution. We can see that the profit loss of greedy algorithm is less than 10%.

6 RELATED WORK

Traffic engineering for WAN. Optimizing WAN performance is a big challenge. Prior technologies such as OSPF[15] and MPLS tunnels [13, 29] can improve the backbone traffic transfer under the knowledge of traffic demands. Recently, SDN is widely used in WAN TE. SWAN[22], B4[23, 24], Bwe[33],OWAN[28] leverage the whole network information to allocate network bandwidth. Network scheduling schemes [25, 30, 50] 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.

Network risk based traffic engineering. [19] studies the network failures and proposes the principles to design robust network, but it doesn't contain any TE framework. [17] suggests that backbone traffic engineering strategies should consider current and past optical layer performance. Neither [19] nor [17] contains TE frameworks. [41] 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 [45],[49]. The reactive methods could take a long time to recover from faults, while applications have already been hurt[38]. FFC [38] and TEAVAR[10] are proactive routing protection methods, but they can't ensure the resources of services under network failures. BATE makes a further step and tries to provide service level bandwidth-based availability. The availability of BATE 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^{2}TCP[47], LPD [52], Amoeba[51] 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 [36] and EyeQ [26], can provide bandwidth guarantees to applications or tenants, while they ignore the network risks and service availability requirements. BATE is different from in two aspects. Firstly, BATE considers the network failures and tries to perform rate limit under the assumption. Secondly, BATE can guarantee bandwidth at a particular level (e.g., 99%).

7 CONCLUSION REMARKS

We present BATE, a framework that attempts to guarantee the availability demands of applications. BATE 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 BATE in real testbed as well as trace-driven simulations. The evaluation results demonstrate that BATE 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] Amazon. 2019. Amazon Compute Service Level Agreement (2019). https://aws.amazon.com/compute/sla/. (2019).
[5] Amazon. 2020. Amazon AppFlow Service Level Agreement. https://aws.amazon.com/cn/appflow/sla/. (2020).
[6] Amazon. 2020. AWS Database Migration Service (AWS DMS) Service Level Agreement. https://aws.amazon.com/cn/dms/sla/. (2020).
[7] aryaka. 2020. Aryaka Private WAN. https://www.aryaka.com. (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] cato. 2020. Cato Managed Services. https://www.catonetworks.com. (2020).

[12] 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.

[13] 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.

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

[15] 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.

[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] 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

[18] 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

[19] 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)*.

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

[21] 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

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

[23] 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

[24] 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

[25] 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

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

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

[32] 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

[33] 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

[34] 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

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

[36] 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

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

[38] 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

[39] 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

[40] 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.

[41] 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

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

[43] 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

[44] 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.

[45] 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

[46] 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

[47] 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

[48] 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*.

[49] 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

[50] 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

[51] 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

[52] 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_d^t for BA demand d over tunnel t should be non-negative, i.e.,

$$f_d^t \geq 0, \quad \forall d \in D, k \in K : t \in T_k. \tag{15}$$

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

$$\sum_{d \in D} \sum_{k \in K : t \in T_k} f_d^t u_t^e \leq c_e, \quad \forall e \in E. \tag{16}$$

R_{dk}^z is the ratio of total reserved bandwidth to demand through pair k for BA demand d under network scenario z and it is defined as:

$$R_{dk}^z = \frac{\sum_{t \in T_k} f_d^t v_t^z}{\mathbf{b}_d^k}, \quad \forall d \in D, z \in \boldsymbol{Z}, k \in K. \tag{17}$$

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

$$A_d^z = \begin{cases} 1 & \forall k \in K : R_{dk}^z \geq 1 \\ 0 & \text{Otherwise} \end{cases}, \forall d \in D, z \in \boldsymbol{Z}. \tag{18}$$

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

$$\begin{cases} R_{dk}^z < 1 - A_d^z + M \times A_d^z, & \forall d \in D, z \in Z, k \in K. \\ R_{dk}^z >= A_d^z, & \forall d \in D, z \in Z, k \in K. \\ A_d^z \in \{0, 1\}, & \forall d \in D, z \in Z. \end{cases} \tag{19}$$

where M is a big integer that is at least larger than the upper bound of $R_{dk}^z, \forall d \in D, k \in K, z \in Z$. It is easily to prove (19) equals to (18). The achieved bandwidth availability of demand d is the sum of all safe network scenarios' probability:

$$S_d = \sum_{z \in Z} A_d^z \times p_z, \quad \forall d \in D. \tag{20}$$

We use g_d to present whether the availability target of d can be satisfied:

$$g_d = \begin{cases} 1 & 1 > S_d \geq \beta_d \\ 0 & \beta_d > S_d \geq 0 \end{cases}, \quad \forall d \in D. \tag{21}$$

If the achieved bandwidth availability (i.e., S_d) is larger than its desired availability target (i.e., β_d), the demand can be admitted. Otherwise, it is rejected, which means the network can't support d. Also, (21) can be changed into the following linear format:

$$\begin{cases} S_d < \beta_d \times (1 - g_d) + g_d, & \forall d \in D. \\ S_d \geq \beta_d \times g_d, & \forall d \in D. \\ g_d \in \{0, 1\}, & \forall d \in D. \end{cases} \tag{22}$$

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

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

B PROOF OF LEMMA 3.1

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 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., $\mathbf{b}_d^k > Capacity(T_k)$) , 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., $s_d < \beta_d$) , 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[12]. 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:

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

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 $\mu_i = 0, \forall d \in D_a$. 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 $\frac{w_i}{\sum_{k\in K} \mathbf{b}_d^k}$;
2 $B_e = c_e, \forall e \in E$;
3 $h_i = 0, \forall d \in D_a$;
4 $F = \{\}$;
5 **for** $d \in D_a$ **do**
6 $\{f'_{ikt}\}, h_i, \{D_e\} \leftarrow$Allocation$(F, i, B)$;
7 **if** $h_i == 1$ **then**
8 $F = F \cup i$;
9 $B_e = D_e, \forall e \in E$;
10 **else**
11 **if** $\sum_{j\in F} w_j < w_i$ **then**
12 $D_e = B_e + \sum_{j\in J} \sum_{k\in K} \sum_{t\in T_k} f_{jkt} u_t^e, \forall e \in E$;
13 $\{f''_{ikt}\}, h'_i, \{D'_e\} \leftarrow$Allocation$(F, i, D)$;
14 **if** $h'_i == 1$ **then**
15 $\{f'_{jtk}\} \leftarrow 0, \forall j \in F, k \in K, t \in T_k$;
16 $\{f'_{itk}\} \leftarrow \{f''_{itk}\}, \forall k \in K, t \in T_k$;
17 $F = i$;
18 $h_i = 1$;
19 ***break***;
20 **else**
21 ***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 $\sum_{k\in K} \mathbf{b}_d^k$ (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:

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

(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. □