BATE: Boosting Bandwidth Availability Over Inter-DC WAN

Paper # 56, 18 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 only a xx% false rejections. 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, 37].

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 [37]. 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, 37, 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, 37, 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., a high ratio of the allocated bandwidth to the negotiated one, while we will provide a hard guarantee, i.e., the negotiated bandwidth must be met.

Secondly, we design BATE, a novel traffic engineering framework that aims for bandwidth availability provision over inter-DC WAN (see §3). BATE is composed of three core components, i.e., admission control, traffic scheduling and failure recovery. The admission control 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 [37], 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 false rejection ratio that is only around xx%; (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.

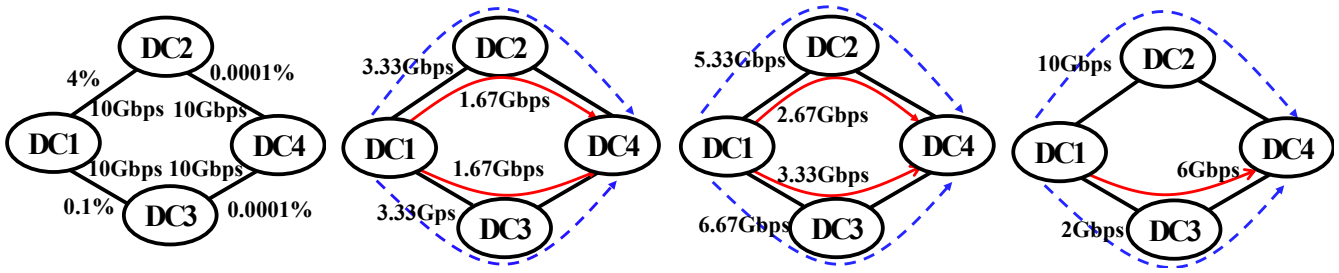
Service	Availability	Refund
Traffic Manager [8]	< 99.99%	10%
VPN Gateway [8]	< 99.95%	10%
VM Instances [4]	< 99.99%	10%
Cosmos DB [8]	< 99.999%	10%
(Azure)	< 99%	25%
DMS [6]	< 99.99%	10%
(AWS)	< 99.0%	30%
	< 95%	100%
AppFlow[5]	< 99.99%	10%
(Amazon)	< 99.95%	25%
	< 95%	100%
SMS[2]	< 95%	10%
(Alibaba)	< 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 more than three orders of magnitude [10, 16]. Therefore,



(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.

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 [37] and TEAVAR [10], two latest WAN traffic engineering schemes that take network failures into account, to this scenario.

FFC [37] 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
$k \in K$	a s(ource)-d(est) pair in the set of all s-d pairs
T_k	the set of tunnels for a s-d pair k
$d=(\mathbf{b}_d, \beta_d)$	a BA demand d, requiring bandwidth $\mathbf{b}_d$ with availability β_d, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, \mathbf{b}_d^2, ... >$ of bandwidth demands over all s-d pairs
$D, \hat{D}$	the set of arrived and admitted demands[1]
t	a tunnel for transmitting traffic[2]
u_t^e	whether tunnel t passes link $e \in E$
c_e, c_t	the remaining capacity on link e or a tunnel t
$\mathbf{z} \in Z$	a network failure scenario in the scenario set
$p_\mathbf{z}$	the probability that a failure scenario $\mathbf{z}$ occurs
$v_t^\mathbf{z}$	whether tunnel t is available under scenario $\mathbf{z}$
$w_e^\mathbf{z}$	whether link e is available under scenario $\mathbf{z}$
Output Variables	
g_d	whether demand d is admitted
f_d^t	bandwidth allocated for demand d over tunnel t
r_d	profit (after refunding) for demand d

3 BATE FRAMEWORK

In this section, we discuss the details of BATE, which contains three parts, i.e., admission control, traffic scheduling and failure recovery, 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 lower 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 contention from the rerouted traffic.

3.1 Abstraction of bandwidth availability

In reality, a customer demanding inter-DC WAN bandwidth resources could be any application or tenant spanning multiple data centers, in either a public or a private cloud. Our abstractions on network failure scenarios and bandwidth availability demands in BATE are as follows.

Network failure scenario model: The inter-DC WAN is modeled as a directed graph $G(V, E)$, where the set of nodes V represent the data centers, and the set of links E represent directed links between them. A network scenario $\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{|E|}\}$ is a vector of link states, where each element $\mathbf{z}_i \in \{0, 1\}$ denotes whether the i-th link is up ($\mathbf{z}_i = 1$) or down ($\mathbf{z}_i = 0$). We assume network operators can use historical data to estimate the failure probability x_i for this link, which are statistically indenpendent. Let Z denote the network scenario set, then the expected probability that a network scenario $\mathbf{z} \in Z$ will happen is given by [10]

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

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

BA demand model: Let K denote the set of all source-destination (s-d) DC pairs. A bandwidth availability demand d is in the form of $(\mathbf{b}_d, \beta_d)$, where $\mathbf{b}_d$ is a vector $< \mathbf{b}_d^1, ..., \mathbf{b}_d^k, ... >$ of bandwidth demands on each s-d pair $k \in K$ [3].

Traffic engineering model: Our Traffic Engineering uses tunnel-based forwarding [10, 22, 37]. For each source-destination node pair $k \in K$ of the inter-DC WAN, we pre-compute a set

[1] When an admitted demand finishes, it will be removed from $\hat{D}$.
[2] Multiple tunnels may exist for a single s-d pair.
[3] Here we omit the start and end time of this demand, but they will be implicitly considered in our online admission and traffic scheduling.

of tunnels T_k with different routing schemes (e.g., k-shortest paths, edge disjoint paths [48], oblivious routing [34], etc.). Each tunnel $t \in T_k$ contains a sequence of links and u_t^e denotes whether tunnel t passes a specific link $e \in E$ or not. We use D and $\hat{D}$ to represent *arrived* demands and *admitted* demands, respectively. Given a new demand d, our admission control scheme will decide whether to admit it. The bandwidth that will be allocated for an admitted demand $d \in \hat{D}$ over tunnel t is denoted by f_d^t.

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

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

.

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

3.2 Admission control

User demands are served in a first-come-first-service (FCFS) manner without preemption. When a new demand d arrives, we have $D = \hat{D} \cup d$. To accommodate as many demands as possible, the optimal admission strategy would be that, if every demand in D can meet its availability target, then d should be admitted. Otherwise, it should be rejected. This can be modeled by a 0-1 Mixed-Integer Linear Programming (MILP) problem which maximizes the number of demands whose availability targets can be satisfied. Appendix A shows the formulation of this problem, which can be proved to be NP-hard by reducing the all-or-nothing multi-commodity flow problem [12] to a special case of it.

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

(1) When a new demand d arrives, we *fix* the bandwidth allocation for all admitted demands in $\hat{D}$, then we check whether d can be satisfied by the remaining network capacity and failure probability. If the answer is positive, then admit d and make a first-time bandwidth allocation for it.
(2) Otherwise, run a greedy algorithm (Algorithm 1) to *conjecture* whether the admitted demands can potentially be rescheduled to accommodate d. If the answer is positive, then admit d and make a temporary bandwidth allocation for it, using the remaining network capacity as far as needed [4].
(3) If d still cannot be accommodated, reject the demand.

The greedy algorithm tries to conjecture, in an efficient way, whether an allocation strategy satisfying all demands (i.e., including d) exists. It works iteratively as follows. In each iteration, It finds the demand which has the smallest product of bandwidth target and availability target (i.e., $\sum_{k \in K} \mathbf{b}_d^k \times \beta_d$) at first (line 2), and tries to allocate bandwidth for each of its s-d pairs one by one. If the remaining network capacity cannot

Algorithm 1: Admission Conjecture

Input: Input parameters shown in Table 2
Output: Whether the new demand can be admitted.

```
1  while true do
2      d = arg_{d' ∈ D} min{ Σ_{k∈K} b_{d'}^k × β_{d'} };
3      for k ∈ K do
4          if b_d^k > remaining capacity of s-d pair k then
5              return False;
6          T'_k = T_k;
7          while b_d^k > 0 do
8              t = arg_{t ∈ T'_k} min{c_t * p_t};
9              f_d^t = min{c_t, b_d^k};
10             T'_k = T'_k \ t;
11             s_d = s_d * p_t;
12             b_d^k = b_d^k − f_d^t;
13             update the remaining capacities of links
                 and tunnels;
14     if s_d < β_d then
15         return False;
16     D = D \ d;
17 return True;
```

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

satisfy this demand, we will give up (line 4-5). Otherwise, it allocates tunnel bandwidth for this demand, where a tunnel with a smaller product of remaining capacity and availability has a higher priority (line 7-13). After this, if the availability target cannot be roughly satisfied, we will give up (line 14-15), otherwise, we go for the next iteration.

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

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

3.3 Traffic scheduling

For *admitted* demands (including the newly admitted ones), we carry out traffic scheduling to further optimize the bandwidth allocation periodically (e.g., every 10 minutes). We model this as a linear programming problem, which maximizes the overall availability to be achieved (in a probabilistic sense), under the constraint that all bandwidth target must be met. Specifically,

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

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

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

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

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

To meet the availability target, we should guarantee the total probability of the qualified scenarios is no less than the availability target, i.e., $\sum_{R_{dk}^{\mathbf{z}} \geq 1} p_{\mathbf{z}} \geq \beta_d$. However, this condition will result in an mixed interger linear programming problem, and we choose to relax it and sovle the following linear programming problem.

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

$$B_d^{\mathbf{z}} \leq R_{dk}^{\mathbf{z}}, \quad \forall d \in \hat{D}, k \in K, \mathbf{z} \in Z \tag{3}$$

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

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

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

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

and

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

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

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

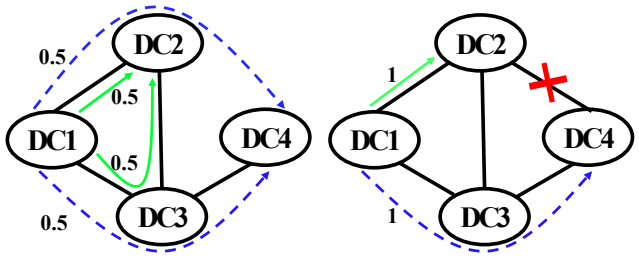
Solving this LP problem directly is possible, but as it considers every possible network scenario, the complexity will increase exponentially with the network size. For instance, the B4 topology [24] has 12 nodes and 38 links, so there are totally 2^{38} network failure scenarios (when only link failures considered). Therefore, an important question is *how to effectively reduce the problem size without affecting the result significantly*. TEAVAR [10] prunes a scenario if its probability is smaller than a threshold. However, such a threshold is difficult to choose, and an enumeration of all possible scenarios is still needed. Instead, we use a much faster pruning method, where at most two concurrent link failures will be considered, and all the remaining scenarios will be aggregated into one special *unqualified* scenario. In this way, the set of scenarios Z and the corresponding probabilities $\{p_{\mathbf{z}}\}$ can be efficiently computed.

3.4 Failure recovery

When failures occur and any tunnel becomes unavailable, traffic can be redistributed across the surviving tunnels. To reduce recovery time, BATE proactively computes backup allocation strategies for potential failure scenarios, so that the surviving tunnels can be used immediately, and packet loss can be mitigated [5].

For example, in Figure 2, there are two users, and the link capacity is 1 everywhere. One user requests a bandwidth of 1 from DC1 to DC2, while the other one requests a bandwidth of 1 from DC1 to DC4. Figure 2(a) shows the original bandwidth allocation when no failures occur, and Figure 2(b) depicts the backup allocation pre-computed for a failure of link DC2→DC4.

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



(a) Original allocation (b) Backup allocation

Figure 2: Bandwidth allocation for DC2→DC4 failure

However, service providers have to refund credits to users if their bandwidth availability targets according the agreed SLAs are violated (see §2), both for users affected *directly* by the failure and for users *collaterally* affected by the recovery procedure. Our failure recovery scheme takes this economic factor into account in the way as follows.

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

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

If for every k, R_{dk} is larger than its demand (i.e., $R_{dk} \geq 1$), then there is no problem since the demanded availability is still satisfied. However, if any R_{dk} falls below 1, then the corresponding SLA will be violated. For simplicity, here we assume a simple pricing and refunding model, where the charge for serving a user demand d is g_d, and if the SLA is violated, a fraction μ_d of g_d will be refunded. We use r_d to denote the profit of demand d with refunding, such that

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

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

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

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

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

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

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

and

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

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

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

The above mixed-integer linear programming (MILP) problem can be proved to be NP-hard, and the proof details can be found in Appendix C. To efficiently solve this problem, we further propose a 2-approximation greedy algorithm. The key idea is to prioritize demands by the ratio of profit to the allocated bandwidth in a non-decreasing order. Due to space limitations, the detailed algorithm, as well as the proof on its optimality, are put into 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

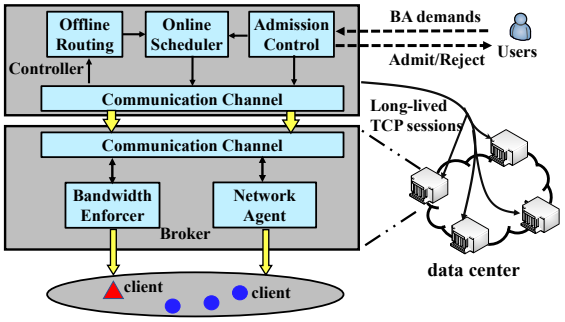


Figure 3: System architecture of BATE

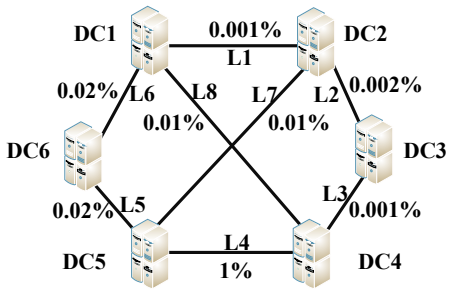


Figure 4: Testbed topology with failure probabilities

in Controller are as follows. (1) Offline Routing. This module maintains the WAN level network topology, and computes TE tunnels between each node pair (i.e., T_k, $\forall$s-d pair $k \in K$), using certain routing algorithms (oblivious routing [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 bandwidth over the tunnels in nearly real-time. The results are sent to the corresponding brokers. (3) Online Scheduler. Periodically, This module performs traffic scheduling (see § 3.3) according to the bandwidth availability demands submitted by users, so that the availability can be optimized in a probabilistic sense. It also pre-computes backup allocation (see § 3.4) for some potential link failures. For each user demand, the normal bandwidth and backup bandwidth allocated over each tunnel (i.e., f_i^t) are then sent to the corresponding brokers. In addition, our system also supports several other TE algorithms including SWAN [22], FFC [37] 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.

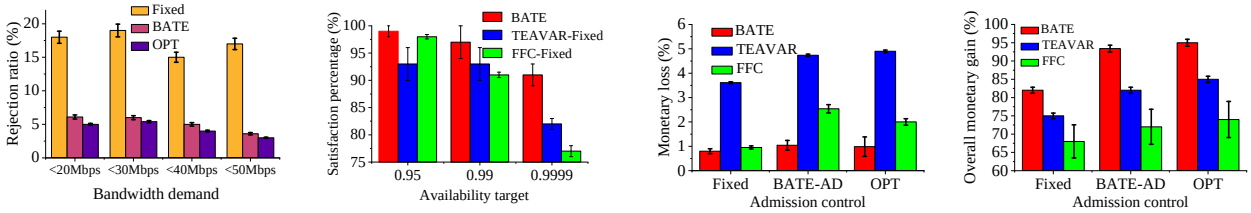
Client sits on each host to carry out rate limiting. It consists of a daemon and a kernel module. The former recieves bandwidth allocation results from the site broker, while the latter sits between the TCP/IP stack and the Linux Traffic Control (TC) module to control outbound traffic rate.

5 EVALUATION

In this section, we use a small testbed and larger scale trace driven simulations to evaluate the performance of BATE. On the testbed, we also implement another two state-of-the-art TE algorithms considering network availability, i.e., FFC [37] and TEAVAR [10]. For simulation, we implement more TE algorithms, including xxx. All codes for our implementation and evaluation will be published, so that interested readers can use them in their own environments or make further improvements.

Our main results are as follows:

(1) BATE consistently ourperforms latests TE algorithms under various topologies, traffic matrices and failure scenarios. With BATE, up to 40% more bandwidth availability demands can be successfully fulfilled, and 30% more profit can be retained when failures occur.



(a) Admission control (b) Bandwidth availability (c) Profit loss after failures (d) Overall profit gain

Figure 5: BA demands arrive with poisson process and their duration are with exponential distribution.

(2) BATE achieves a good tradeoff between efficiency and optimality. Compared with the optimal solutions, (i) our admission control algorithm can speed up the admission procedure by 30× at the expense of only xx% false rejections, (ii) our pruning-augmented scheduling algorithm runs xx× faster while wasting only 6% total bandwidth and losing only 8% overall availability, and (iii) our greedy failure recovery algorithm can reduce the reaction time by xx×, where the profit loss is below 10%.

5.1 Testbed evaluation

Testbed setup We build a testbed with 6 servers to emulate a small inter-DC WAN connecting 6 DCs, as shown in Figure 4. The inter-DC links run at 1Gbps, and we add 100ms delay on each link to emulate a WAN environment. Each server is equipped with 4 Intel Xeon E5-2620 CPUs, 64GB memory and 4 Ethernet NICs, and on each server we start 20 VMs, which are all connected to an Open vSwitch [43]. The VMs run CentOS 7 and use Linux v4.15.6 kernel [31] with our rate limiting module plugged in. Every second, we emulate link failures by shutting down the corresponding network interfaces under the failure probabilities shown in Fig. 4, and bring them online after 3 seconds. We also deploy our controller and brokers on extra VMs. The network agent module in each broker uses Floodlight [14] to control the vSwitch, while the latter monitors link status and reports any failure to the former. If not stated otherwise, we use 4 shortest paths between each source-destination pair as the tunnels in TE algorithms.

Evaluations on continuous demand arrivals. We first conduct experiments where user demands are generated from models used in some latest inter-DC traffic scheduling algorithms [10, 28, 38, 51]. For each source-destination pair, the arrival of user bandwidth demands follows a poisson process, and the demand duration follows an exponential distribution. The mean inter-arrival time we use is 2 minutes, and the mean duration is 5 minutes. The demanded bandwidth is uniformly distributed between 10 Mbps and 50 Mbps. The availability targets and refunding ratio are chosen from the Azure SLAs [8], and we assume a unit price is charged for 1 Mbps. Each experiment lasts 100 minutes and is repeated for 50 times, where link failures occur probabilistically,

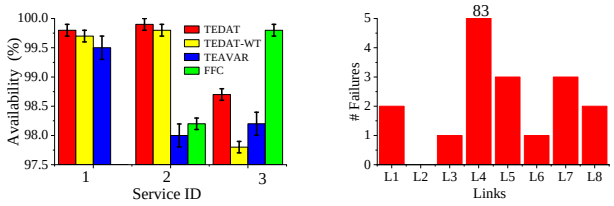
Admission control. We evaluate how demands can be correctly admitted by BATE, comparing with both the optimal admission strategy, and a strategy that assumes a *fixed* bandwith allocation for demands that have already been admitted, which is effectively step (1) in BATE. Figure 5(a) demonstrates that BATE performs closely to the optimal strategy, i.e., the false rejection ratio is at most 1%, while the *fixed* algorithm rejects least 10× more admissible demands.

Traffic scheduling. We evaluate, once a user demand is admitted, how often its bandwidth availability requirement can be guaranteed. Since we emulate different link failures according to their probabilities in each second, we can measure how the bandwidth a user actually uses deviates from its requirement. If such a deviation is less than 1%, we regard the bandwidth availability as *satisfied in that second*. Figure 5(b) shows the overall fraction of satisfaction, under different levels of availability requirements (i.e., 95%, 99% and 99.99%). We note that, FFC-fixed (or TEAVAR-fixed) in the figure represents applying FFC (or TEAVAR) only to demands admitted by the fixed admission control strategy, where the total bandwidth required for the admitted demands will be much lower. BATE always achieves the highest availability, even compared with FFC-fixed and TEAVAR-fixed. In particular, it has a clear advantage for high availability requirements (e.g., >99.99%).

Failure Recovery. We evaluate when failures do occur and cause SLA violations, how profit loss can be mitigated by our failure recovery scheme. Figure 5(c) depicts the fraction of profit loss caused by SLA violations under three different admission control strategies, i.e., fixed, BATE-AD (which is the strategy BATE uses) and optimal, where baseline for each algorithm is the profit it can achieve when no failures occur. Compared with TEAVAR, FFC has a lower profit loss ratio due to its conservative bandwidth allocation.

Table 3: Scheduled results of different schemes.

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



(a) Bandwidth availability (b) Failures of links

Figure 6: An simple example with three user demands whose details are shown in Table 3.

Table 4: Network topologies used in the simulations

Topology Name	#Nodes	#Links
IBM	18	48
B4	12	38
ATT	25	112
a national-level backbone	14	32

Overall Profit. Figure 5(d) plots the overall profit gain comparison among BATE, FFC and TEAVAR for all arrived demands. Thanks to the hard guarantee of bandwidth availability and optimization of profit gain after failures, BATE can maintain at least 15% more profit than FFC and TEAVAR.

Due to space limitations, more details of the evaluations are shown in Appendix E.

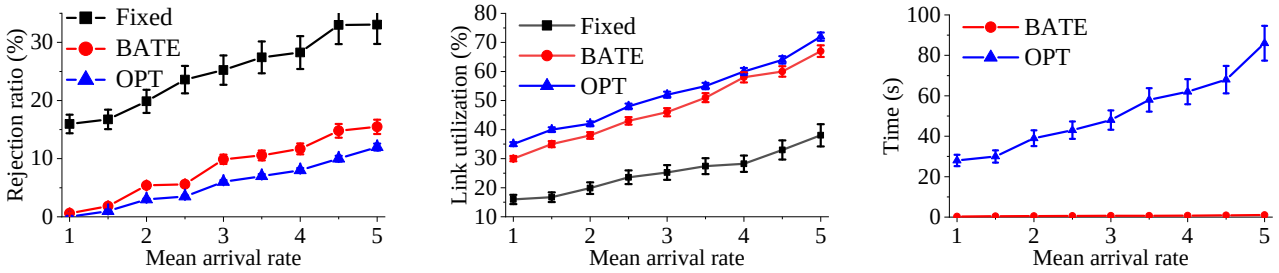
Evaluations on parallel demands. Now we use another example with three user demands to illustrate more details of BATE. In our evaluation, we also compare with another scheme named BATE-TE, i.e, the traffic scheduling part of BATE, with its fast failure recovery scheme abandoned. Demand-1 requires 1000Mbps from DC1 to DC3, demand-2 requires 500Mbps from DC1 to DC4, and demand-3 requires 1500Mbps from DC1 to DC5, with their availability target set as 99.5%, 99.9% and 95%, respectively. We start their traffic simultaneously, assuming all of them have been admitted, and their bandwidth on each path, as shown in Table 3, is determined by different TE algorithms. The experiment lasts 100s and is repeated by 100 times. Figure 6(a) shows the fraction of time each bandwidth availability demand is satisfied, using the same method as in Figure 5(a), i.e, for each second, a gap of more than 1% bandwidth means the demand is not satisfied in that slot. It shows that all the three demands can reach their availability targets under BATE, while TEAVAR and FFC may fail for some users. With an investigation on the bandwidth allocation result in Table 3, we can see that, FFC reserves too much bandwidth for failure recovery, so that demand-1 never gets enough bandwidth (250 Mbps allocated v.s. 1500 Mbps demanded), and its achieved bandwidth availability is always 0. Even it allocates enough bandwidth for demand-2, the achieved bandwidth availability (98.2%) is still lower than required (99.9%). On the other hand, TEAVAR does not make a good match between the link failure probability and the availability users ask for. For example, for demand-2, which needs the highest level of availability (99.9%), TEAVAR still allocates 250 Mbps on link L4, which has the highest failure probability (1%) [7]. On the contrary, BATE matches demands and links well, and does not use L4 for user-2.

5.2 Simulations

Simulation setup. We also conduct simulations on four real network topologies, including B4 [24], ATT [10], IBM[34] and a national-level backbone. Table 4 shows the topology details [8]. Following the conclusion on WAN failures in literature studies[10, 40], we simulate link failures according to a Weibull distribution with its shape $\lambda = 0.8$ and scale $k = 0.00001$. We generate the demand workload in a similar way to that in the testbed. The arrivals of BA demands follow a Poisson process, with a mean inter-arrival time of 2 minutes. The duration of each demand follows an exponential distribution, and the mean duration corresponds to 1000 time slots. The required bandwidth in each user demand is randomly drawn from the traffic metrices (we have collected 200 matrices for each topology) with a proper scale down factor (we use 5), so that between each source-destination pair, multiple users can be served simultaneously. The required availability in each user demand is drawn from 10 different SLAs of Azure[8]. In our simulations, besides FFC and TEAVAR, we also compare against several other TE algorithms, including SWAN [22], SMORE [34] and B4 [24]. They

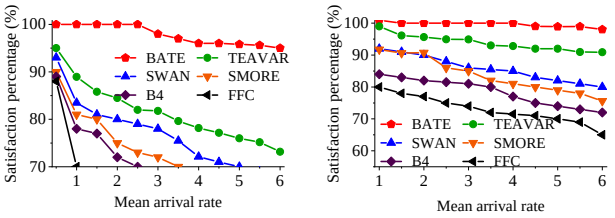
[7] In Figure 6(b), we plot the actual number of failures that occur in the 100 experiments, where L4 fails most frequently.

[8] For B4, ATT and IBM, we get their topologies, link capacities and traffic matrices from the authors of TEAVAR[10], and for the national backbone, we conduct a direct measurement on it but hide its name due to anonymity requirement.



(a) Rejection percentage (b) Link utilization (c) Admission time

Figure 7: BATE **vs. Fixed admission control under different network topologies.**



(a) No extra admission control (b) Fixed admission control

Figure 8: Traffic scheduling schemes comparisons.

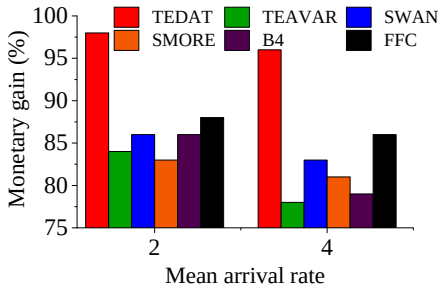


Figure 9: Profit gain after failures

have not explicitly considered availability, but pay attention to total throughput, link utilization or user fairness. These TE algorithms will be activated every 10 time slots. We assume at most one link failure (i.e., no concurrent failures) in FFC, use 99.9% (the maximum value in the user demands) as the default availability target in TEAVAR, and let SWAN maximize the total throughput of all users. With the above settings, each simulation lasts 150,000 slots (corresponding to 100 days if each slot represents 1 minute), and the results that achieved by each algorithm on each topology are calculated on 5 independent simulations with different workload traces.

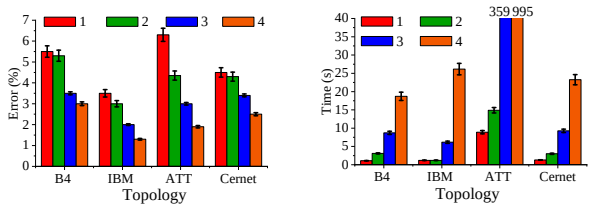
Evaluation results.

Figure 7(a) shows BATE rejects only about 3% more demands than the optimal solution and can accept up to 50% more demands than the *Fixed* consistently. Figure 7(b) demonstrates BATE is able to maintain at least 10% higher link utilization than *Fixed* algorithm. 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 7(c), 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.

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 8(a) demonstrates that BATE can even make up to 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 8(b) shows BATE performs at least 10% better than other algorithms under the *Fixed* admission control method.

Figure 9 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.

Traffic scheduling algorithm cuts off the scenarios that can hardly happen. Let BATE denote the overall allocated bandwidth of all admitted demands with scenario pruning and *OPT* denote the allocated bandwidth of optimal solution without scenario pruning. Then the error can be defined as



(a) Cutoff error (b) Optimization time

Figure 10: Impact of pruning for traffic scheduling.

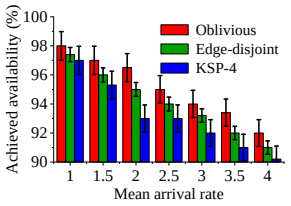


Figure 11: Routings.

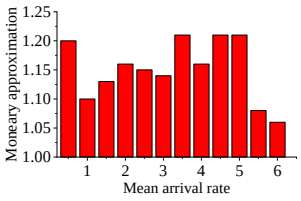


Figure 12: Approximation.

$\frac{|\text{BATE}-OPT|}{|OPT|}$. Figure 10 (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 10 (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 11 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 over-utilization through diverse and low-stretch path selection.

We propose a greedy for failure recovery optimization problem. Figure 12 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

Optimizing WAN performance is a big challenge. One important topic is on network utilization or fairness. For example, early studies focus more on tuning parameters of widely used routing protocols, such as OSPF [15] and MPLS [13, 29], for given traffic matrices. Recently, Software defined network (SDN) based technologies, including SWAN[22], B4[23, 24], Bwe[33] and OWAN[28], rely on a centralized view to optimize bandwidth allocations. There are also scheduling algorithms [25, 30, 50] using SDN to prioritize WAN traffic. These work mainly consider aggregated traffic in a macro level, while BATE handles traffic demands of users.

As more applications are depolyed in cloud or datacenters, many work study how to provide performance guarantee for intra-DC or inter-DC user traffic, including flow deadline [47, 51, 52] and flow rate [26, 36]. However, they do not provide adquate mechanisms to deal with potential or actual failures.

Network failures (or uncertainties) have also been considered in various aspects for large scale network evnironments, including design datacenter networks [19] and optical networks [17], stochastic models [41] [9] and failure recovery methods [45, 49]. BATE studies both proactive and reactive traffic engineering schemes to take network failures into account, so that violations on service level agreements can be avoided or mitigated. As far as we know, FFC [37] and TEAVAR [10] are two pieces of work that are most close to BATE, in the sense that they also try to provide certain performance guarantee for inter-DC WAN, even under failures. However, they have not taken into account the heterogeneities and competitions of user demands, and the economic interests of service providers.

7 CONCLUSION

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

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] 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

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

[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 THE ADMISSION CONTROL PROBLEM

Firstly, the bandwidth allocation results f_d^t for BA demand d over tunnel t should be non-negative, i.e.,

$$f_d^t \geq 0, \quad \forall d \in D, k \in K, t \in T_k. \tag{13}$$

Then total traffic through link e should not exceed its capacity c_e:

$$\sum_{d \in D} \sum_{k \in K, t \in T_k} f_d^t u_t^e \leq c_e, \quad \forall e \in E. \tag{14}$$

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

$$R_{dk}^{\mathbf{z}} = \frac{\sum_{t \in T_k} f_d^t v_t^{\mathbf{z}}}{\mathbf{b}_d^k}, \quad \forall d \in D, \mathbf{z} \in \mathbf{z}, k \in K. \tag{15}$$

Under network scenario $\mathbf{z}$, tunnel t might be unavailable (i.e., $v_t^{\mathbf{z}} = 0$). For every source-destination pair k, if the total reserved bandwidth through all the available tunnels is larger than $\mathbf{b}_d^k$, then bandwidth requirement can be satisfied even tunnel t fails and network scenario $\mathbf{z}$ can be regarded as *qualified*. Let $A_d^{\mathbf{z}}$ denote whether scenario $\mathbf{z}$ is qualified (i.e., $A_d^{\mathbf{z}} = 1$) or not (i.e., $A_d^{\mathbf{z}} = 0$) for a BA demand d:

$$A_d^{\mathbf{z}} = \begin{cases} 1 & \forall k \in K : R_{dk}^{\mathbf{z}} \geq 1 \\ 0 & \text{Otherwise} \end{cases}, \forall d \in D, \mathbf{z} \in \mathbf{z}. \tag{16}$$

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

$$\begin{cases} R_{dk}^{\mathbf{z}} < 1 - A_d^{\mathbf{z}} + M \times A_d^{\mathbf{z}}, & \forall d \in D, \mathbf{z} \in Z, k \in K. \\ R_{dk}^{\mathbf{z}} >= A_d^{\mathbf{z}}, & \forall d \in D, \mathbf{z} \in Z, k \in K. \\ A_d^{\mathbf{z}} \in \{0, 1\}, & \forall d \in D, \mathbf{z} \in Z. \end{cases} \tag{17}$$

where M is a big integer that is at least larger than the upper bound of $R_{dk}^{\mathbf{z}}, \forall d \in D, k \in K, \mathbf{z} \in Z$. It is easily to prove (17) equals to (16). Here, our consideration is tunnel t might be unavailable (i.e., $v_t^{\mathbf{z}} = 0$) under a network scenario, but if $R_{dk}^{\mathbf{z}} \geq 1$ holds, then this scenario is still *qualified*, i.e., $\mathbf{z} \propto < d, \{f_d^t\} > \quad \Leftrightarrow \quad \forall k, A_d^{\mathbf{z}} = 1$. The achieved bandwidth availability of demand d is the sum of all qualified network scenarios' probability:

$$S_d = \sum_{\mathbf{z} \in \mathbf{z}} A_d^{\mathbf{z}} \times p_{\mathbf{z}}, \quad \forall d \in D. \tag{18}$$

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

$$y_d = \begin{cases} 1 & 1 > S_d \geq \beta_d \\ 0 & \beta_d > S_d \geq 0 \end{cases}, \quad \forall d \in D. \tag{19}$$

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, (19) can be changed into the following linear format:

$$\begin{cases} S_d < \beta_d \times (1 - y_d) + y_d, & \forall d \in D. \\ S_d \geq \beta_d \times y_d, & \forall d \in D. \\ y_d \in \{0, 1\}, & \forall d \in D. \end{cases} \tag{20}$$

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

$$\begin{aligned} &maximize \sum_{d \in D} y_d \\ &s.t. (13), (14), (15), (17), (18), (20) \end{aligned} \tag{21}$$

B PROOF OF THEOREM 1

Proof. We use the contradiction method to prove, i.e., there is a BA 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., $\mathbf{b}_d^k >$ remaining capacity of s-d pair k) , Algorithm 1 won't admit the demand (Line 4-5). Case (ii) is also impossible, because if the bandwidth availability is smaller than its target (i.e., $s_d < \beta_d$) , Algorithm 1 will reject the demand (Line 14-15). This completes the proof. □

C PROOF OF NP-HARDNESS IN FAILURE RECOVERY

Proof. The all-or-nothing multi-commodity flow problem, which is known to be NP-hard[12], can be regarded as a special case of our failure recovery 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. L_{pe} denotes whether path p goes through link e and f_{ip} is the allocation result of commodity i over path p. The all-or-nothing multi-commodity flow problem tries to find a maximum weight routable set:

$$\begin{aligned} &maximize \sum_{i=1}^{k} w_i \times y_i \\ &s.t. \qquad \forall e \in E : \sum_{i=1}^{k} \sum_{p \in \mathcal{P}_i} f_{ip} L_{pe} \leq c_e \\ &\forall 1 \leq i \leq k : y_i = \begin{cases} 1 & \sum_{p \in \mathcal{P}_i} f_{ip} \geq d_i \\ 0 & \sum_{p \in \mathcal{P}_i} f_{ip} < d_i \end{cases} \end{aligned} \tag{22}$$

Where y_i denotes whether commodity flow i can be routable. We transform the all-or-nothing multi-commodity flow problem to a special instance of failure recovery problem and consider a special case, in which $\mu_d = 0, \forall d \in \hat{D}$. 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, a failure scenario z
Output: $\{f_d^t\}, F$

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

In this part, we will introduce a greedy algorithm, shown in Algorithm 2, to solve failure recovery problem. Let F denote the BA demands set that derive full profit (i.e. $h_d = 1$). Firstly, it sorts all the accepted demands $d \in \hat{D}$ in non-decreasing order according to the ratio of demand profit to aggregate bandwidth demands, where the aggregate bandwidth demands are derived as $\sum_{k\in K} \mathbf{b}_d^k$ (Line 1). The ordered sequence prefers demands that have large profit and small bandwidth. The algorithm then loops all the ordered admitted demands and tries to allocate resources with remaining network capacity (Line 5-9). If the network is able to support current demand, then add to F (Line 7). If the network is unable to support current demand but it has larger profit than the total profit of previous ones in F, the algorithm will try to recycle total resources that are allocated to F and test that if allocating total network resources can support current demand (Line 12-16). If this is true, then algorithm will prefer current demand, otherwise, the algorithm finishes the iteration (Line 17-18). Compared with the bruce force algorithm, Algorithm 2 can derive solution in $O(|\hat{D}||T_k||E|)$, which is Polynomial time. However, it achieves this at the cost of performance loss, which is proven as follows.

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

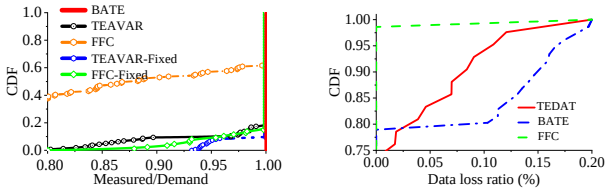
Proof. Algorithm 2 prefers accepted demands according to the following sequence:

$$\frac{g_1}{\sum_{k\in K} \mathbf{b}_1^k} \geq \frac{g_2}{\sum_{k\in K} \mathbf{b}_2^k} \geq .... \tag{23}$$

(23) 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 $max\{g_{n+1}, \sum_{i=1}^{n} g_i\}$ as the value. Let OPT denote the optimal solution and it is obvious that $\sum_{i=1}^{n} g_i \leq OPT$. Also, we have $\sum_{i=1}^{n+1} g_i \geq OPT$. This holds, since we've already made the density of network as high as possible by the greedy method. If we violate the link capacity constraint and put the $n+1$ demand into links, then links are fulfilled. There is no other way that the density of links are greater than this, that is, the value is greater than OPT. $\sum_{i=1}^{n+1} g_i/2 \leq max\{\sum_{i=1}^{n} g_i, g_{n+1}\}$. Therefore, $OPT/2 \leq max\{\sum_{i=1}^{n} g_i, g_{n+1}\}$. This completes the proof. □

E MORE EVALUATION DETAILS

we plot in Figure 13(a), for each algorithm, the ratio of the allocated bandwidth to the demanded bandwidth. The CDF curve shows that FFC is too conservative in bandwidth allocation, and fails to allocate proper bandwidth in almost 60% time. On the other hand, although TEAVAR provides bandwidth fairly well, it ignores the diverse availability requirements of different users, and achieves a lower satisfaction ratio than BATE.



(a) Measured and demand ratio (b) Data loss comparison

Figure 13: More details of testbed evaluations

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.