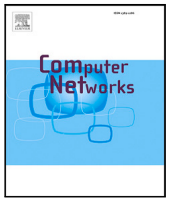
Computer Networks 197 (2021) 108271

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet



Achieving high reliability and throughput in software defined networks

Xuwei Yang [a], Hongli Xu [a,*], Jianchun Liu [e], Chen Qian [b], Xingpeng Fan [a], He Huang [c], Haibo Wang [d]

[a] School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui, 230027, China
[b] Department of Computer Engineering, University of California at Santa Cruz, Santa Cruz, CA 95064, USA
[c] School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu, 215006, China
[d] Department of computer and information science and engineering, University of Florida, FLorida, FL 32611, USA
[e] School of Data Science, University of Science and Technology of China, Hefei, Anhui, 230027, China

ARTICLE INFO

Keywords:
Software defined networks
Reliability
Failure probability
Approximation

ABSTRACT

Flow routing is one of the most important issues in a software defined network (SDN), and faces various challenges. For example, each link may not be reliable all the time (or with a failure probability), and the flow-table size on each switch is limited. Existing solutions about reliable flow routing may result in a longer failure recovery delay, a large number of flow entries or massive control overhead. To this end, we propose to achieve throughput optimization with the constraint that the forwarding reliability probability of each switch pair should exceed a threshold (*e.g.*, 99.9%). We formally define the reliable flow routing (RLFR) problem with flow-table size constraint. We present a rounding-based algorithm and analyze its approximation performance. We further extend our algorithm to preserve the throughput of each switch pair even with link failures. We implement our proposed algorithms on the SDN platform. The experimental results and large-scale network simulation results show that our algorithms can improve the network throughput by about 48.0% and reduce the maximum number of required flow entries by about 53.1%, compared with the existing solutions under the reliability requirement.

1. Introduction

With the rapid development of network technology, more and more applications come into our lives, resulting in heavy traffic of the network. In legacy networks, it is hard to schedule such a large amount of traffic efficiently due to the distributed control. On contrary, with the help of the global perspective and centralized control ability, SDN can provide fine-grained traffic routing control in an optimal way, which has received wide attention in recent years. Some previous works, *e.g.*, SWAN [1] and B4 [2], make full use of the advantages of SDN to optimize the resource utilization and improve the network throughput.

High reliability is a critical requirement for many applications. For example, an interruption between each originator-destination (OD) switch pair will result in a large amount of expensive raw material waste in advanced manufacturing networks [3]. Moreover, low reliability will also affect electronic payment and other businesses in financial networks, which will cause huge economic losses [4]. However, as a negative effect, link failures in a network is very likely to occur during working hours [5]. Though it may be predicted based on sudden drops in optical signal quality (*e.g.*, with a 50% chance of an outage within an hour of a drop event and a 70% chance of an outage within one day [6]), the indulgence of link failure will reduce the network reliability. One may think that switch failure will also occur in a network. However, the previous works have shown that switch failure accounts for only 0.21% of all failure events [7]. Thus, we only take the link failure into account to improve the network reliability in this paper.

In order to ensure the high reliability of SDNs, the traffic passing through the failed primary path(s) needs to bypass the failure link(s) using the backup path(s) when the failure event occurs. To recover from link failures, existing works can be classified into two strategies: the reactive protection and the proactive protection, respectively. Under the reactive strategy [8,9], after detecting a failure event, the controller will recalculate the backup paths for all affected traffic and install the forwarding rules on switches. Since there is no need to install rules for the backup paths in advance, the reactive strategy will significantly reduce the resource cost, such as flow entries. Due to extra route computation and rule update, it may lead to a longer recovery delay (*e.g.*, more than 1 s [9]), which cannot meet the requirements of many applications. For example, voice propagation delay above 400 ms in interactive applications is unacceptable to human [10].

* Corresponding author.
E-mail address: xuhongli@ustc.edu.cn (H. Xu).

https://doi.org/10.1016/j.comnet.2021.108271
Received 9 December 2020; Received in revised form 16 June 2021; Accepted 27 June 2021
Available online 7 July 2021

More seriously, a long recovery delay may even cause packet loss and retransmission. Thus, this strategy is infeasible for many real-time network applications.

As an alternative way, the proactive strategy [11–13] requires the controller to deploy backup paths for all primary paths before the occurrence of the underlying failure events. When some links fail, the traffic through these links will be quickly rerouted to the backup path(s) without disturbing the controller, thus reducing the failure recovery delay. However, the pre-deployed backup paths will consume a large number of flow-table entries on each switch [13]. On most commodity switches, flow rules are stored in ternary content addressable memory (TCAM), which is expensive and power-hungry [14]. As a result, there are only thousands of flow entries on the mainstream SDN switches [15]. Although there are more entries on SDN switches recently, too many occupied entries on a switch will degrade the flexibility of data plane. For example, the per-rule update time is 3.3 ms in an empty TCAM table and increased to 18 ms when only 600 rules in the flow-table. The rule insertion time will further increase with a large number of installed rules on a switch [16]. Due to traffic dynamics in practice, the switch needs to update hundreds of new rules per second in an SDN [17]. If too many flow entries are installed on a switch, the switch cannot update these flow rules in time. In addition, managing a large number of flow entries will also cause huge controller overhead [17]. Thus, the proactive strategy is not suitable in a large-scale network scenario.

To make full utilization of network resources, some traffic engineering solutions *e.g.*, [18,19], achieve failure recovery by re-allocate traffic across multiple paths after failures occur, but ignore link failure probability. Some prior works assume the uniform failure probability of all links [20]. In practice, the failure probability of all links is different, and may differ by more than three orders of magnitude [21]. Thus, it is unreasonable to ignore the difference of failure probabilities among all links. Inspired by financial risk theory, Bogle et al. [21] explicitly accounts for the likelihood of different failure events and adjusts the bandwidth of each flow to minimize a formal notion of risk to an acceptable level. However, this solution has two main disadvantages. (1) It is unable to ensure the reliability requirement between each originator-destination (OD) switch pair; and (2) needs a lot of flow-table resources on some switches. Moreover, these works do not consider the quantitative relationship between reliability, flow-table size and throughput, which is a critical challenge for SDNs.

To meet these challenges and constraints, we study the throughput-maximization reliable flow routing in SDNs. In this paper, we choose a set of paths for each switch pair in advance with two constraints. One is the reliability requirement for each flow. Specifically, it is required that the reliability probability of each switch pair should exceed a threshold α (*e.g.*, 99.9%) [21]. In other words, the probability of simultaneous failure of all paths between each switch pair is less than 1- α (*e.g.*, 0.1%). The other is the flow-table size constraint. The previous reliable routing solutions [20,21] often ignore the flow-table size constraint. As a result, many paths may go through some hot spot switches and the flow-table size constraint on these switches may be violated. Therefore, we carefully choose some paths for each flow without overusing the flow table entries on each switch. After path selection, we derive a weight for each path and divide flows among these paths according to the weight. When some links fail, we can quickly adjust the weights of paths for affected flows to migrate the traffic from the failed paths to other healthy paths.

The main contributions of this paper are as follows. We formally define the reliable flow routing problem with flow-table size constraint in SDNs. We propose a rounding-based algorithm for this problem and analyze its approximation performance. When some failure events occur, we further consider how to implement the throughput preserving reliable routing scheme, and design the optimal solution using linear programming. Our proposed algorithms are implemented on a small-scale testbed with OVS. We then perform evaluation through large-scale network simulations. The experimental results and extensive simulation results show that our proposed algorithms improve the network throughput by about 48.0%, compared with the alternatives. Moreover, the proposed algorithms can reduce the failure recovery delay by about 72.7% compared with the reactive strategy, and reduce the number of required flow entries by about 53.1% compared with the proactive strategy.

2. Preliminaries

In this section, we first introduce the network model, the flow model and the path reliability model. Then, we give the preliminaries of the reliable flow routing problem.

2.1. Network and flow models

An SDN typically consists of a logically-centralized controller, and a set of switches $V = \{v_1, \dots, v_m\}$, with $m = |V|$. The network topology can be modeled by $G = (V, E)$, where E is a set of links connecting these switches. The capacity of each link $e \in E$ is characterized by $c(e)$, and each switch v is only capable of a limited number $\beta(v)$ of flow entries. Since we focus on the data-plane performance (*e.g.*, load balancing, network throughput), the number of controllers will not significantly impact these metrics. Thus, we assume only one controller in the control plane for simplicity.

We denote the flows based on originator-destination (OD) switch pairs as $\Gamma = \{f_1, \dots, f_n\}$, with $n = |\Gamma|$. A flow f is the aggregation of traffic with the same originator (ingress) switch and destination (egress) switch [14,21]. Let $r(f)$ be the estimated traffic rate (or flow rate) of f through long-term observations.

2.2. Path reliability model

The controller pre-computes a set of candidate paths, denoted as T^f, for each flow f. For simplicity, we enumerate all the candidate path sets $D(f) = \{d | d \subseteq T^f,\ d \neq \phi\}$ for each flow f in advance. Obviously, if $|T^f| = k$, it follows $|D(f)| = 2^k - 1$. In this paper, we consider the impact of underlying link failure events on the network performance and resource cost in an SDN. The failure probability of each link e is denoted as $\mathcal{P}(e)$. For each path t, its failure probability can be expressed as

$$\mathcal{P}(t) = 1 - \prod_{e \in t} (1 - \mathcal{P}(e)) \tag{1}$$

The failure probability of a given path set d is the probability that all paths in this set fail simultaneously. To this end, we should consider two cases. The first case is that these paths are link-disjoint. The failure of each path $t \in d$ is independent and the failure probability of all paths in set d is

$$\mathcal{P}(d) = \prod_{t \in d} \mathcal{P}(t) \tag{2}$$

The second case is that some paths may share at least one link. The failure of the shared links will affect the reliability of multiple paths, which makes the computation of failure probability of a path set more difficult. To this end, we define E^d as the set of links which are shared by multiple paths in d. We use Θ^d to keep all the possible combinations of shared links in E^d. Obviously, if $|E^d| = l$, $|\Theta^d| = 2^l$. For each shared link set $B^d \in \Theta^d$, we define $\mathcal{P}(B^d)$ as the probability that all links in set B^d fail and other links in $E^d \setminus B^d$ work well. That is,

$$\mathcal{P}(B^d) = \prod_{e \in B^d} \mathcal{P}(e) \cdot \prod_{e \in E^d \setminus B^d} (1 - \mathcal{P}(e)) \tag{3}$$

For ease of calculation, we define $\mathcal{P}(t|B^d)$ as the failure probability of path t under the condition that only all links in set B^d fail. That is

$$\mathcal{P}(t|B^d) = \begin{cases} 1 & , \text{if } e \in t, \exists e \in B^d \\ 1 - \prod_{e \in t \setminus E^d} (1 - \mathcal{P}(e)) & , \text{if } e \notin t, \forall e \in B^d \end{cases} \tag{4}$$

We then define $\mathcal{P}(d|B^d)$ as the failure probability of path set d under the condition that only all links in set B^d fail according to multiplication formula of probability [22].

$$\mathcal{P}(d|B^d) = \prod_{t \in d} \mathcal{P}(t|B^d) \tag{5}$$

According to the total probability theorem [22], we obtain the failure probability of path set d as

$$\mathcal{P}(d) = \sum_{B^d \in \Theta^d} \mathcal{P}(B^d)\mathcal{P}(d|B^d) \tag{6}$$

We then denote the reliability probability $\mathcal{R}(d)$ of a path set d as $1-\mathcal{P}(d)$. For the sake of unification, we define the function $\mathcal{F}(\{z_f^t\}|\forall t \in T^f)$ to represent the reliability probability of flow f, where z_f^t denotes whether f selects path t or not.

For ease of understanding, we give an example to explain the failure probability of a path set. In Fig. 1, a flow f from v_1 to v_4 selects three paths t_1, t_2 and t_3. In the left plot, the failure probability of paths $\{t_1, t_2, t_3\}$ is $\{0.19, 0.37, 0.28\}$ according to Eq. (1) (*e.g.*, $\mathcal{P}(t_1) = 1-(1-0.1)\times(1-0.1) = 0.19$). Since these paths are link-disjoint, we further compute the failure probability of path set $d_1 = \{t_1, t_2, t_3\}$ according to Eq. (2). That is, $\mathcal{P}(d_1) = 0.19 \times 0.28 \times 0.37 = 0.019684$. If paths t_1 and t_3 share one link v_1v_5 as shown in the right plot, it is difficult to compute the failure probability of the path set d_1. We first get $\Theta^{d_1} = \{\phi, \{v_1v_5\}\}$, where $B_1^{d_1} = \phi$ and $B_2^{d_1} = \{v_1v_5\}$ with $\mathcal{P}(B_1^{d_1}) = 0.8$ and $\mathcal{P}(B_2^{d_1}) = 0.2$ according to Eq. (3). If link v_1v_5 is healthy, the failure probability of paths $\{t_1, t_2, t_3\}$ is $\{0.19, 0.37, 0.1\}$ according to the second part of Eq. (4). The failure probability of the path set d_1 is $\mathcal{P}(d_1|B_1^{d_1}) = \prod_{t\in d_1}\mathcal{P}(t|B_1^{d_1}) = 0.19\times0.37\times0.1 = 0.00703$ according to Eq. (5). If the shared link v_1v_5 fails, both t_1 and t_3 also fail, and the failure probability of paths $\{t_1, t_2, t_3\}$ is $\{1, 0.37, 1\}$ according to the first part of Eq. (4). The failure probability of the path set d_1 is $\mathcal{P}(d_1|B_2^{d_1}) = \prod_{t\in d_1}\mathcal{P}(t|B_2^{d_1}) = 1\times0.37\times1 = 0.37$ according to Eq. (5). So the failure probability of the path set d_1 is $\mathcal{P}(d_1) = \mathcal{P}(B_1^{d_1})\mathcal{P}(d_1|B_1^{d_1}) + \mathcal{P}(B_2^{d_1})\mathcal{P}(d_1|B_2^{d_1}) = 0.8\times0.00703+0.2\times0.37 = 0.079624$ according to Eq. (6). In this example, we select three paths $\{t_1, t_2, t_3\}$ for flow f in the right plot of Fig. 1. It follows $z_f^{t_1} = z_f^{t_2} = z_f^{t_3} = 1$. We compute its reliability function as $\mathcal{F}(\{z_f^{t_1}, z_f^{t_2}, z_f^{t_3}\}) = \mathcal{F}(\{1,1,1\}) = 1-0.079624 = 0.920376$.

2.3. Definition of reliable flow routing problem

In this section, we formally define the reliable flow routing (RLFR) problem for an SDN. We will choose one or several paths from T^f for each flow f with two constraints. One is the reliability requirement. It is required that the reliability probability of the chosen path set for each flow should exceed the reliability threshold α (*e.g.*, 99.9%). The other is the flow-table size constraint. If switch v lies on the path(s) of a flow, the controller will install a flow entry on switch v for this flow. The number of installed flow entries on switch v should not exceed its flow-table size $\beta(v)$. We should note that if switch v lies on multiple paths of a flow (*e.g.*, switch v_1 in the left plot of Fig. 1), a group entry is required to support the multi-path forwarding [15]. Specifically, each group entry contains multiple action buckets, and each bucket contains an action (*e.g.*, forward to an output port, drop) along with its probability (or weight). By properly setting the action field and the weight of each bucket in a group entry, the multi-path routing based on weight allocation can be realized. We will assign a weight of each path for flow f, which determines the fraction of the traffic amount of f through this path. Our objective is to balance the load among all links. We give the definition of RLFR in Eq. (7).

$$\min \lambda$$

$$S.t \begin{cases} y_f^t \le z_f^t, & \forall t \in T^f, f \\ \sum_{t\in T^f} y_f^t = 1, & \forall f \in \Gamma \\ \mathcal{F}(\{z_f^t\}|\forall t \in T^f) \ge \alpha, & \forall f \in \Gamma \\ z_f^t \cdot I_t^v \le r_f^v, & \forall t, v, f \\ \sum_{f\in\Gamma} r_f^v \le \beta(v), & \forall v \in V \\ \sum_{f\in\Gamma}\sum_{t\in T^f: e\in t} y_f^t \cdot r(f) \le \lambda \cdot c(e), & \forall e \in E \\ z_f^t, r_f^v \in \{0,1\}, & \forall t \in T^f, f, v \\ y_f^t, \in [0,1], & \forall t \in T^f, f \end{cases} \tag{7}$$

Variable y_f^t denotes the weight of path t for flow f. The first set of inequalities constrains the weight of each path. The second set of equations tells that all traffic of each flow will be forwarded successfully. The third set of inequalities indicates that the reliability probability of the chosen path set for each flow should exceed a threshold of α. Function $\mathcal{F}$ calculates the reliability probability of the chosen path set for a flow, which can be derived by Eqs. (1)–(6). The fourth and the fifth sets of inequalities denote the flow-table size constraint on each switch, in which constant I_t^v denotes whether switch v lies on path t or not, and variable r_f^v denotes whether flow f passes through switch v or not. The sixth set of inequalities expresses the link capacity constraint. Our objective is to achieve the load balancing among all links, that is, min λ.

2.4. The PRLF problem

Since the reliability function $\mathcal{F}$ in Eq. (7) is very complex, we cannot give the closed-form description for the RLFR problem. It is difficult to derive an efficient algorithm for this problem directly. Alternatively, we introduce an approximate problem, called PRLF, and design a rounding-based algorithm for the modified version.

We can transform the RLFR problem into the PRLF problem according to the following steps. First, we take the idea of some flow routing works [14,23], in which one flow will choose one route path from a candidate set. To this end, we use set $D(f)$ to keep all possible combinations of paths in T^f except ϕ. Obviously, if $|T^f| = k$, it follows $|D(f)| = 2^k - 1$ in the RLFR problem. In order to reduce the complexity, the controller selects an appropriate number of candidate path sets $\bar{D}(f)$ from $D(f)$ for flow f in PRLF. Different from the RLFR problem, we choose one candidate path set $d \in \bar{D}(f)$ that satisfies the reliability requirement for each flow in the PRLF problem. Thus, we give the problem formalization of the PRLF problem as follows.

$$\min \lambda$$

$$S.t \begin{cases} x_f^d = 0, & \forall f \in \Gamma, \mathcal{P}(d) \ge (1-\alpha) \\ \sum_{d\in\bar{D}(f)} x_f^d = 1, & \forall f \in \Gamma \\ z_f^t = \sum_{d\in\bar{D}(f): t\in d} x_f^d, & \forall f \in \Gamma, t \in T^f \\ \sum_{f\in\Gamma}\sum_{d\in\bar{D}(f): v\in d} x_f^d \le \beta(v), & \forall v \in V \\ \sum_{t\in d} y_f^{d,t} = x_f^d, & \forall f \in \Gamma, d \in \bar{D}(f) \\ y_f^{d,t} = 0, & \forall f \in \Gamma, d \in \bar{D}(f), t \notin d \\ \sum_{f\in\Gamma}\sum_{t\in T(f): e\in t}\sum_{d\in\bar{D}(f)} y_f^{d,t} & \\ \cdot r(f) \le \lambda \cdot c(e), & \forall e \in E \\ x_f^t, z_f^t \in \{0,1\}, & \forall t, f, d \\ y_f^{d,t}, \in [0,1], & \forall t, f \end{cases} \tag{8}$$

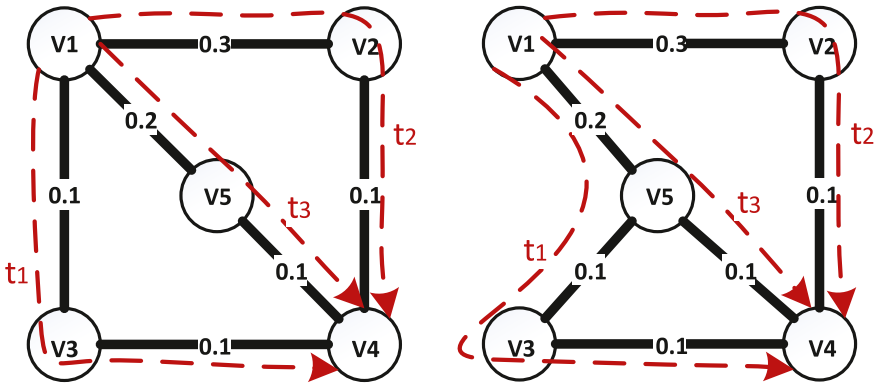


Fig. 1. An example to illustrate the failure probability of a path set for OD-pair (v_1, v_4). Black solid lines and color thick lines denote links and flows, respectively. The number inside each link denotes its failure probability. *Left plot*: paths are link-disjoint; *Right plot*: paths are not link-disjoint.

Variable x_f^d denotes whether flow f selects the path set d for packet forwarding or not. Variable $y_f^{d,t}$ denotes the weight of path t for flow f when path set d is selected for routing, which will determine its traffic rate through path t. The first and the second sets of inequalities indicate that each flow f will select a path set with a reliability probability of at least α. The third and the fourth sets of inequalities denote the flow-table size constraints. The fifth to the seventh sets of inequalities denote the link capacity constraints. The objective is to balance the load among all links, that is, $\min\ \lambda$.

In the PRLF problem, we choose a path subset $\bar{D}(f)$ from $D(f)$ for each flow f. One may think that the network performance can be improved, if we choose more paths for each flow as input. However, due to time and space constraints, we only consider a small number of candidate path sets for each flow as input. We will evaluate the impact of the number of candidate path sets per flow on the network performance in Section 4. Note that the identical parallel machine scheduling (IPMS) problem [24] is a special case of the PRLF problem. As IPMS is NP-hard, the PRLF problem is also NP-hard. We give the detailed proof in Appendix.

3. Algorithm design

3.1. RDBP: Rounding-based algorithm for PRLF

We then design a rounding-based algorithm RDBP for path selection and weight assignment in the PRLF problem. To solve the integer linear program in Eq. (8), the algorithm first constructs a linear program, denoted as LP_1, as a relaxation of the PRLF problem. Specifically, we relax the variables $\{x_f^d\}$ and $\{z_f^t\}$ to be fractional. Since LP_1 is a linear program, the first step of RDBP solves it in polynomial time with a linear program solver (*e.g.*, pulp [25]). Assume that the optimal solutions for LP_1 are denoted as $\{\tilde{x}_f^d\}, \{\tilde{z}_f^t\}$ and $\{\tilde{y}_f^{d,t}\}$, and the optimal result is denoted as $\tilde{\lambda}$. As LP_1 is a relaxation of PRLF, $\tilde{\lambda}$ is a lower-bound result for this problem.

The second step will determine how to select a feasible path set and assign a weight for each selected path. We obtain integer solutions $\{\hat{x}_f^d\}$, $\forall f \in \Gamma$ and $\forall d \in \bar{D}(f)$, using the randomized rounding method [26]. Specifically, we set $\hat{x}_f^d$ to 1 with probability $\tilde{x}_f^d$, which means that flow f will select path set d. Otherwise, $\hat{x}_f^d$ is set to 0. If the path set d' is selected for flow f (*i.e.* $\hat{x}_f^{d'} = 1$), we set the weight to $\frac{\tilde{y}_f^{d',t}}{\tilde{x}_f^{d'}}$ for each path $t \in d'$ (*i.e.*, we set $\hat{z}_f^t$ to 1 and set $\hat{y}_f^{d',t}$ to $\frac{\tilde{y}_f^{d',t}}{\tilde{x}_f^{d'}}$). By the end of this step, we have determined the paths and their weights for each flow. The RDBP algorithm is formally described in Alg. 1.

Algorithm 1 RDBP: Rounding-Based Algorithm for PRLF

Step 1: Solving the Relaxed PRLF Problem
Construct a linear program LP_1.
Obtain the optimal solutions $\{\tilde{x}_f^d\}, \{\tilde{z}_f^t\}$ and $\{\tilde{y}_f^{d,t}\}$.
Step 2: Path Selection and Weight Assignment
Set $\{\hat{x}_f^d\}, \{\hat{z}_f^t\}$ and $\{\hat{y}_f^{d,t}\}$ to 0.
for each flow $f \in \Gamma$ **do**
  Select one path set $d \in \bar{D}(f)$ and set $\hat{x}_f^d$ to 1 with probability $\tilde{x}_f^d$.
for each flow $f \in \Gamma$ **do**
  for each path set $d \in \bar{D}(f)$ **do**
    for each path $t \in T^f$ **do**
      if $\hat{x}_f^d = 1$ **then**
        Set $\hat{y}_f^{d,t}$ to $\frac{\tilde{y}_f^{d,t}}{\tilde{x}_f^d}$, if $t \in d$. Set $\hat{y}_f^{d,t}$ to 0, if $t \notin d$.
        Set $\hat{z}_f^t$ to 1, if $t \in d$.
Select path t for flow f if $\hat{z}_f^t = 1$ with weight $\sum_{d \in \bar{D}(f)} \hat{y}_f^t$.

3.2. Performance analysis

To analyze the approximation performance of the proposed RDBP algorithm, we give two famous theorems for probability analysis.

Theorem 1 (*Chernoff Bound*). *Given n independent variables: $z_1, z_2, \ldots, z_n$, where $\forall z_i \in [0,1]$. Let $\mu = \mathbb{E}[\sum_{i=1}^n z_i]$. Then,* $\mathbf{Pr}\left[\sum_{i=1}^n z_i \geq (1+\epsilon)\mu\right] \leq e^{\frac{-\epsilon^2\mu}{2+\epsilon}}$, *where ϵ is an arbitrary positive value.*

Theorem 2 (*Union Bound*). *Given a countable set of n events: $A_1, A_2, \ldots, A_n$, each event A_i happens with possibility $\mathbf{Pr}(A_i)$. Then, $\mathbf{Pr}(A_1 \cup A_2 \cup \cdots \cup A_n) \leq \sum_{i=1}^n \mathbf{Pr}(A_i)$.*

By the first set of equations in Eq. (8), the reliability requirement for each flow can be satisfied. We then give the approximation performance for link load and the flow-table size constraint.

Lemma 3. *Let $\tilde{u}(v,f)$ denote the number of required flow-table entries for f on switch v by the optimal result of the linear program LP_1. The RDBP algorithm can guarantee that the expected number of required flow entries for flow f on switch v is the same as $\tilde{u}(v,f)$.*

Proof. Let variable $\kappa(v,f)$ denote the number of required flow-table entries for flow f on switch v by our algorithm. The expectation of variable $\kappa(v,f)$ is:

$$\mathbb{E}[\kappa(v,f)] = \sum_{d \in \bar{D}(f): v \in d} \mathbf{Pr}\left[\hat{x}_f^d = 1\right]$$
$$= \sum_{d \in \bar{D}(f): v \in d} \tilde{x}_f^d = \tilde{u}(v,f) \quad (9)$$

Eq. (9) shows that the expected number of required flow entries for f on switch v by RDBP is the same as that by the optimal solution of the linear program LP_1. □

Theorem 4. *Assume that the minimum flow-table size of all switches is denoted by β^{min}. After the rounding process, the number of required flow entries on each switch will not exceed β^{min} by a factor of $\frac{3\log n}{\beta^{min}}+3$, where n is the number of switches in an SDN.*

Proof. We use a variable $\widetilde{u}(v)$ to denote the number of flow table entries installed at switch v computed by RDBP. By the fourth set of inequalities in Eq. (8), the expected number of flow entries at switch v is:

$$\mathbb{E}\left[\sum_{f\in\Gamma}\kappa(v,f)\right]=\sum_{f\in\Gamma}\mathbb{E}\left[\kappa(v,f)\right]$$
$$=\sum_{f\in\Gamma}\widetilde{u}(v,f)=\widetilde{u}(v)\le\beta(v) \tag{10}$$

Combining Eqs. (8), (9) and (10), we have:

$$\begin{cases}\frac{\kappa(v,f)\beta^{min}}{\beta(v)}\in[0,1]\\ \mathbb{E}\left[\sum_{f\in\Gamma}\frac{\kappa(v,f)\beta^{min}}{\beta(v)}\right]\le\beta^{min}.\end{cases} \tag{11}$$

Note that the last part of Eq. (11) is derived from Eq. (10). By applying Theorem 1, assume that ρ_1 is an arbitrary positive value. It follows:

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{\kappa(v,f)\beta^{min}}{\beta(v)}\ge(1+\rho_1)\cdot\beta^{min}\right]\le e^{\frac{-\rho_1^2\beta^{min}}{2+\rho_1}} \tag{12}$$

Now, we assume that

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{\kappa(v,f)}{\beta(v)}\ge(1+\rho_1)\right]\le e^{\frac{-\rho_1^2\beta^{min}}{2+\rho_1}}\le\frac{\mathcal{H}}{n} \tag{13}$$

where $\mathcal{H}$ is the function of network-related variables (such as the number of switches n, *etc.*.) and $\mathcal{F}\to 0$ when the network size grows.

The solution for Eq. (13) can be expressed as:

$$\rho_1\ge\frac{\log\frac{n}{\mathcal{H}}+\sqrt{\log^2\frac{n}{\mathcal{H}}+8\beta^{min}\log\frac{n}{\mathcal{H}}}}{2\beta^{min}} \tag{14}$$

Set $\mathcal{H}=\frac{1}{n^2}$. Eq. (13) is transformed into:

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{\kappa(v,f)}{\beta(v)}\ge(1+\rho_1)\right]\le\frac{1}{n^3},\text{where}\quad\rho_1=\frac{3\log n}{\beta^{min}}+2 \tag{15}$$

By applying Theorem 2, we have,

$$\mathbf{Pr}\left[\bigvee_{v\in V}\sum_{\gamma\in\Gamma}\frac{\kappa(v,f)}{\beta(v)}\ge(1+\rho_1)\right]$$
$$\le\sum_{v\in V}\mathbf{Pr}\left[\sum_{\gamma\in\Gamma}\frac{\kappa(v,f)}{\beta(v)}\ge(1+\rho_1)\right]$$
$$\le n\cdot\frac{1}{n^3}\le\frac{1}{n^2},\quad\rho_1=\frac{3\log n}{\beta^{min}}+2 \tag{16}$$

Note that the second inequality holds, because there are at most n switches in the network G. Eq. (16) means that the proposed RDBP algorithm can guarantee that the total number of flow entries installed at switch $v\in V$ will not exceed the threshold $\beta(v)$ by a factor of $1+\rho_1=\frac{3\log n}{\beta^{min}}+3$. □

In the following, we analyze the approximation performance for the link capacity constraint. We first show the feasibility of the weight assignment by RDBP.

Lemma 5. *For each flow f, RDBP can guarantee the total weight of all selected paths for flow f is 1.*

Proof. Assume that d' is the selected path set for f by RDBP. It follows $\widehat{y}_f^{d,t}=0$ if $t\notin d$ or $d\ne d'$ by Line 12 of RDBP. Thus, the total weight of all selected paths for f is:

$$\sum_{t\in T^f}\Big(\sum_{d\in\widetilde{D}(f)}\widehat{y}_f^{d,t}\Big)$$
$$=\sum_{t\in d'}(\widehat{y}_f^{d',t})=\sum_{t\in d'}\Big(\frac{\widetilde{y}_f^{d',t}}{\widetilde{x}_f^{d'}}\Big)=\frac{\widetilde{x}_f^{d'}}{\widetilde{x}_f^{d'}}=1 \tag{17}$$

Eq. (17) shows that the summation of weight for each path $t\in T(f)$ is one, which ensures the feasibility of the RDBP algorithm for PRTE. □

Lemma 6. *We use $\widetilde{l}(e,f)$ to denote the traffic load on link e from flow f by the result of the linear program LP_1. The RDBP algorithm can guarantee that the expected traffic load on link e from flow f is same as the optimal solution $\widetilde{l}(e,f)$ of the linear program LP_1.*

Proof. Assume that d' is the selected path set for flow f by RDBP. It follows $\widehat{y}_f^{d,t}=0$ if $t\notin d$ or $d\ne d'$ by Line 12 of RDBP. Let variable $q(e,f)$ denote the traffic load on link e from flow f by RDBP. The expectation of variable $q(e,f)$ is:

$$\mathbb{E}\left[q(e,f)\right]=\sum_{t\in T^f:e\in t}\sum_{d'\in\widetilde{D}(f)}$$
$$\mathbf{Pr}\left[\widehat{x}_f^{d'}=1\right]r(f)\sum_{d\in\widetilde{D}(f)}\widehat{y}_f^{d,t}$$
$$=\sum_{t\in T^f:e\in t}\sum_{d'\in\widetilde{D}(f)}(\widetilde{x}_f^{d'}\cdot r(f)\cdot\frac{\widetilde{y}_f^{d',t}}{\widetilde{x}_f^{d'}})$$
$$=\sum_{t\in T^f:e\in t}\sum_{d'\in\widetilde{D}(f)}\widetilde{y}_f^{d',t}\cdot r(f)=\widetilde{l}(e,f) \tag{18}$$

Eq. (18) shows that the expected load on link e from flow f by RDBP is the same as that in the optimal solution of linear program LP_1. □

Assume that the minimum capacity of all links is denoted by c^{min}. We define a constant γ as follows:

$$\gamma=\min\{\frac{\widetilde{\lambda}\cdot c^{\min}}{r(f)},f\in\Gamma\} \tag{19}$$

Theorem 7. *The proposed RDBP algorithm guarantees that the total traffic on any link $e\in E$ will not exceed the link capacity by a factor of $\frac{4\log n}{\gamma}+3$.*

Proof. The traffic load of link e after the first step is denoted by $\widetilde{l}(e)$. By the definition, variables $q(e,f)$ with $f\in\Gamma$ are mutually independent. According to Eq. (18), the expected traffic load on link e is:

$$\mathbb{E}\left[\sum_{f\in\Gamma}q(e,f)\right]=\sum_{f\in\Gamma}\mathbb{E}\left[q(e,f)\right]=\sum_{f\in\Gamma}\widetilde{l}(e,f)=\widetilde{l}(e) \tag{20}$$

By the seventh set of inequalities in Eq. (8), we have:

$$\widetilde{l}(e)=\sum_{f\in\Gamma}\sum_{t\in T(f):e\in t}\sum_{d\in D}\widetilde{y}_f^{d,t}\cdot r(f)\le\widetilde{\lambda}\cdot c(e) \tag{21}$$

Combining Eqs. (20), (21) and the definition of γ in Eq. (19), we have:

$$\begin{cases}\frac{q(e,f)\cdot\gamma}{\widetilde{\lambda}\cdot c(e)}\in[0,1]\\ \mathbb{E}\left[\sum_{f\in\Gamma}\frac{q(e,f)\cdot\gamma}{\widetilde{\lambda}\cdot c(e)}\right]\le\gamma.\end{cases} \tag{22}$$

By applying Theorem 1, assume that ρ_2 is an arbitrary positive value. It follows:

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{q(e,f)\cdot\gamma}{\widetilde{\lambda}\cdot c(e)}\ge(1+\rho_2)\gamma\right]\le e^{\frac{-\rho_2^2\gamma}{2+\rho_2}} \tag{23}$$

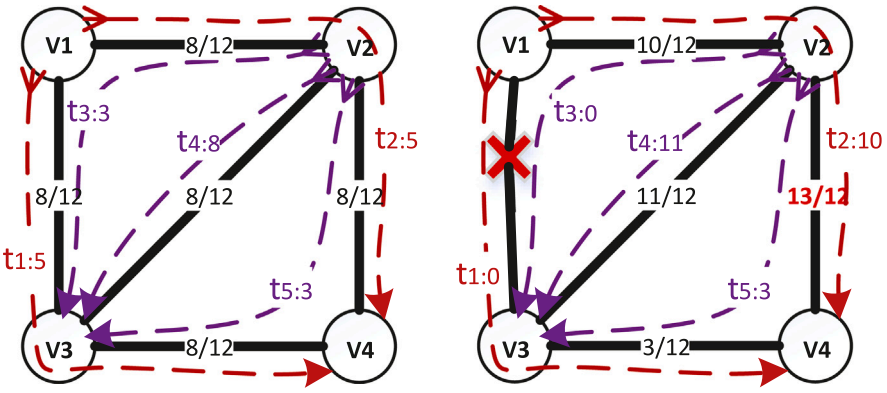


Fig. 2. Illustration of the throughput reduction when link v_1v_3 fails. Black solid lines and color thick lines denote links and flows, respectively. The label x/y (*e.g.*, 8/12) inside each link denote the link load (*i.e.*, 8) and the link capacity (*i.e.*, 12). The label $t_i : z$ (*e.g.*, $t_1 : 5$) inside each flow denote the path id (*e.g.*, t_1) and the traffic amount through this path (*e.g.*, 5). *Left plot*: before link failure; *Right plot*: after link failure.

Now, we assume that

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{q(e,f)}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho_2)\right]\leq e^{\frac{-\rho_2^2\gamma}{2+\rho_2}}\leq\frac{\mathcal{F}}{n^2} \tag{24}$$

where $\mathcal{F}$ is the function of network-related variables (such as the number of switches n, *etc.*.) and $\mathcal{F}\to 0$ when the network size grows.

The solution for Eq. (24) can be expressed as:

$$\rho_2\geq\frac{\log\frac{n^2}{\mathcal{F}}+\sqrt{\log^2\frac{n^2}{\mathcal{F}}+8\gamma\log\frac{n^2}{\mathcal{F}}}}{2\gamma},\quad n\geq 2 \tag{25}$$

Set $\mathcal{F}=\frac{1}{n^2}$. Eq. (24) is transformed into:

$$\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{q(e,f)}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho_2)\right]\leq\frac{1}{n^4},\text{where}\quad\rho_2=\frac{4\log n}{\gamma}+2 \tag{26}$$

By applying Theorem 2, we have,

$$\begin{aligned}&\mathbf{Pr}\left[\bigvee_{e\in E}\sum_{f\in\Gamma}\frac{q(e,f)}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho_2)\right]\\&\leq\sum_{e\in E}\mathbf{Pr}\left[\sum_{f\in\Gamma}\frac{q(e,f)}{\widetilde{\lambda}\cdot c(e)}\geq(1+\rho_2)\right]\\&\leq[\frac{1}{2}n(n-1)]\cdot\frac{1}{n^4}\\&\leq\frac{1}{2}n^2\cdot\frac{1}{n^4}=\frac{1}{2n^2},\quad\rho_2=\frac{4\log n}{\gamma}+2\end{aligned} \tag{27}$$

Note that the third inequality holds, because there are at most $\frac{1}{2}n(n-1)$ links in the network G. Eq. (27) means that the proposed RDBP algorithm can guarantee that the total traffic on any link $e\in E$ will not exceed the fractional solution by a factor of $1+\rho_2=\frac{4\log n}{\gamma}+3$. □

Approximation Factors: According to our analyses, the flow-table size constraint will not be violated by a factor of $\frac{3\log n}{\beta^{min}}+3$, and the link capacity will not be violated by a factor of $\frac{4\log n}{\gamma}+3$ by routing a full percentage of flows on the chosen path set. It means that our proposed RDBP algorithm can achieve the optimal solution, violating the flow-table size constraint by a factor $\frac{3\log n}{\beta^{min}}+3$ and the link capacity constraint by a factor $\frac{4\log n}{\gamma}+3$ at most, which is also called bi-criteria approximation [27].

We should address that, in most situations, the RDBP algorithm can achieve almost the constant bi-criteria approximation. For example, let $\widetilde{\lambda}$ be 0.4 (with a moderate value). Consider a large-scale network with $n=1000$ switches, so that $\log n\approx 10$. The link capacity of today's networks will be a bandwidth of 10 Gbps at least [14]. Observing the practical traffic traces, the maximum flow rate may reach 10 Mbps or 100 Mbps. Under two cases, $\frac{c^{\min}}{r(f)}$ will be 10^3 and 10^2. The approximation factors for the link capacity constraint are 3.1 and 4, respectively. Since β^{min} is usually more than 1000, the approximation factor for the flow-table size constraint is 3.003. In other words, our RDBP algorithm can achieve almost the constant bi-criteria approximation for the PRLF problem in many network situations.

3.3. Throughput preserving reliable routing

The PRLF problem in Eq. (8) only guarantees that each flow will be forwarded through at least one path with high reliability. However, it cannot preserve throughput for a flow when some links/switches fail. In this section, we will study throughput preserving reliable routing scheme for an SDN and give the weight adjustment solution to recover from failures.

3.3.1. Motivation

When a failure event occurs, the traffic on the failed link(s) will be blocked. The controller should migrate these affected flows (not all flows) from the failed paths to other healthy paths, so as to reduce the control overhead and rule update cost. However, it may result in throughput reduction. Here we illustrate the throughput reduction by PRLF after a link failure.

As shown in Fig. 2, an SDN consists of four switches. The capacity of each link is 12. For simplicity, we only consider two flows in the network. One is from v_1 to v_4, denoted as f_1 (red dotted arrow), the other is from v_2 to v_3, denoted as f_2 (purple dotted arrow). The flow rates of f_1 and f_2 are 10 and 14, respectively. f_1 selects two paths t_1 and t_2, and f_2 selects three paths t_3, t_4 and t_5 for forwarding. According to the RDBP algorithm, the load on $\{t_1,t_2,t_3,t_4,t_5\}$ is $\{5,5,3,8,3\}$. At this time, the load of all links is 8, illustrated in the left plot of Fig. 2. When the link v_1v_3 fails, the traffic on t_1 can be migrated to t_2, which will result in congestion on link v_2v_4 and further throughput reduction, as shown in the right part of Fig. 2. In fact, we can allocate the traffic rate on $\{t_1,t_2,t_3,t_4,t_5\}$ to $\{5,5,2,10,2\}$. As a result, when link v_1v_3 fails, traffic on t_1 and t_3 can be migrated to t_2 and t_4, respectively, without link congestion and further throughput reduction.

3.3.2. Algorithm design

We extend our PRLF problem to achieve throughput preserving reliable routing scheme (TPRS) when some links fail. Before the problem definition, we first consider the concept of failure scenario [21], in which one or several links may fail simultaneously. Through long-term observations, the controller knows the candidate failure scenarios, and the probability of each failure scenario in the network. For example, considering a scenario s with only two failed links e_1 and e_2 (and all other links work well), it can be denoted as $s(\{e_1,e_2\})$. The probability of scenario $s(\{e_1,e_2\})$ is $\mathcal{P}_{s(\{e_1,e_2\})}=\mathcal{P}(e_1)\cdot\mathcal{P}(e_2)\prod_{e_i\in E\setminus\{e_1,e_2\}}(1-\mathcal{P}(e_i))$ [21]. The number of scenarios (*i.e.*, $2^{|E|}$) increases exponentially

as the number of links increases. Therefore, it is impossible to consider all scenarios in a certain scale network. Based on the marginal effect [28], we can achieve high throughput and reliability by ignoring some scenarios with very low probability [21]. Specifically, we focus on the scenarios whose failure probability is larger than $1-\alpha$ (*e.g.*, α is 99.9%), which is defined in Eq. (8). We call such a scenario as a *candidate failure scenario*.

We then introduce the throughput preserving reliable routing scheme. It adopts the same paths for each flow as that by RDBP, so that it satisfies the flow-table size constraint on each switch and the reliability requirement for each flow. The final solution of RDBP is denoted as $\hat{z}_f^t$. Specifically, $\hat{z}_f^t$ denotes whether flow f selects path t for routing or not by RDBP. We will reassign the weight of all selected paths for flows.

When a candidate failure scenario occurs, we will adjust the weights of paths for affected flows to migrate the traffic on the failed paths to other healthy paths. Specifically, we set the weight of the path passing through the failed link to 0 and increase the weight of other paths. $A^t(s)$ denotes whether path t works well or not in the candidate scenario s. Variable $w_f^{t_1,t_2}(s)$ denotes the weight of flow f that is migrated from path t_1 to path t_2 when candidate failure scenario s occurs. $u_f^t(s)$ is the final weight of path t for flow f after recovery from candidate failure scenario s. The weight of path for each flow that does not pass the failed links will not be changed. Our objective is to balance the load among all links for all candidate failure scenarios. We give the definition of TPRS in Eq. (28).

$$\min \mathcal{O}$$

$$S.t\begin{cases} y_f^t \le \hat{z}_f^t, & \forall t \subset T^f, f \\ \sum_{t\in T^f} y_f^t = 1, & \forall f \\ \sum_{t_2\in T^f} w_f^{t_1,t_2}(s) = y_f{}^{t_1}, & \forall t_1,t_2 \subset T^f, s, f \\ w_f^{t_1,t_1}(s) = y_f^{t_1}\cdot A^{t_1}(s), & \forall t_1 \subset T^f, s, f \\ u_f^{t_1}(s) = y_f^{t_1} + \sum_{t_2\in T^f\setminus t_1} w_f^{t_2,t_1}(s) & \\ \quad - \sum_{t_2\in T^f\setminus t_1} w_f^{t_1,t_2}(s), & \forall t_1,t_2 \subset T^f, s, f \\ \sum_{f\in F}\sum_{t\in T^f: e\in t} u_f^t(s)\cdot r(f) & \\ \le \mathcal{O}\cdot c(e), & \forall e, s \\ y_f^t, w_f^{t_1,t_2}(s) \in [0,1], & \forall t_1,t_2,s,f \end{cases} \tag{28}$$

The first set of inequations denotes the weight assignment for each selected path by RDBP. The second set of equations tells that the traffic of each flow will be forwarded from source to destination. The third to the fifth sets of equalities compute the final weight of path t for flow f after recovery from failure scenario s. It is noted that, for the flow that does not pass the failed links, the weights of its selected paths will not be changed, which is shown in the fourth set of inequations. The sixth set of inequalities denotes the link capacity constraint after recovery from different failure scenarios. The objective of TPRS is to balance the load among all links for all candidate failure scenarios, that is, $\min\ \mathcal{O}$.

Eq. (28) is a linear program and can be solved in polynomial time with a linear program solver, *e.g.*, pulp [25]. By solving Eq. (28), we can obtain the weight assignment and path weight adjustment solution for each failure scenario s (*i.e.*, $w_f^{t_1,t_2}(s)$) to recover from different failure scenarios without throughput reduction. We set the weight assignment in the network and store each failure scenario along with the corresponding path weight adjustment solution in the database. In case of link failures, the controller can detect the failure scenario, query the database, obtain the path weight adjustment solution, and adjust the selected paths' weights of each OD-pair accordingly. For convenience, we denote this algorithm as RDBP-T. If the failure scenario is an event with low probability, the controller cannot find the path weight adjustment solution in the database. As a result, the controller will adopt a reactive solution CALFR [8] to deal with this failure event.

3.4. Discussion

We assume that failures are independent when we calculate the failure probability of each path in Section 2.2. In fact, failure events of different links are related. For example, if a switch fails, all links connected to this switch will be down. We take this relationship of different link failure events into account and reconsider the failure probability of each path. To this end, we should extend the problem definition to the case of failure dependence. We redefine $\mathcal{P}(e)$, which denotes the probability that link e fails when its connected switches work well. The failure probability of each switch v is denoted as $\mathcal{P}(v)$. For each path t, its failure probability can be expressed as

$$\mathcal{P}'(t) = 1 - \prod_{e\in t}(1-\mathcal{P}(e))\prod_{v\in t}(1-\mathcal{P}(v)) \tag{29}$$

Some paths may share at least one switch. Besides the failure of shared link affects the reliability of multi path, the failure of the shared switches will also affect it, which makes the computation of failure probability of a path set more difficult. To this end, we define V^d as the set of switches which are shared by multiple paths in d. We use η^d to keep all the possible combinations of shared switches in V^d. Obviously, if $|V^d| = l$, $|\eta^d| = 2^l$. For each shared switch set $C^d \in \eta^d$, we define $\mathcal{P}(C^d)$ as the probability that all switches in set C^d fail and other switches in $V^d \setminus C^d$ work well. That is,

$$\mathcal{P}(C^d) = \prod_{v\in C^d}\mathcal{P}(v)\cdot\prod_{v\in V^d\setminus C^d}(1-\mathcal{P}(v)) \tag{30}$$

For ease of calculation, we define $\mathcal{P}(t|C^d)$ as the failure probability of path t under the condition that only all switches in set C^d fail. That is

$$\mathcal{P}(t|C^d) = \begin{cases} 1 & ,\text{if } v\in t, \exists v\in C^d \\ 1-\prod_{v\in t\setminus V^d}(1-\mathcal{P}(v)) & ,\text{if } v\notin t, \forall v\in C^d \end{cases} \tag{31}$$

We then define $\mathcal{P}(d|C^d)$ as the failure probability of path set d under the condition that only all switches in set C^d fail according to the multiplication formula of probability [22].

$$\mathcal{P}(d|C^d) = \prod_{t\in d}\mathcal{P}(t|C^d) \tag{32}$$

According to the total probability theorem [22] and Eq. (6), we obtain the failure probability of path set d as

$$\mathcal{P}'(d) = \mathcal{P}(d)\cdot\sum_{C^d\in\eta^d}\mathcal{P}(C^d)\mathcal{P}(d|C^d) \tag{33}$$

We finally denote the reliability probability $\mathcal{R}'(d)$ of a path set d as $1-\mathcal{P}'(d)$.

4. Performance evaluation

In this section, we first introduce the metrics and benchmarks for performance comparison (Section 4.1). We deploy an SDN on a small testbed with OVS and evaluate the effectiveness of our proposed algorithm on small-scale networks (Section 4.2). We finally compare our proposed solution with some benchmarks through simulations on large-scale networks (Section 4.3).

4.1. Performance metrics and benchmarks

We adopt seven main metrics for performance evaluation: (1) the maximum link load ratio (MLR); (2) the network throughput factor (NTF); (3) the maximum number of required flow entries among all switches (MNE); (4) the reliability satisfied ratio of all OD-pairs (RSR); (5) the failure recovery delay (FRD); (6) the running time (RNT); (7) the control overhead (CTO).

To measure the network performance, we adopt the maximum link load ratio (MLR) and network throughput factor (NTF) as two metrics. We measure the traffic load $l(e)$ of each link e during system

running, and the maximum link load ratio is defined as: $MLR = \max\{l(e)/c(e), e \in E\}$. The lower MLR means better load balancing. When the flow rate is increasing, some links may be congested and only fractional traffic of each flow can be forwarded to the destination to avoid congestion. Each flow f can be forwarded traffic amount of $\omega \cdot r(f)$ from source to destination, where ω is the network throughput factor (NTF), with $0 < \omega \leq 1$. We measure the maximum number of required flow entries among all switches (MNE). The smaller MNE means less resource consumption on a switch. Each OD-pair will be allocated one or several appropriate paths. According to the statistical information [21], we calculate the failure probability of the selected path set for each flow by Eqs. (1)–(6). We say that an OD-pair satisfies the reliability requirement if the reliability probability of this OD-pair exceeds a threshold α (*e.g.*, $\alpha = 99.9\%$). We adopt the reliability satisfied ratio of all OD-pairs (RSR), which represents the proportion of the number of OD-pairs that satisfy the reliability requirement to the total number of OD-pairs. The larger RSR means higher reliability of an SDN. We also measure the duration of any packet loss after a single link failure (FRD) for all benchmarks. In addition, we measure the running time (RNT) of the RDBP algorithm.

When a failure event occurs, the controller needs to update the flow/group-table entries on some switches so as to successfully forward traffic after detecting the failure, which results in additional control overhead. We measure the number of control commands [29] (*e.g.*, OFPFlowMod, OFPGroupMod messages) to indicate control overhead (CTO) as a metric. We also measure the MLR and NTF after recovery from a failure event.

The proposed RDBP solution needs to input the reliability probability of each path set in advance. To this end, we can get the working time and failure time of each link through long-time measurement, and then calculate the failure probability of each link. Based on this, we can calculate the reliability probability of each path set. In order to ensure the accuracy of the calculated reliability probability, we update the reliability probability of the path regularly according to the latest measurement results [21], so as to ensure the effectiveness of the RDBP algorithm.

To evaluate the performance of our proposed RDBP solution, we choose two related benchmarks.

1. The first one is BLR [13], which is a proactive failure recovery method. Each flow will select a primary path and pass through this path when no failure occurs. It also determines a link-disjoint backup path. When any link on the primary path fails, the affected traffic will be routed along the backup path without disturbing the controller [13], which is implemented by using the fast failover function [29] at the group table of each switch.
2. The second one is CALFR [8], which is a reactive failure recovery method. After detecting a failure event, the controller needs to explore a backup path for each affected flow. As a result, some flow-table entries for backup paths will be installed to recover from failure. Specifically, CALFR adopts flexible flow aggregation to further reduce the use of flow-table entries.

4.2. System implementation on OVS platform

With the development of network virtualization technology and cloud computing, virtual switch has attracted the attention of many major manufacturers (*e.g.*, Amazon) and open source projects (*e.g.*, Openstack). Open Virtual Switch (OVS) [30], with its good support for OpenFlow protocol, has become the most popular virtual switch in the field of SDN. We implement the RDBP, BLR and CALFR benchmarks on a small-scale OVS testbed. The OVS platform is comprised of three parts: an SDN controller, 7 SDN-enabled virtual switches and 6 hosts, as shown in Fig. 3. This topology is called Sanren [31] from South Africa. The bandwidth of each link is set to 1 Gbit/s. As we focus on the performance of the data plane and the controller does not participate

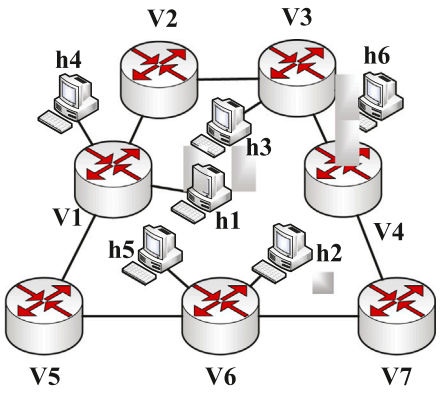


Fig. 3. OVS testbed topology.

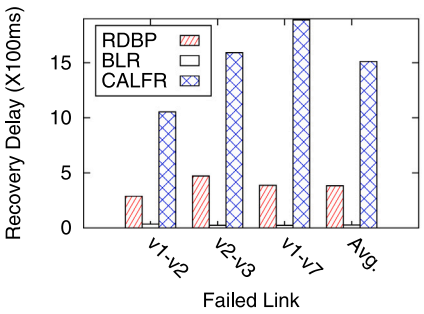


Fig. 4. Failure recovery delay vs. failed link.

in data forwarding, we omit the controller in Fig. 3 for simplicity. For more convenient deployment and management of the network, we adopt the virtualization technology for system implementation. Specifically, all 7 SDN-enabled switches are implemented using Open Virtual Switches (version 2.7.2 [30]), and each host is implemented using the kernel-based virtual machine (KVM). Each OVS and the connected KVMs are implemented on a server with a CORE i5-3470 processor and 8 GB of RAM. For example, $\{v_1, h_1, h_4\}$, $\{v_2\}$, $\{v_3, h_3, h_6\}$, $\{v_4\}$, $\{v_5\}$, $\{v_6, h_2, h_5\}$ and $\{v_7\}$ are run on 7 servers, respectively. Besides, we use Ryu [32] that supports the OpenFlow v1.3 standard as the controller software running on another server with a CORE i7-8700K processor and 32 GB of RAM. We execute each test 20 times and average the numerical results. We use "ovs-ofctl mod-port switch port down" command to turn down a port on the OVS, so that the related link fails.

The first set of testings observes the failure recovery delay and the number of control commands. Let host h_1 send ICMP packets to h_3 as fast as possible by using "ping -f" command. We measure the time during which no response packet is received by the sender h_1 when some link fails. The results are shown in Fig. 4. Since CALFR needs to recalculate the backup path(s) and install rules to the data plane, the failure recovery delay of CALFR is much larger than that of the other two solutions. Because the controller only needs to modify the weight of two group-table entries in RDBP, the failure recovery delay of RDBP is less than that of CALFR. As BLR is a proactive solution, the data plane can automatically switch traffic to backup path(s) without interfering with the controller. The failure recovery delay of BLR is smaller than that of the other two benchmarks. Specifically, the failure recovery delay of RDBP, BLR and CALFR is 287 ms, 35 ms and 1052 ms, respectively when we break link v_1v_2 in Fig. 3. That is, our proposed RDBP algorithm reduces the failure recovery delay by 72.7% compared with CALFR. When a failure event occurs, the controller will install backup paths by sending control commands (*i.e.*, OFPFlowMod messages) to some switches in CALFR and modify the weight of group

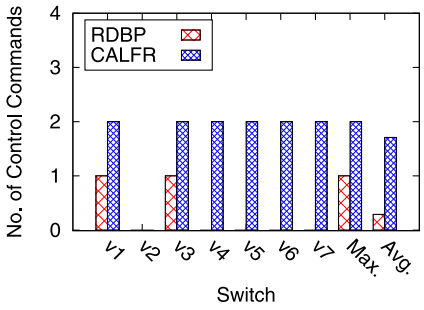


Fig. 5. Number of control commands on each switch.

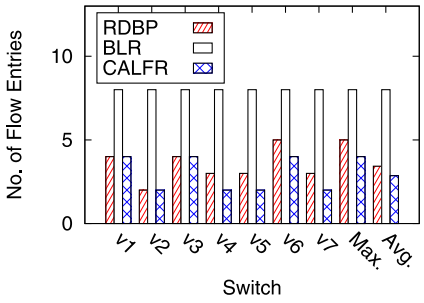


Fig. 6. Number of required flow entries on each switch.

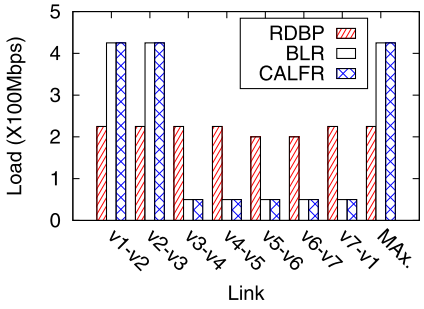


Fig. 7. Link load.

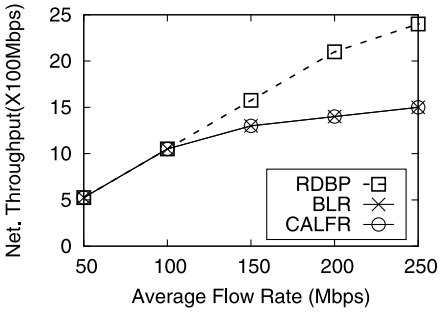


Fig. 8. Network throughput vs. average flow rate.

entries by sending control commands (*i.e.*, OFPGroupMod messages) to a few switches in RDBP. We count the number of control commands on each switch and the results are shown in Fig. 5. The average number of control commands on each switch in RDBP and CALFR is 0.29 and 1.71, respectively. Our RDBP solution achieves a lower control overhead than CALFR.

The second testing observes the number of required flow entries on each switch without link failure as shown in Fig. 6. Since BLR is a proactive solution, both primary paths and backup paths need to be installed in advance. As a result, BLR requires more flow entries on each switch than CALFR and RDBP. As CALFR is a reactive solution, only the primary paths need to be installed. CALFR requires fewer flow entries on each switch than the other two solutions. Specifically, the maximum number of required flow entries among all switches in RDBP, BLR and CALFR is 5, 8 and 4, respectively.

The third set of testings shows the link traffic load and network throughput in Figs. 7–8. Each host will send UDP traffic to all the other hosts, and the average flow rate is 100 Mbps with 2–8 distribution [33]. Since BLR and CALFR choose the same primary path for each flow, the load on each link under these two solutions is the same before a failure event occurs, in which v_1v_2 (or v_2v_3) burdens the maximum load (*i.e.*, 425 Mbps) among all links by BLR and CALFR. Because RDBP uses a multi-path forwarding mechanism, the flow traffic is divided on multiple paths as evenly as possible, and the load among all links is very balanced. The maximum link load by RDBP is 225 Mbps, which is far less than that by BLR and CALFR. Fig. 8 shows that the network throughput increases with the increasing flow rate for three solutions. However, the network throughput of BLR and CALFR increases more slowly than that of RDBP as the average flow rate increases. Our proposed RDBP algorithm achieves higher network throughput than BLR and CALFR when the average flow rate is more than 100 Mbps. Specifically, the network throughput of RDBP, BLR and CALFR is 2.4 Gbps, 1.5 Gbps and 1.5 Gbps, respectively when the average flow rate is 250 Mbps. RDBP improves network throughput by 60.0% compared with BLR as well as CALFR. In a word, Figs. 7 and 8 show that our RDBP algorithm can achieve better routing performance than the other two benchmarks.

According to the testing results, we can make the following conclusions. RDBP can significantly improve traffic throughput by 60.0% compared with BLR and CALFR. Moreover, RDBP reduces the maximum number of required flow entries by 37.5% compared with BLR, and reduces the failure recovery delay and control overhead by 72.7% and 83.0% compared with CALFR.

4.3. Simulation results on large-scale networks

Network Topologies: We choose two kinds of typical topologies for large-scale network simulations. The first one is a data center topology called Fat-tree [34], denoted as (a), which contains 80 switches (including 16 core switches, 32 aggregation switches, and 32 edge switches) as well as 128 servers (each edge switch connecting to 4 servers). The link capacity is 1/5/20 Gbit/s for the link connecting host and edge switch/ link connecting edge switch and aggregate switch/ link connecting aggregate switch and core switch, respectively in topology (a). The second topology is a campus network, denoted as (b), which contains 100 switches (including 17 core switches and 83 edge switches) as well as 475 servers from Monash university [35]. The link capacity is 1 Gbit/s for all links in topology (b). These two topologies represent various networks with different features. For example, Fat-tree (a) is structured and designed for data center networks, while the campus topology (b) is unstructured and for local area networks. By default, we set an RYU controller [32] in both topologies for management. The link capacity between host and edge switch is 1 Gbps for both topologies in our simulations. Moreover, Less than 20% of the top-ranked flows may be responsible for more than 80% of the total traffic [33]. Thus, we allocate the size for each flow according to this 2–8 distribution. We set the reliability threshold as 99.9% by default. We use link down command to cause the link failure in the test. The failure probability of the link conforms to the Gaussian distribution, and its average value is 0.1%. We execute each simulation 100 times and average the numerical results.

We run four groups of simulations to check the performance of the proposed RDBP algorithm.

Important parameter settings for RDBP: The first set of simulations evaluates the effect of the number of candidate path sets on the performance of RDBP. We have introduced the concept of candidate path sets in the problem definition of PRLF (*i.e.*, Eq. (8)). For each flow

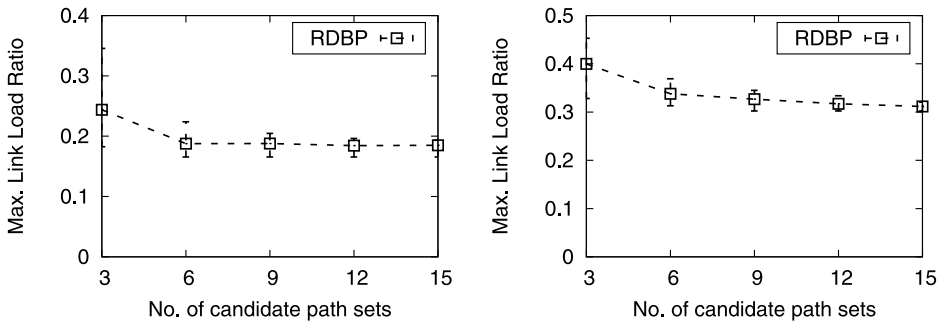


Fig. 9. Maximum link load ratio vs. Number of candidate path sets. *Left plot*: Topology (a); *right plot*: Topology (b).

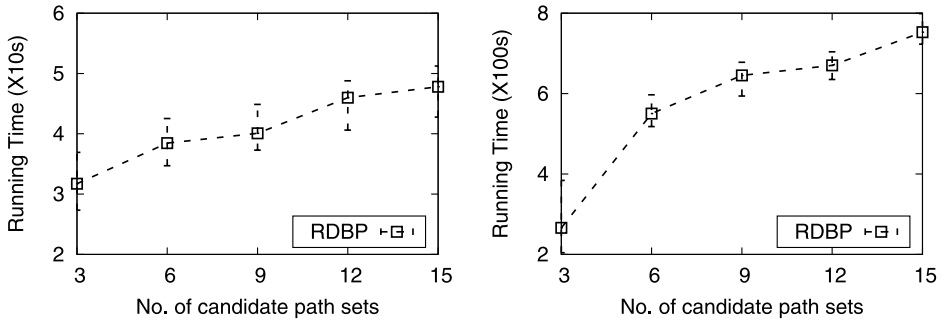


Fig. 10. Running time vs. number of candidate path sets. *Left plot*: Topology (a); *right plot*: Topology (b).

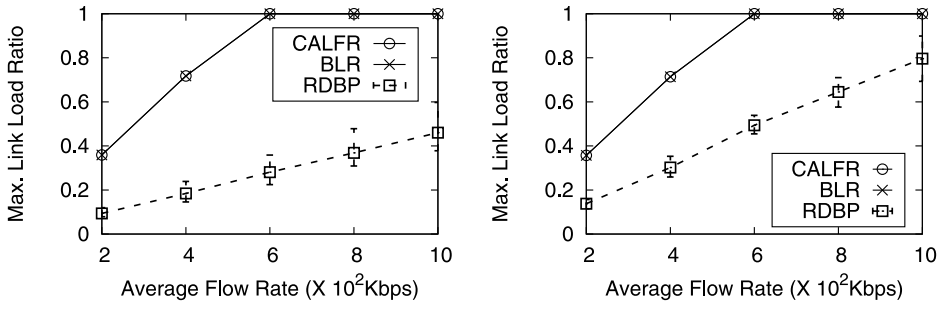


Fig. 11. Maximum link load ratio vs. average flow rate. *Left plot*: Topology (a); *right plot*: Topology (b).

f, the number of candidate path sets may reach $2^{|T^f|}-1$, where $|T^f|$ is the number of candidate paths for flow f. We set 4 paths for each flow, and there are 15 candidate path sets at most. We test the maximum link load ratio and running time by changing the number of candidate path sets from 3 to 15. Fig. 9 shows the MLR of RDBP in two topologies. The MLR of RDBP decreases with the increasing number of candidate path sets. However, the change is small especially when the number of path sets is more than 6. Specifically, the MLR of RDBP is 0.188 and 0.185, respectively when we choose 6 and 15 candidate path sets in topology (a). Therefore, it is reasonable to choose only a few path sets for RDBP. We then test the running time of RDBP by changing the number of path sets for each flow. Fig. 10 shows that the RNT of RDBP increases with the increasing number of candidate path sets. Combining Figs. 9 and 10, we can choose the appropriate number of path sets (*e.g.*, 6) by the tradeoff between the MLR and RNT in the following simulations.

Network performance before link failure recovery: The second set of simulations evaluates the maximum link load ratio (MLR) and network throughput factor (NTF) before the appearance of failure event by changing the average flow rate from 200 kbps to 1000 kbps in both topologies. The MLRs of all three algorithms increase with the increasing flow rate. As we choose the same primary path for each OD-pair under BLR and CALFR, their MLRs are the same all the time. The MLR of RDBP is less than that of BLR and CALFR all the time. When the average flow rate exceeds 600 kbps in both topologies, some links are congested by BLR and CALFR, while our proposed RDBP algorithm still works well. It means that the proposed RDBP algorithm can achieve better load balancing than the other two benchmarks. Specifically, when the average flow rate is 400 kbps, the MLRs of RDBP, BLR and CALFR are 0.18, 0.71 and 0.71, respectively by the left plot of Fig. 11. That is, the proposed RDBP algorithm reduces MLR by 74.6% compared with BLR and CALFR.

We then observe the network throughput factor (NTF) before the appearance of the failure event by changing the average flow rate in two topologies, and the simulation results are shown in Fig. 12. The NTFs of all three algorithms decrease with the increasing flow rate. When we increase the flow rate, the bandwidth of some links will be exhausted, which leads to link congestion. At this time, the network throughput is reduced to avoid congestion. The NTFs of all three methods are 1 when the average flow rate is less than 400kbps in both topologies. When the average flow rate exceeds 600 kbps, only RDBP can achieve the throughput factor of 1. However, it is not the case for BLR and CALFR. Specifically, the NTFs of RDBP, BLR and CALFR are 1, 0.70 and 0.70, respectively when the average flow rate is 800 kbps in topology (a). That is, our RDBP solution improves NTF by 30.3% compared with BLR and CALFR.

Overhead for link failure recovery: The third set of simulations evaluates the maximum number of required flow entries (MNE) and

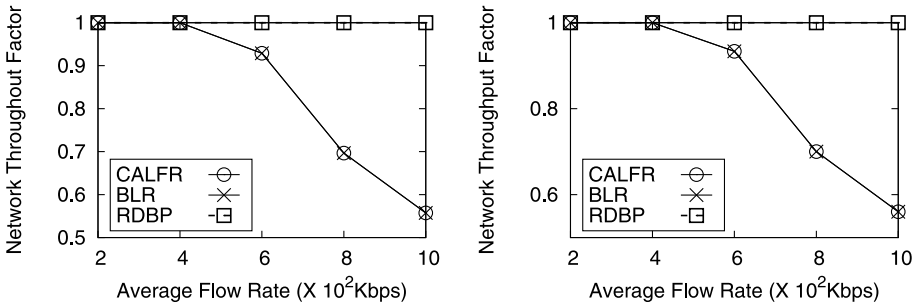


Fig. 12. Network throughput factor vs. average flow rate. *Left plot*: Topology (a); *right plot*: Topology (b).

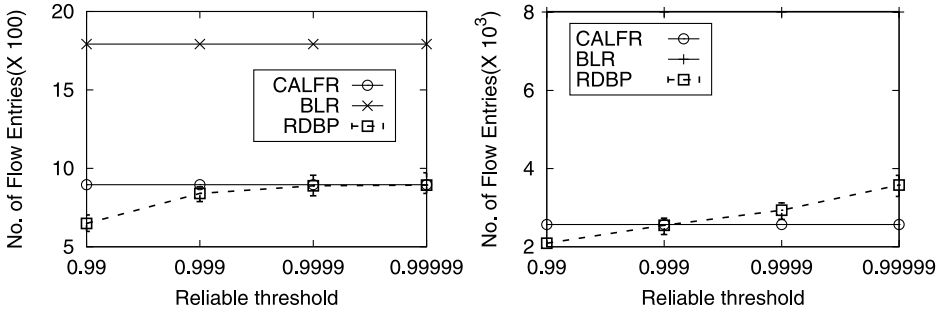


Fig. 13. Maximum number of required flow entries vs. reliability threshold α. *Left plot*: Topology (a); *right plot*: Topo. (b).

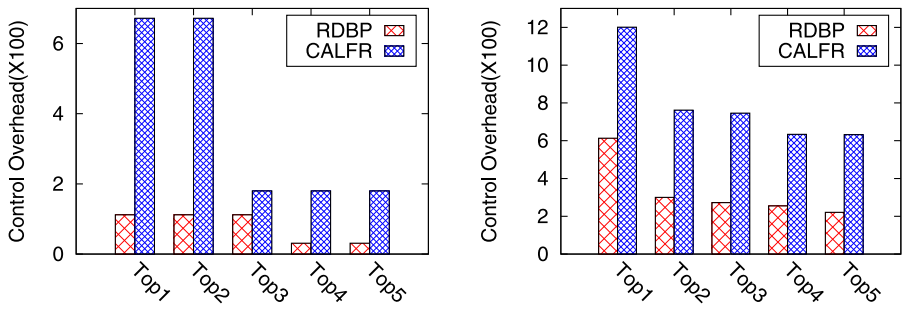


Fig. 14. Control overhead over different failure scenarios. *Left plot*: Topology (a); *right plot*: Topology (b).

control overhead (CTO). One or several appropriate paths will be allocated to each OD-pair. According to the statistical information [21], we calculate the failure probability of the selected path set for each flow by Eqs. (1)–(6). We say that an OD-pair satisfies the reliability requirement if the reliability probability of this OD-pair exceeds a threshold α (*e.g.*, $\alpha = 99.9\%$). Note that α is the minimum reliability requirement for each flow. We measure the maximum number of required flow entries (MNE) by changing the reliability threshold α in both topologies, as shown in Fig. 13, which indicates the overhead in the data plane. RDBP can guarantee that the reliability probability of each flow should exceed the reliability threshold α. The larger value of α means better robustness of the proposed RDBP algorithm. It is obvious that the MNE of RDBP increases with the increasing reliability threshold α in both topologies. With a larger threshold, each OD-pair will use more paths to forward traffic and each switch will consume more flow entries for path installment. As BLR is a proactive method that deploys all backup paths in advance, BLR achieves higher MNE than the other two solutions. As CALFR is a reactive method, it consumes a small amount of flow entries on all switches in the network. However, it does not guarantee that the number of required flow entries on each switch is balanced. Some switches may suffer from severe flow entries consumption by CALFR, which results in high MNE in some situations. Our RDBP solution limits flow entries consumption on each switch. The MNE of RDBP is less than that of CALFR in topology (a) all the time, but we obtain the opposite conclusion in topology (b) when the reliability threshold α is larger than 99.99%. Specifically, the MNEs of RDBP, BLR and CALFR are 840, 1792 and 896 in topology (a) when the reliability threshold α is 99.9%. That is, our RDBP algorithm reduces MNE by 53.1% and 6.25% compared with BLR and CALFR.

When the failure event occurs, the controller needs to adjust the flows' routes to ensure network reliability, which will incur extra control overhead. We then observe the control overhead (CTO or the number of control commands) when different failure scenarios appear in both topologies. The CTO of BLR is always 0, as it is a proactive failure recovery solution. Because CALFR is a reactive solution, it needs to compute a backup path for each affected flow and install rules at the data plane, which results in significant control overhead. However, in RDBP, the controller needs to only modify the weight on a few switches where multiple paths of the same flow intersect, which will cause lightweight control overhead. So the CTO of RDBP is less than that of CALFR. We test the CTOs of RDBP and CALFR when different failure scenarios appear in both topologies and list the maximum five CTO values among all failure scenarios in Fig. 14. Specifically, the maximum CTOs of RDBP and CALFR are 112 and 672, respectively in topology (a). Our proposed RDBP algorithm reduces the control overhead by 83.3% compared with CALFR.

Network performance after link failure recovery: The fourth set of simulations evaluates the robustness of all benchmarks, including the reliability satisfied ratio of all OD-pairs (RSR), maximum link load ratio (MLR) and the network throughput (NTF) after recovery

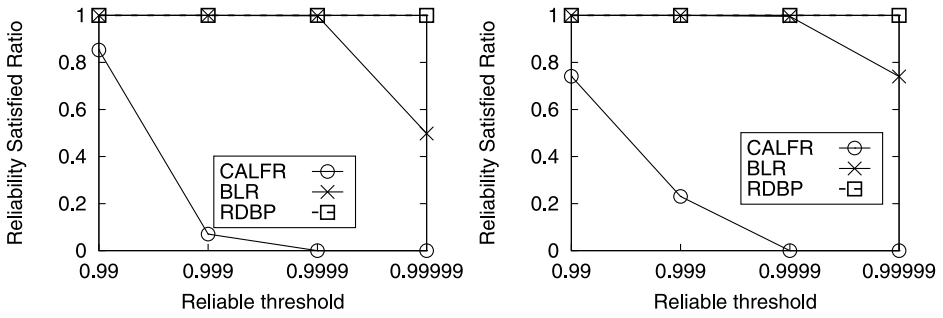


Fig. 15. OD-pair available ratio vs. reliability threshold. *Left plot*: Topology (a); *right plot*: Topology (b).

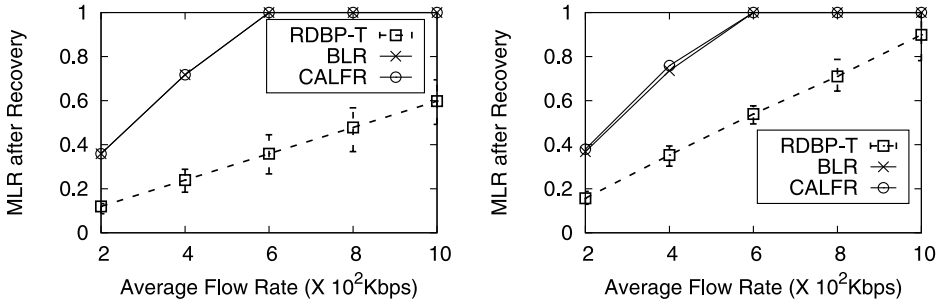


Fig. 16. Max. link load ratio after failure recovery vs. average flow rate. *Left plot*: Topology (a); *right plot*: Topology (b).

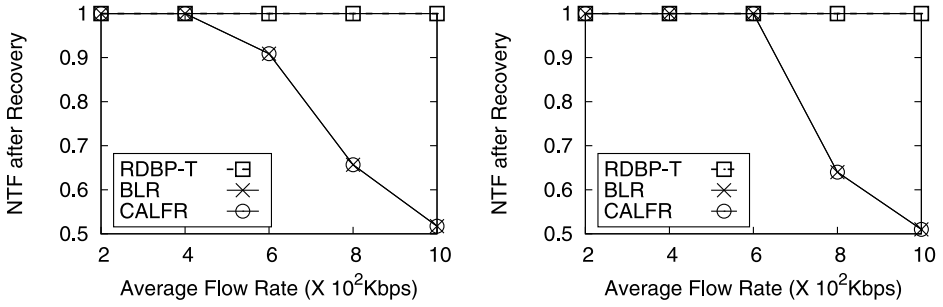


Fig. 17. NTF after failure recovery vs. average flow rate. *Left plot*: Topology (a); *right plot*: Topology (b).

from failure scenarios. We observe the RSR by changing the reliability threshold α in both topologies, and the simulation results are shown in Fig. 15. When we increase the reliability threshold, the RSR of RDBP is always 100%. However, the RSRs of BLR and CALFR decrease with the increase of reliability threshold α in both topologies. When the reliability threshold is 99.9% in topology (a), only 7.1% of OD-pairs can meet the reliability threshold by CALFR, and all OD-pairs can meet the reliability threshold by BLR and RDBP. When the reliability threshold is 99.999% in topology (a), the RSR of BLR falls to 49.8% and will further decline with the increase of reliability threshold α, which shows higher reliability of RDBP compared with BLR and CALFR. In a word, the proposed RDBP algorithm increases the RSR by 12.6% and 76.9% compared with BLR and CALFR, respectively. We evaluate the MLR and NTF after recovery from different failure scenarios in Figs. 16 and 17. When some links fail, the affected flow can rapidly work well by RDBP-T. Fig. 16 shows MLR after recovery from failure by changing the average flow rate in both topologies. We break some links and measure the MLR after failure recovery. MLRs of all three solutions increase with the increasing flow rate. The MLR of RDBP-T is lower than that of BLR and CALFR all the time. When the average flow rate exceeds 600kbps in both topologies (a) and (b), MLRs of BLR and CALFR achieve 1, which means congestion on some links. At this time, our proposed RDBP-T algorithm still works well. As a result, RDBP-T can achieve better load balancing after failure recovery compared with others. Specifically, the MLRs of RDBP-T, BLR and CALFR are 0.23, 0.72 and 0.72, respectively when the average flow rate is 400kbps in topology (a). That is, our RDBP-T algorithm reduces MLR by 68.1% compared with BLR and CALFR. We then test the achievable throughput factor after the network recovers from all candidate failure scenarios by changing the average flow rate in both topologies. The simulation results are shown in Fig. 17. As RDBP-T is a throughput preserving solution, the NTF of RDBP-T is always 1. However, it is not the case for BLR and CALFR. In fact, when the average flow rate is more than 400 Kbps, the NTFs of BLR and CALFR are similar and decrease with the increase of average flow rate in topology (a), which means that BLR and CALFR should reduce throughput to avoid congestion. Specifically, when the average flow rate is 1000 Kbps, the NTFs of RDBP-T, BLR and CALFR are 1, 0.52 and 0.52, respectively in topology (a). RDBP-T improves the NTF by 48.0% compared with others after failure recovery.

We also observe the maximum link load ratio (MLR) by changing the average link failure probability from 0.1% to 0.9% in topology (a), and the simulation results are shown in Fig. 18. With the increase of the average link failure probability, the failure probability of each path will increase. At this time, due to the limited flow table on each switch, some suitable paths cannot be selected. As a result, the MLR of RDBP increases with the increasing average link failure probability. Specifically, the MLR of RDBP is 23.9% and 37.3% when the average link failure probability is 0.1% and 0.5%, respectively.

We can make some conclusions according to the simulation results. First, by Figs. 11–12, our proposed RDBP algorithm can reduce the

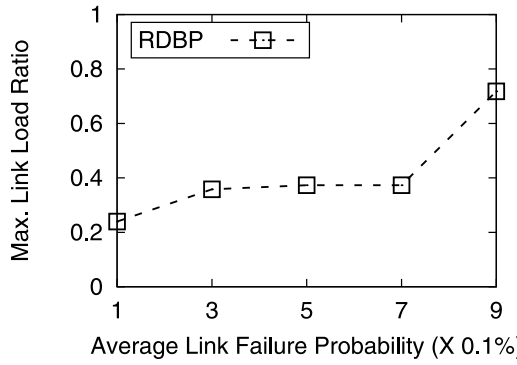


Fig. 18. Maximum link load ratio vs. average link failure probability in topology (a).

MLR by 74.6% and improve the NTF by 30.3% compared with BLR and CALFR without failure. Second, by Fig. 13, RDBP can make more efficient use of flow entries than BLR. Specifically, RDBP reduces MNE by 53.1% compared with BLR. According to Fig. 14, when some links fail, RDBP can reduce the control overhead by 83.3% compared with the reactive method (*i.e.*, CALFR) to fulfill failure recovery. Third, by Fig. 15, we can find out that RDBP can achieve higher path reliability, compared with BLR and CALFR. Finally, by Figs. 16–17, RDBP-T reduces the MLR by 68.1% and improves the traffic throughput by 48.0% compared with BLR and CALFR after recovery from failure scenarios. The simulation results under large-scale topologies are consistent with the testing results at the SDN testbed. Thus, RDBP can achieve better network performance with less resource consumption (*i.e.*, TCAM) and lower failure recovery delay.

5. Related work

Since software defined network can exploit a global view of the whole network and easy to perform global updates by the central controller, it has been favored by many traffic engineering for WAN recently (*e.g.*, B4 [2], SWAN [36] and FFC [20]). Prior work optimizing the bandwidth allocation by leveraging re-configurable optical devices [37,38], under inaccurate knowledge of traffic demands [19,39], but they all disregard the failure event. Suchara et al. [18] achieve failure recovery by re-allocate traffic across multiple paths after failures occur, but ignore failure probabilities. Inspired by financial risk theory, Teavar [21] explicitly accounts for the likelihood of different failure events and adjusts the bandwidth for each flow to minimize a formal notion of risk to an acceptable level as well as improve the network throughput.

A crucial challenge faced by TE is how to react to the presence of underlying link/node failures. Existing solutions to recover underlying link/node failures can be divided into 2 categories reactive strategy and proactive strategy. The difference between these two strategies is obvious. When some links fail in the network, the reactive strategy installs backup paths for affected flows when detecting the failure. Cheng et al. [8] design a new resilience approach to balance failure recovery time and forwarding entries occupation, it aggregates all traffic flows assigned to the same backup path into a "big" flow in order to reduce the number of reconfigured forwarding rules. Wang et al. [9] define the link failure problem in SDN and propose a 2-stage algorithm, rapid connectivity restoration in an acceptable time(50s), and backup path adjustment to reallocate network resources based on bandwidth and delay requirements. It can reduce the recovery time with QoS requirements. Since the controller needs to participate in failure recovery, reactive strategy will cause real-time controller overhead. In the meantime, take action after failure occurs can cause an impossible-to-ignore recovery time.

Proactive strategy recovers the failed network before the failure occurs by deploying additional alternate paths in advance. The underlying failure will be recovered rapidly without the intervention of the controller. Chu et al. [11] aim to solve the resilience issue in a hybrid SDN network where SDN switch and traditional IP router coexist, it redirects traffic on the failed link to a designated SDN switch through IP tunnels. This designated SDN switch decides how to route the flow to appropriate next hop. After that the traffic can be further delivered to its destination. This work deploys SDN switches as little as possible to guarantee traffic reachability in the presence of any single link failure. Li et al. [12] slightly modified [11] by replacing some SDN switches with traditional switches. It uses traditional switches as designated switches only when necessary to further reduce the number of SDN switches needed. Mohan et al. [13] develop two proactive rerouting algorithms (*i.e.* FLR and BLR) to react to underlying failure events. FLR computes a set of backup paths for each primary path, each path in the set is corresponds to one link on the primary path. BLR reroutes the affected flows backward from the failed link to the source node and then flows to the destination along a link-node-disjoint path, which is computationally simpler than FLR.

6. Conclusion

In this paper, we have introduced the reliable flow routing problem for better dealing with failure event. We have defined the RLFR problem for minimizing the maximum load of all links with both reliability and flow-table size constraints in SDNs. We propose a rounding-based algorithm RDBP for this problem and analyze its approximation performance. We further consider how to implement the throughput preserving reliable routing scheme for each OD-pair. We have implemented the RDBP algorithm on an OVS testbed. The extensive simulation results show high efficiency of our proposed algorithms.

CRediT authorship contribution statement

Xuwei Yang: Conceptualization, Methodology, Writing - review & editing, Software. **Hongli Xu:** Conceptualization, Methodology, Writing - review & editing, Supervision. **Jianchun Liu:** Investigation, Software. **Chen Qian:** Conceptualization. **Xingpeng Fan:** Methodology, Software. **He Huang:** Methodology. **Haibo Wang:** Investigation, Methodology.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This article was supported in part by the National Science Foundation of China (NSFC) under Grants 61822210, 61936015, and U1709217; and in part by Anhui Initiative in Quantum Information Technologies under Grant AHY150300; and in part by the Fundamental Research Funds for the Central Universities under Grant WK5290000001.

Appendix. NP hardness of PRLF

To show the NP-hardness, we first give the following definition.

Definition 1 (*Identical Parallel Machines Scheduling (IPMS) Problem [40]*)**.** Given m parallel machines and n independent jobs, each job is to be assigned to one of the machines that have identical processing speed. Thus, every job will take the same amount of processing time on each machine. The objective is to schedule jobs on suitable machines to minimize the makespan.

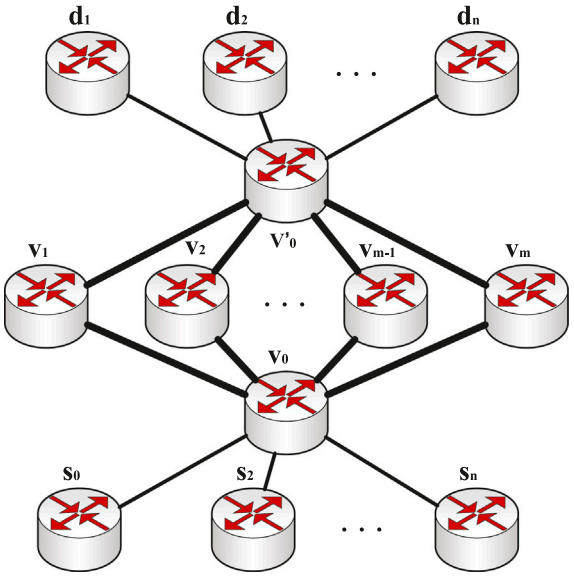


Fig. 19. A special example of the PRLF problem.

Proof. Since IPMS is NP-hardness, We prove the NP-hardness by showing that the IPMS [40] is a special case of the PRLF problem. We consider an arbitrary IPMS instant $\mathcal{S}$. There are a set of m machines and a set of n jobs. Moreover, the processing time of job i on each machine is denoted as t_i.

Next, we consider a special example of the PRLF problem. Given a network topology as shown in Fig. 19. There are n ingress switches denoted as s_i with $i = 1, 2, \ldots, n$ and n egress switches denoted as d_i with $i = 1, 2, \ldots, n$. Each flow is routed from s_i to d_i, with $i = 1, 2, \ldots, n$. A switch v_0 is connected to all ingress switches and another switch v_0' is connected to all egress switches. Moreover, there are m switches denoted as v_j with $j = 1, 2, \ldots, m$ that connect to both v_0 and v_0'. The capacity of link v_0v_j and v_jv_0' are set as c_0 and ∞, respectively, where c_0 is constant. Assume that the failure probability of all paths from s_i to d_i are far less than the threshold $1-\alpha$, and each available path set contains only one path. In addition, assume that switch v_0 has at most n flow entries. So we will select only one path for each OD pair (s_i, d_i) to forward its traffic $t_i \cdot c_0$, so as to minimize the maximum traffic load ratio in a network. Thus, we regard each flow from s_i to d_i and each link v_0v_j as a job i and machine j, respectively. Moreover, the processing time for job i is $\frac{t_i \cdot c_0}{c_0} = t_i$. This is just the IPMS instant $\mathcal{S}$. As a result, each IPMS instance is a special instant of PRLF, which shows the NP-hardness of the PRLF problem. □

References

[1] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven WAN, in: ACM SIGCOMM, 2013, pp. 15–26, https://dl.acm.org/citation.cfm?id=2486012.

[2] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, et al., B4: Experience with a globally-deployed software defined WAN, in: ACM SIGCOMM, 2013, pp. 3–14, https://dl.acm.org/citation.cfm?id=2486019.

[3] A. Hatami-Marbini, S.M. Sajadi, H. Malekpour, Optimal control and simulation for production planning of network failure-prone manufacturing systems with perishable goods, Comput. Ind. Eng. (2020) 106614.

[4] A. Krause, S. Giansante, Interbank lending and the spread of bank failures: A network model of systemic risk, J. Econ. Behav. Organ. 83 (3) (2012) 583–608.

[5] R. Govindan, I. Minei, M. Kallahalla, B. Koley, A. Vahdat, Evolve or die: High-availability design principles drawn from googles network infrastructure, in: Proceedings of the 2016 ACM SIGCOMM Conference, ACM, 2016, pp. 58–72, https://dl.acm.org/citation.cfm?id=2934891.

[6] M. Ghobadi, R. Mahajan, Optical layer failures in a large backbone, in: Proceedings of the 2016 Internet Measurement Conference, ACM, 2016, pp. 461–467, https://dl.acm.org/citation.cfm?id=2987483.

[7] V. Paxson, End-to-end routing behavior in the internet, IEEE/ACM Trans. Netw. 5 (5) (1997) 601–615, https://ieeexplore.ieee.org/abstract/document/649563/.

[8] Z. Cheng, X. Zhang, Y. Li, S. Yu, R. Lin, L. He, Congestion-aware local reroute for fast failure recovery in software-defined networks, IEEE/OSA J. Opt. Commun. Networking 9 (11) (2017) 934–944, https://ieeexplore.ieee.org/abstract/document/8113138/.

[9] L. Wang, L. Yao, Z. Xu, G. Wu, M.S. Obaidat, CFR: A cooperative link failure recovery scheme in software-defined networks, Int. J. Commun. Syst. 31 (10) (2018) e3560, https://ieeexplore.ieee.org/abstract/document/8113138/.

[10] Y. Chen, T. Farley, N. Ye, Qos requirements of network applications on the internet, Inf. Knowl. Syst. Manag. 4 (1) (2004) 55–76, https://content.iospress.com/articles/information-knowledge-systems-management/iks00061.

[11] C.-Y. Chu, K. Xi, M. Luo, H.J. Chao, Congestion-aware single link failure recovery in hybrid SDN networks, in: 2015 IEEE Conference on Computer Communications (INFOCOM), IEEE, 2015, pp. 1086–1094, https://ieeexplore.ieee.org/abstract/document/7218482/.

[12] D. Li, J. Wu, D. Wang, Single-link failure recovery with or without software-defined networking switches, in: 2018 International Conference on Information and Computer Technologies (ICICT), IEEE, 2018, pp. 87–91, https://ieeexplore.ieee.org/abstract/document/8356846/.

[13] P.M. Mohan, T. Truong-Huu, M. Gurusamy, TCAM-Aware local rerouting for fast and efficient failure recovery in software defined networks, in: 2015 IEEE Global Communications Conference (GLOBECOM), IEEE, 2015, pp. 1–6, https://ieeexplore.ieee.org/abstract/document/7417309/.

[14] X. Yang, H. Xu, L. Huang, G. Zhao, P. Xi, C. Qiao, Joint virtual switch deployment and routing for load balancing in SDNs, IEEE J. Sel. Areas Commun. 36 (3) (2018) 397–410, https://ieeexplore.ieee.org/abstract/document/8314714/.

[15] G. Zhao, H. Xu, S. Chen, L. Huang, P. Wang, Deploying default paths by joint optimization of flow table and group table in SDNs, in: 2017 IEEE 25th International Conference on Network Protocols (ICNP), IEEE, 2017, pp. 1–10, https://ieeexplore.ieee.xilesou.top/abstract/document/8117539/.

[16] X. Jin, H.H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, R. Wattenhofer, Dynamic scheduling of network updates, in: Proceedings of the 2014 ACM Conference on SIGCOMM, ACM, 2014, pp. 539–550.

[17] P. Wang, H. Xu, L. Huang, C. Qian, S. Wang, Y. Sun, Minimizing controller response time through flow redirecting in SDNs, IEEE/ACM Trans. Netw. (2018) https://dl.acm.org/citation.cfm?id=3190751.

[18] M. Suchara, D. Xu, R. Doverspike, D. Johnson, J. Rexford, Network architecture for joint failure recovery and traffic engineering, in: Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, ACM, 2011, pp. 97–108.

[19] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C.L. Lim, R. Soulé, Semi-oblivious traffic engineering: The road not taken, in: 15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18), 2018, pp. 157–170.

[20] H.H. Liu, S. Kandula, R. Mahajan, M. Zhang, D. Gelernter, Traffic engineering with forward fault correction, in: ACM SIGCOMM Comput. Commun. Rev., 44, (4) ACM, 2014, pp. 527–538.

[21] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, M. Schapira, TEAVAR: striking the right utilization-availability balance in wan traffic engineering, in: Proceedings of the ACM Special Interest Group on Data Communication, ACM, 2019, pp. 29–43, https://dl.acm.org/citation.cfm?id=3342069.

[22] Y.S. Chow, H. Teicher, Probability Theory: Independence, Interchangeability, Martingales, Springer Science & Business Media, 2003.

[23] A. Haghani, S.-C. Oh, Formulation and solution of a multi-commodity, multi-modal network flow model for disaster relief operations, Transp. Res. A 30 (3) (1996) 231–250, https://sciencedirect.xilesou.top/science/article/pii/0965856495000208.

[24] D. Johnson, Computers and Intractability-A Guide to the Theory of NP-Completeness, Freeman, San Fransisco, CA, 1979, https://ci.nii.ac.jp/naid/10030608028/.

[25] S. Mitchell, M. OSullivan, I. Dunning, PuLP: a linear programming toolkit for python, The University of Auckland, Auckland, New Zealand, 2011, http://www.optimization-online.org/DBFILE/2011/09/3178 http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.416.4985&rep=rep1&type=pdf.

[26] A. Srinivasan, Approximation Algorithms Via Randomized Rounding: a Survey, in: Series in Advanced Topics in Mathematics, Polish Scientific Publishers PWN, 1999, pp. 9–71, https://dl.acm.org/citation.cfm?id=3190751.

[27] R. Cohen, L. Lewin-Eytan, J.S. Naor, et al., On the effect of forwarding table size on SDN network utilization, in: Proc. IEEE INFOCOM, IEEE, 2014, pp. 1734–1742, https://www.sciencedirect.com/science/article/pii/S1876735410000024.

[28] A. Chateauneuf, M. Cohen, Risk seeking with diminishing marginal utility in a non-expected utility model, J. Risk Uncertain. 9 (1) (1994) 77–91, https://link.springer.com/article/10.1007/BF01073404.

[29] The Openflow Switch, openflowswitch.org.

[30] Open vswitch, http://openvswitch.org/.
[31] The Network Topology from the South Africa, http://www.topology-zoo.org/maps/Sanren.jpg.
[32] J.-M. Kang, T. Lin, H. Bannazadeh, A. Leon-Garcia, Software-defined infrastructure and the SAVI testbed, in: International Conference on Testbeds and Research Infrastructures, Springer, 2014, pp. 3–13, https://link.springer.com/chapter/10.1007/978-3-319-13326-3_1.
[33] A.R. Curtis, J.C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, S. Banerjee, Devoflow: Scaling flow management for high-performance networks, in: ACM SIGCOMM, Vol. 41, (4) ACM, 2011, pp. 254–265, https://dl.acm.org/citation.cfm?id=2018466.
[34] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: ACM SIGCOMM Computer Communication Review, Vol. 38, (4) ACM, 2008, pp. 63–74, https://dl.acm.org/citation.cfm?id=1402967.
[35] The Network Topology from the Monash University, http://www.ecse.monash.edu.au/twiki/bin/view/InFocus/LargePacket-switchingNetworkTopologies.
[36] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, R. Wattenhofer, Achieving high utilization with software-driven WAN, in: ACM SIGCOMM Computer Communication Review, Vol. 43, (4) ACM, 2013, pp. 15–26.
[37] A. Mahimkar, A. Chiu, R. Doverspike, M.D. Feuer, P. Magill, E. Mavrogiorgis, J. Pastor, S.L. Woodward, J. Yates, Bandwidth on demand for inter-data center communication, in: Proceedings of the 10th ACM Workshop on Hot Topics in Networks, ACM, 2011, p. 24.
[38] X. Jin, Y. Li, D. Wei, S. Li, J. Gao, L. Xu, G. Li, W. Xu, J. Rexford, Optimizing bulk transfers with software-defined optical WAN, in: Proceedings of the 2016 ACM SIGCOMM Conference, ACM, 2016, pp. 87–100.
[39] D. Applegate, E. Cohen, Making intra-domain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs, in: Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, ACM, 2003, pp. 313–324.
[40] M.R. Garey, D.S. Johnson, Computers and intractability: A guide to the theory of npcompleteness (series of books in the mathematical sciences), ed, Comput. Intractability (1979) 340.

Xuwei Yang received B.S. degree in network engineering from the Chang'an University and the Ph.D. degree in computer science and technology from the University of Science and Technology of China in 2016 and 2021, respectively. His main research interest is software defined networks and cloud network.

Hongli Xu received the B.S. degree in computer science and the Ph.D. degree in computer software and theory from the University of Science and Technology of China in 2002 and 2007, respectively. He is currently an Associate Professor with the School of Computer Science and Technology, University of Science and Technology of China. He has authored or coauthored over 70 papers, and held about 30 patents. His main research interest is software-defined networks, cooperative communication, and vehicular ad hoc network.

Jianchun Liu received B.S. degree in 2017 from the North China Electric Power University. He is currently a Ph.D. candidate in the School of Data Science, University of Science and Technology of China (USTC). His main research interests are software defined networks, network function virtualization, edge computing and federated learning.

Chen Qian received the B.S. degree from Nanjing University in 2006, the M.Phil. degree from The Hong Kong University of Science and Technology in 2008, and the Ph.D. degree from The University of Texas at Austin in 2013, all in computer science. He is currently an Assistant Professor with the Department of Computer Engineering, University of California at Santa Cruz. His research interests include computer networking, network security, and Internet of Things. He has authored over 60 research papers in highly competitive conferences and journals. He is a member of the ACM.

Xingpeng Fan received B.S. degree in 2017 from the University of Science and Technology of China . He is currently a Ph.D candidate student in Computer Science at the University of Science and Technology of China. He will receive the doctor's degree in 2022. His main research interest is software defined networks, data center network, cloud computing and edge computing.

Dr. He Huang is an associate professor in the School of Computer Science and Technology at Soochow University, P.R. China. He received his Ph.D. degree in Department of Computer Science and Technology from University of Science and Technology of China (USTC), in 2011. His current research interests include traffic measurement, spectrum auction, privacy preserving in auction, and algorithmic game theory. He is a Member of both IEEE and ACM.

Haibo Wang received B.S. degree in 2016 and M.S. degree in 2019 from the University of Science and Technology of China . He is currently a Ph.D. student in the Department of Computer and Information Science and Technology, University of Florida . His main research interest is software defined networks and Internet traffic measurement.