FLAIR: A Fast and Low-Redundancy Failure Recovery Framework for Inter Data Center Network

Yuchao Zhang, *Member, IEEE*, Haoqiang Huang, *Member, IEEE*, Ahmed M. Abdelmoniem, *Member, IEEE*, Gaoxiong Zeng, *Member, IEEE*, Chenyue Zheng, *Student Member, IEEE*, Xirong Que, *Member, IEEE*, Wendong Wang, *Member, IEEE*, and Ke Xu, *Fellow, IEEE*

***Abstract*—Due to the fast developments of 5G and IoT technologies, Inter-Datacenter (Inter-DC) networks are facing unprecedented pressure to duplicate large volumes of geographically distributed user data in a real-time manner. Meanwhile, with the expansion of Inter-DC networks scale, link/node failures also become increasingly frequent, negatively affecting the data transmission efficiency. Therefore, link failure recovery methods become of utmost importance. Many works investigated fast failure recovery, yet none of them consider the deployment overhead of such recovery schemes. While in this article, we found that the side-effect of deploying recovery strategies and the future availability of the recovered transmissions are also crucial for fast recovery. So we propose a fast and low-redundancy failure recovery framework, FLAIR, which consists of a fast recovery strategy FRAVaR and a redundancy removal algorithm ROSE. FRAVaR takes full consideration of deployment overhead by minimizing shuffle traffic. On its base, ROSE regularly eliminates the cumulative rerouting redundancy by removing unnecessary routing updates. The experiment results on 4 realistic network topologies show that FLAIR successfully reduces up to 48.2% deployment overhead compared with the state-of-the-art solutions, and thus reduces up to 70.2% recovery speed and improves up to 36% network utilization.**

***Index Terms*—Inter data center network, failure recovery, routing optimization.**

I. Introduction

In RECENT years, user data from edge network is continuously growing, which requires near real-time transmissions between geographically distributed data centers (DCs). In order to keep pace with such large number of data transmissions, Internet Service Providers (ISPs) are extending their Inter-DC networks at an incredible speed. For example, Google tripled its Inter-DC network B4 [1] scale to carry its data traffic which has increased by 100 times in five years [2]. At the same time, the requirement for network availability has also increased even to over 99.99% [3], because even second network failure will cause 100GBs network traffic loss and even affect ISP's global services [4]. In the real network environment, link failure are becoming more and more frequent with the expanding inter-DC network scale. According to the report in [5], in a practical Inter-DC wide area network with about 200 routers and 6000 links, the probability of link failures in every 5 minutes is close to 25%, and that in every 10 minutes is approximately 40%.

The Inter-DC network needs a fast failure recovery framework to handle such frequency failures and improve its availability. Many previous efforts have focused on designing near optimal recovery schemes. These works can be classified into two categories. 1) Pre-failure resource reservation. Redundant resources like bandwidth or paths will be reserved before data transmissions. When failure occurs, the reserved resources will be used to maintain network availability. This kind of solution is adopted by previous works such as FFC [5] and SMORE [6]. FFC [5] limits the transmission bandwidth for network congestion caused by failure recovery. In SMORE [6], many routes are combined into one flexible group and backed up with each other via R $\ddot{a}$ cke's oblivious routing algorithm [7]. 2) After-failure recovery. For example, TEAVAR [8] allocates as many routers as possible, with a low probability of failure, and achieves link availability by redistributing flows to other links. The above work has very good results in calculating the optimal recovery plan, but in order to achieve fast failure recovery in real networks, it is also necessary to take the deployment overhead into consideration. Otherwise, with high deployment overhead, it will still result in slow overall recovery speed even if with an optimal recovery plan. The existing solutions, to our best knowledge, fail to address the following three key concerns in real inter-DC scenarios.

Deployment overhead: The measurements in [9] show that updating routing rules takes noticeable time. For example, changing only one routing rule requires up to 466 rule movements in routing tables. Updating 20 K rules results in millisecond-level delays. While the completion time of most transmissions is less than a few milliseconds (90% of the traffic between DCs is less than 200KB [10], and the bandwidth is GB-level) Therefore, those "optimal strategies" that require a large number of routing

Manuscript received 8 May 2023; revised 16 April 2024; accepted 20 April 2024. Date of publication 2 May 2024; date of current version 7 June 2024. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62172054 and Grant 62072047, and in part by Beijing Nova Program under Grant 2023140. Recommended for acceptance by X. Chu. *(Gaoxiong Zeng contributed in this work independently while not affiliated with Huawei.) (Corresponding author: Yuchao Zhang.)*

Yuchao Zhang, Haoqiang Huang, Chenyue Zheng, Xirong Que, and Wendong Wang are with the Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: yczhang@bupt.edu.cn; hhq_erii@bupt.edu.cn; zcy2001@bupt.edu.cn; rongqx@bupt.edu.cn; wdwang@bupt.edu.cn).

Ahmed M. Abdelmoniem is with the Queen Mary University of London, E1 4NS London, U.K (e-mail: ahmed.sayed@qmul.ac.uk).

Gaoxiong Zeng is with Huawei Technologies, Shenzhen 518129, China (e-mail: gaoxiong.zeng@huawei.com).

Ke Xu is with the Tsinghua University, Beijing 100190, China (e-mail: xuke@tsinghua.edu.cn).

This article has supplementary downloadable material available at https://doi.org/10.1109/TCC.2024.3393735, provided by the authors.

Digital Object Identifier 10.1109/TCC.2024.3393735

updates is likely to cause a large number of small flows to retransmit, leading to low recovery efficiency.

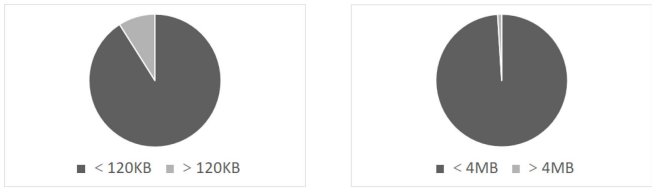
Heavy traffic shuffle: Recovery plans always results in a large number of flow path adjustments with innumerable transmissions, especially in large Inter-DC networks. These flows, whose transmission paths are adjusted, are called "shuffled" flows. During the rerouting process, these shuffled flows require a lot of time to deal with out-of-order packet arrivals, congestion, and packet loss. These processes contribute take significant time to the overall failure recovery process. Therefore, we need a new method that can limit the amount of shuffled traffic to accelerate overall failure recovery.

Routing redundancy: Frequent failure recovery brings redundancy to the global routing which will finally degrade network performance. Even with the best optimizations, 4 ms control plane failure recovery still produces 2.4% routing redundancy [11], which would possibly result in longer path and longer transmission delay. Therefore, we point out that after frequent failure recovery, global routing needs to be re-optimized to mitigate redundancies.

To address the above problems, we propose *FLAIR*, a fast failure recovery framework. In expanding our conference paper FRAVaR [12], we have developed and incorporated a new system component called "ROSE", effectively combining both into an integrated system for enhanced functionality. FLAIR constitutes two major parts: a fast failure recovery algorithm (FRAVaR) and a routing optimization algorithm (ROSE). FRAVaR is a two-phase algorithm with rerouting and traffic shuffling. To minimize the deployment overhead and reduce traffic shuffle, we propose *Incremental Rerouting* (IR). IR isolates link failures within a limited range of networks by carefully selecting a few available links to replace the failed ones. The reduction of routing update speeds up recovery strategy deployment. Then, we adopt Value at Risk theory (*VaR*) [13] to devise a flow shuffle scheme, which further increases the availability of our reroute results. When FRAVaR is executed multiple times and the accumulated redundancy exceeds a certain threshold, ROSE will be triggered to eliminate the routing redundancy. Based on the State Estimation (SE) model, ROSE searches and implements the global optimal routing plan. First, based on routing state modeling, it estimates whether there are redundancies in existing routing based on SE theory to estimate routing state under linear time complexity. Then, for those redundant routing, ROSE searches for best state routing with a heuristic algorithm. We evaluate the performance of FLAIR on four real large-scale Inter-DC network topologies (AGIS, ATT, Global Center, and IBM). Experiment results show that FLAIR reduces up to 48.2% deployment overhead compared with the state-of-the-art solutions, thence improves network utilization by at most 36.0% and reduces overall recovery time by up to 70.2%.

The main contributions of this paper are as follows:

- We for the first time disclose the deployment overhead to achieve fast failure recovery in real large-scale inter-DC networks.
- We propose a fast failure recovery framework FLAIR, which constitutes two novel algorithms (i.e., FRAVaR and ROSE). FRAVaR introduces an incremental reroute algorithm *IR* and a traffic shuffle algorithm based on VaR. ROSE estimates the routing state with State Estimation (SE) theory and eliminates unnecessary routing updates.
- We evaluate the performance of FLAIR on four real Inter-DC topologies and demonstrate the effectiveness of FLAIR on reducing the deployment overhead and accelerating overall failure recovery speed.



(a) FBHadoop (b) WebSearch

Fig. 1. Flow size distribution of Inter-DC network data sets.

The remainder of this paper is organized as follows. We discuss the challenges of Inter-DC network failure recovery and motivate the design of FLAIR in Section II. In Section III we briefly introduce the recovery principle and process of FLAIR. Then, we introduce the specific framework model in Sections IV and V. We then show evaluations of FLAIR and other recovery solutions, including the implementation settings and results in Section VI. We review related work about failure recovery in Section VII. Finally, we conclude the paper in Section VIII.

II. Motivation

We find that, intuitively, the deployment overhead is one of the obstacles to fast link recovery. In the traditional routing mechanism, SDN controller sends out new routing rules to routers that new routing paths traverse, and these routers update the new rules into their routing table. But at the same moment, there are thousands of flows on the paths and they have to wait for the update to complete. When failures occur, the network stops transmitting these flows, deploys a new routing strategy, rebuilds connections, and restarts transmission. Over this process, the problem is that the updating speed is not faster than the transmission of most packets and re-transmission introduces significant delays. To locate the next hop in routing tables within a fraction of a millisecond, Ternary Content Addressable Memory (TCAM), for its concurrent and efficient searching structure, is broadly installed in routers as the proprietary memory to store routing tables.

Unfortunately, unlike the efficiency in searching, updating rules in TCAM is not swift enough. We show the flow size distribution in Inter-DC networks from two large data sets, FBHadoop [10] in Fig. 1(a) and WebSearch [14] in Fig. 1(b). More than 90% of flows' sizes are less than 120 KB in FBHadoop and approximately 99% of flows' sizes in WebSearch are within 4 MB. That is to say, in 400 G Inter-DC networks, almost every flow finishes transmission within 0.1 milliseconds, while the time cost of updating 20 K routing rules is a hundredfold. And that is what hinders a fast link failure recovery. Although optimizations for TCAM [15], [16] require fewer entry moves and less time cost, the latency caused by routing rule updates

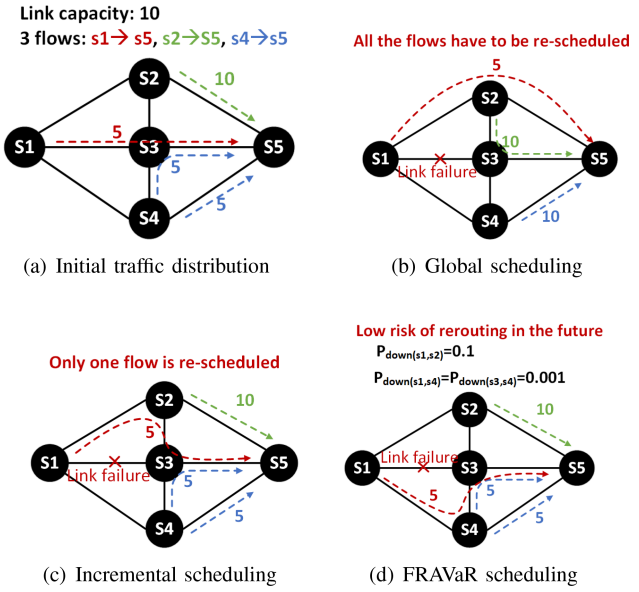


Fig. 2. Network example of recovery rerouting.

is hard to be ignored compared to the flow completion time in high-speed networks [17].

Besides, restarting transmission is never a slight task and it is likely to influence applications. For instance, flow connection establishment needs TCP connection establishment first, which consumes at least 1.5 Round-Trip-Time (RTT) and also CPU resources. In this case, CPU resources assigned to handle the transmission sharply rise , leaving fewer CPU resources for other works. As a result, the performance of both the network and those applications dependent on the hosts (e.g., distributed machine learning), drops down. And this is the reason why the scale of flows' re-transmission matters significantly in network performance.

FLAIR is first motivated by the observation that although global rerouting achieves better scheduling performance, it causes a larger scale of flow shuffle, which further introduces many negative effects: 1) The controller has to solve a global optimization problem, which is highly complex and time-consuming. 2) The network has to update more routing rules in the routers on the reroute paths. 3) Shuffling from one path to others leads to the release and reconstruction of session connections, bringing additional delays to flows' completion times (FCT).

We illustrate our motivation in the example shown in Fig. 2. By default, all flows are routed following the ECMP rules. Fig. 2(a) shows an initial traffic distribution with four flows, f_1: $s_1 \rightarrow s_3 \rightarrow s_5$, f_2: $s_2 \rightarrow s_5$, f_3: $s_4 \rightarrow s_3 \rightarrow s_5$, and f_4 : $s_4 \rightarrow s_5$. The dashed curves represent flows and the numbers represent the traffic volume they carry. When link $s_1 \rightarrow s_3$ fails, the controller re-computes all the transmissions and adjusts flow assignment like Fig. 2(b), i.e., f_1 to $s_1 \rightarrow s_2 \rightarrow s_5$. This strategy requires updating all the flows in this network, to deploy its rescheduling result. Recall the complex deployment process shown at the beginning of this section, so many rescheduled transmissions certainly bring high deployment overhead.

However, if we adjust f_1 to $s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_5$, as shown in Fig. 2(c), other flows would not be affected anymore and thus avoids re-transmission. Although it's not the optimal transmission path for f_1 compared to Fig. 2(b), only two routers need to be reconfigured and all the other routers can remain the same as before so that f_1 can be rescheduled quickly and the whole network can fast react to link failures. This also isolates these link failures within only one path $s_1 - s_3 - s_5$, making it unable to affect other flows outside this range. Such isolation also contributes to reducing computation complexity.

While Fig. 2(c) seems to be a better choice when all the links are equally stable, it is not good enough when different links have different failure probabilities. As shown in Fig. 2(d), if the failure probability of link $s1 \rightarrow s2$ is 0.1, while that of link $s1 \rightarrow s4$ and $s4 \rightarrow s3$ are both 0.001, there would be a great risk that the scheduling results in Fig. 2(c) which is unstable because f_1 is more likely to fail again leading to more rescheduling in the future. In contrast, the assignment in Fig. 2(d), which enjoys a more stable path, can shuffle f_1 to $s_1 \rightarrow s_4 \rightarrow s_3 \rightarrow s_5$ to result in a recovery schedule that enjoys higher availability.

It is not difficult to find that in the process of global recovery, 80% of nodes need to deploy new routing rules. In the incremental recovery process, routing updates only need to be deployed on 60% of nodes. Obviously, in the global recovery, the four flows only need a 6-hop routing to complete the transmission and traverse through fewer redundant links. However, it needs to deploy routing updates on more nodes and it also causes a large number of flows that might not need rescheduling. Incremental recovery achieves a similar recovery effect with less deployment. Flows need a 7-hop routing, while routing updates and flow shuffle are fewer.

Nevertheless, Fig. 2(c) and (d) introduce routing redundancy. Considering that Inter-DC network is much more complicated with an increasing number of links, the global recovery requires a large number of shuffled flows and long latency due to updating routing rules. What makes matters worse is that it also increases the FCT which is affected by re-transmissions. With the illustrative example above, we can see that incremental scheduling can speed up the recovery by only adjusting the affected flows rather than computing a global optimal re-transmission scheme. And, we further note that incremental scheduling should take link failure probability into account to reduce the risk of rescheduling. Meanwhile, the incremental rerouting achieves little updating overhead at the cost of routing redundancy. In Fig. 2(c) and (d), the path from s1 to s5 is not the optimal one. Additionally the redundancy would accumulate and finally degrades transmission performance due to a longer propagation delay. Subsequently, the routing needs to be re-optimized after the frequent recovery. This motivates the design of FLAIR, which reveals the potential of reacting to link failures in near real-time at the scale of tens of thousands of links with different failure probabilities.

In brief, incremental rerouting distinguishes itself from traditional link protection approaches [6], [18], [19] by significantly reducing the overall deployment overhead and simplifying the computational complexity involved in network reconfiguration following a link failure. From the previous artificial example, we have shown that it is more efficient for only part of the network

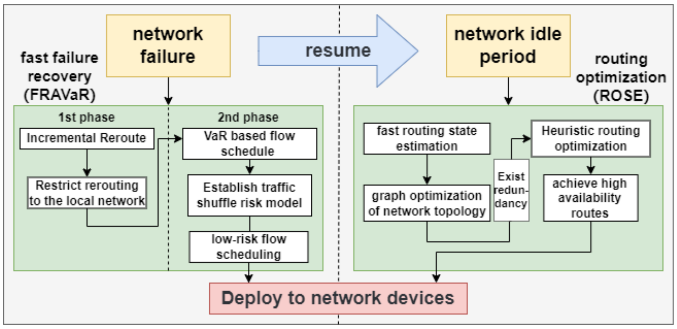


Fig. 3. The workflow diagram of FLAIR's structure.

to participate in failure recovery while trading off a negligible performance impact. FLAIR brings in the following benefits: 1) fewer changes of routing rules, 2)smaller scales of shuffle traffic and 3) low-redundant routing, which greatly speeds up the process of the network recovering from failures and returning to a normal operation state.

III. System Overview

Motivated by the above discussion, FLAIR comes up with a very simple idea, *less is better*. The fewer routes we adjust, the fast reaction we achieve. The fewer flows we shuffle, the fast recovery we achieve. FLAIR consists of two main components, FRAVaR and ROSE. When network failures occur, FLAIR first starts FRAVaR to complete fast rerouting and flow re-transmission. After repeatedly recovering from failure, ROSE starts for routing optimization during the network idle period.

Fig. 3 shows the internal operation process of FLAIR. First, *FRAVaR* tries to adjust the minimum number of transmissions that are already built up.Following this principle, we narrow the range of reroutes within a local part of the network while still maintaining the performance. Along the way, we propose IR, an incremental reroute algorithm. IR adjusts routing rules only on a short route bypass the failed link and carefully selects rerouting paths by the breadth-first-searching (BFS) algorithm which promises completeness of routes recovery. We use the visited tags generated by BFS to guarantee all the routes are disjoint. In this way, IR isolates link failures into a small domain of the network. Based on IR, we then strengthen FRAVaR by introducing a minimum loss flow shuffle algorithm based on the Value at Risk theory. Specifically, we calculate the mathematical expectation of the flow shuffle scale, or so-called "risk", and minimize it by solving a linear programming problem. We combine VaR with IR and introduce FRAVaR in more detail in the next section.

Second, *ROSE* aims to achieve the same routing purpose with as few routing rules as possible. To this end, ROSE introduces a routing state model that takes failure probability, path capacity and routing hops into consideration. Drawing on this model, ROSE leverages heuristics to find highly available, low-redundancy routes. The process incurs notable overheads–as re-computation and updates for routing between data center pairs are frequently necessary. To solve this problem, we introduce SE theory to fast estimate routing state under linear computational complexity. The main idea of SE is to convert routing optimization to a graph optimization problem. Then, we derive the condition for which a routing under the best state should meet, through abstract mathematical proof. As long as the routing between any two DCs satisfies the condition, ROSE determines that the routing needs no update and therefore, it reduces the complexity by eliminating unnecessary parts of the routing optimization.

IV. FRAVaR Design

In this section, we first introduce a network model and next FRAVaR's detail design by modelling it as a two-phase solution, IR as the incremental rerouting algorithm and Var as the minimal flow shuffle algorithm.

A. Network Model

In the beginning, we consider an Inter-DC network $G = (V, E)$ with nodes V and directed links E. The link from node i to node j is simply denoted as ij and c_{ij} is the capacity of link ij. A route r is the set of head-to-tail links and, for example, a route from node a to f is expressed as $r = ab, bc, cd, df$. R is the general set of routes. A flow is defined as f and the demand scale of it is defined as d_f. Then, the scale of f allocated to route r is defined as x_r^f. y^{ij} denotes whether link ij is available, which $y^{ij} = 0$ indicates that ij is broken and $y^{ij} = 1$ means ij is available. Similarly, y^r defines the availability of route r. The failure probability of link ij, when considering link failure as a mutually independent event, is simply a decimal p_{ij} between 0 and 1.

B. Incremental Rerouting

Although a global schedule brings an optimal performance in throughput, utilization, availability, MLU, etc, it also slows down the processing speed of recovery, and the time delay is elongated. A little sacrifice of performance would extremely improve recovery speed. Here comes two design principles behind IR.

A trick to boost efficiency: In traditional traffic engineering (TE), recovery algorithms are running on the whole network. Global failure recovery solutions can achieve good performance in many ways, but not all. The scale of links chosen to recover the network needs to be pruned to improve recovery efficiency. When a link failure occurs, global recovery algorithms take the whole network into consideration, making the scale of route space too large to explore and resulting in extremely high computation overhead. So the critical point of our fast recovery solution is to reduce the exploration space while maintaining network performance. An intuitive idea is to focus on a subset of the network which recovering on the sub-network achieves a close resilience performance to recovering on the whole network. Following this idea, we design a local search algorithm for rerouting to isolate failures.

Avoiding Shared Risk Link Group (SRLG): SRLG is one of the problems that reroute algorithms try to avoid. A SRLG is a

Algorithm 1: Incremental Reroute (IR) Method.

Input: G, ij, n
Output: P

```
c := 0;
push i into minimum heap Q;
while c < n and Q not empty do
    T := top of Q;
    pop Q;
    h := last node of T;
    foreach hk ∈ E, vis[hk] = false and k ∉ T do
        set T' = T;
        push k into T';
        if k = j then
            push T' into P;
            c := c + 1;
        else
            push T' into Q;
            vis[lg] := true, ∀lg ∈ SRLG_hk
```

group of links that share the common resources. In link failure recovery problems, it indicates multiple paths share the common links. They "share" the failure risk when they share the same link and whenever this shared link fails, each route that traverses this SRLG goes down. This makes it one of the most critical situations because the network has to recover multiple routes simultaneously. Moreover, the shared link in SRLGs restricts the available capacity to the flows traversing the routes. As a solution, we enhance IR by examining the visited tags and achieve an SRLG-disjoint reroute algorithm.

Here we introduce an incremental reroute algorithm of our fast recovery framework, IR, and show it in Algorithm 1. IR scans the Inter-DC network G, collects failure links(e.g., ij), counts the number of expected rerouting routes(e.g., n), and designates a subset of the network, denoted as P. IR then applies BFS on P and implements it with a minimum heap Q. Moreover, IR marks a visited tag on the nodes when they are traversed so that later searches refuse to select these nodes with the visited tag. The visited tag provides two guarantees: 1) With the help of BFS, it ensures that when a node is reached, hops of the route are minimal; and 2) the Intersection of routes is avoided. Through the visited tag and BFS, IR offers a set of rerouting paths P with mutually disjoint and shortest paths.

IR introduces minimum heap Q to accelerate the search process. BFS uses extensive time and space to store the adjacent nodes and choose the best one from them. Referring to other graph searching algorithms like Dijkstra, GBFS, and A-Star, a common characteristic of them is the introduction of a priority queue to reduce selection time and used space. At the same time, we define a cost function f_k to sort the heap, while the top of the heap is the link with minimal f_k. For example, in A-Star, the cost function is $f_k = Distance(source, k) + Distance(k, destination)$. Here we introduce a cost function for IR, which considers both capacity and resilience.

$$f_k = p_{hk} \cdot \frac{c_{hk}}{c_{\max}}. \tag{1}$$

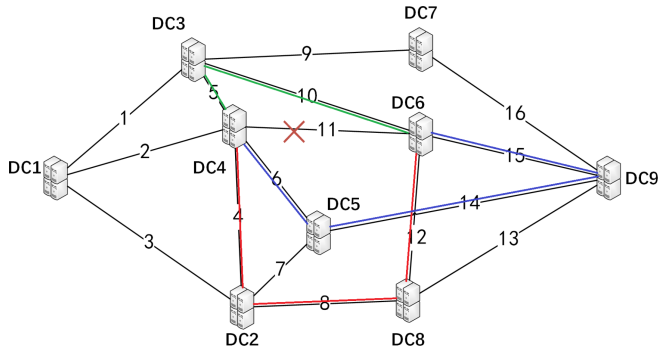


Fig. 4. Example of IR searching process.

Where h is the current node being travelled and k is one of the adjacent nodes. $c_{\max}$ is the maximum capacity of all the available links. With the help of minimum heap, IR chooses the link with the minimal cost within time cost $O(logK)$, where K is the number of links in the sub-network.

Fig. 4 shows an example of how IR searches $n = 3$ rerouting routes for a link failure (i.e., link 11 $(DC4, DC6)$ is broken). First we begin with node $DC4$ and get four links adjacent to it. That is, link 2 to $DC1$, link 4 to $DC2$, link 5 to $DC3$, and link 6 to $DC5$. We mark visited tags on $DC1$, $DC2$, $DC3$, and $DC5$. And the next round, we get ten routes derived from the previous four routes and they are {Link 2, Link 1}, {Link 2, Link 3}, {Link 4, Link 3}, {Link4, Link7}, {Link 4, Link 8}, {Link 5, Link 1}, {Link 5, Link 10}, {Link 5, Link 9}, {Link 6, Link 7} and {Link 6, Link 14}, while six of them meet the nodes with visited tag and are abandoned. For example, the first route, {link 2, link 1}, reaches $DC3$ which has been marked a visited tag by route {link 5} and it denotes that the hop of {link 2, link 1} is more than {link 5}. Among the rest four routes, we obtain a reroute route, {link 5, link 10}, and three routes for the next round. Then, in the third round, we get four routes {Link 4, Link 8, Link 12}, {Link 6, Link 14, Link 15}, {Link 4, Link 8, Link 13} and {Link 5, Link 9, Link 16}, while the first half of them are valid (the optimal path). Thus, IR obtains three SRLG-disjoint rerouting routes with minimum hops and finishes searching.

C. Flow Shuffle and Formulation

We design a risk evaluation function to describe the expectation of the loss for the flow shuffle scheme and formulate a flow shuffle problem within this function. A natural solution to evaluate the expectation of flow loss is to simulate every possible scenario in which *every subset of surviving links fails*. The time complexity is $O(2^N)$ where N is the number of links in network G. Obviously, such time complexity growing with the size of the network is unacceptable. Now, with IR narrowing the scale of the network, the time complexity is reduced to $O(2^K)$ where K is the number of links in the sub-network, and in most cases, we have $K \ll N$ and thus making reactive computation of these scenarios more practical.

Here we introduce the probability of these scenarios. First we denote a scenario as q and the availability of link ij in q is denoted as y_q^{ij}. That is, in scenario q, link ij is available ($y_q^{ij} = 1$) or down ($y_q^{ij} = 0$). And route availability y_q^r is derived from the following expression:

$$y_q^r = \prod_{hk \in r} y_q^{hk}. \tag{2}$$

We obtain the probability of link failures from historical records, and the probability p_q of scenario q to occur is naturally the probability of paths combinations as follows:

$$p_q = \prod_{ij \in E} \left[(p_{ij} * y_q^{ij}) + (1 - p_{ij}) * (1 - y_q^{ij}) \right]. \tag{3}$$

For the reason that the scenarios unlikely to occur are of little value and little risk, we prune the scenarios with $p_q \leq 10^{-6}$. Then, we introduce the percentage of flows NOT meeting the demands. Despite different link capacities on Inter-DC network, the loss is capable of measuring by proportion. Thus, the loss of scenario q is formulated as follows:

$$\theta_q = \sum_f [1 - \frac{\sum_{r \in R} x_r^f y_r^q}{d_f}]^+. \tag{4}$$

Where θ_q is the loss and d_f is the demand of flow f. The notation $[\alpha]+$ means 0 when $\alpha \leq 0$ and it guarantees that in extreme situations, like network overload, the loss is not less than 0. Especially, in the scenario that all links are available, the loss is 0. So the "risk" Θ of the re-allocation scheme is given as

$$\Theta = \sum_q p_q \theta_q. \tag{5}$$

So far, we have finished the construction of our framework and the objective is clearly defined. The formulation of the "risk" problem is shown as follows:

$$\begin{aligned} \min \quad & \Theta \\ \text{s.t.} \quad & (2) - (5) \\ & \sum_f x_f^r \leq c_{ij}, \forall ij \in r \qquad (6) \\ & \sum_r x_f^r \geq d_f. \qquad (7) \end{aligned}$$

Notably, constraint (6) specifies that the allocated share of all flows is not greater than the minimal link capacity of the path and it provides guarantees for minimum congestion control in the network. Constraint (7) indicates that all the shuffled data of a flow is not less than its demand. Note that, linear programming gets no solution in some corner cases. Especially in the cases where most of the network breaks down while the data request is still large, it is impossible to meet constraints (6) and (7) simultaneously and hence a balanced distribution should be implied. Even though these are extreme situations, in most cases the LP is solvable.

In Appendix A, available online we provide a performance bound of FRAVaR and explain the reason why IR is based on the BFS algorithm.

V. ROSE Design

In this section, we introduce FLAIR's routing optimization method namely ROSE. To this end, we present a state model to evaluate the path status. Based on this model, we explain how to search paths that are under the "best" state and how to reduce computation and deployment overhead with SE theory.

A. The Path State Model

In the failure recovery framework, the state of the path is decided by the transmission distance, the capacity of links and the failure probability. Based on the network model in Section IV-A, we propose a micro model to describe path status. First, the capacity of the path is determined by the minimum capacity of links that it traverses [20]

$$c_r = \min_{ij \in r} c_{ij}. \tag{8}$$

Similarly, the failure probability of a path is determined by the mathematical expectation of failure probability of links that it traverses

$$p_r = (1 - \prod_{ij \in r} (1 - p_{ij})). \tag{9}$$

For the reason that it is hard to measure the accurate transmission distance, we use routing hops η_r to indicate the actual distance of path r. Therefore, the state of path r could be defined by the weight Φ_r as below

$$\Phi_r = \frac{c_r \cdot p_r}{\eta_r}. \tag{10}$$

We define a path under the optimal state as the path with the highest weight among all paths.

$$\Phi_r \geq \Phi_{\bar{r}}, \forall \bar{r} \in R'. \tag{11}$$

Where R' is the set of paths that connects the same source and destination nodes as path r.

B. Path Search Over the State Model

Based on the micro model of the path state, we use a heuristic method to search for the optimal path. As shown in Algorithm 2, it searches the highest weighted path between two nodes i and j. Node set UN and VN separately contain the nodes selected in path r_{ij} and the nodes not selected. In the beginning, it pushes node i into the selected node set UN. In each epoch, it selects a node k from VN, based on the largest sum of $\Phi_{r_{ik}}$ and $\Phi_{r_{kj}}$. At the end of each epoch, it pushes the selected node of this cycle into UN and erases it from VN. Then, it updates the weight of nodes in VN with the nodes in UN. Here, the weight of $\Phi_{r_{kj}}$ is roughly predicted by the existing routing.

C. Leveraging State Estimation (SE) Theory

The heuristic method promises that the optimal path is searchable, while the computation and deployment overhead is still too high. Suppose that the Inter-DC network has N nodes, then the controller has to compute N^2 paths for each node pair, which introduces $O(N^3) \sim O(N^4)$ overhead. Hence, we

Algorithm 2: Heuristic Path Search Method.

Input: G, ij
Output: r_{ij}

```
push i into node set UN;
while j ∉ UN do
    foreach u ∈ UN do
        Φ_{r_ik} := 0;
        Φ_{r_kj} := 0;
        foreach v adjacent to u do
            if Φ_{r_iu+v} + Φ_{r_vj} ≥ Φ_{r_ik} + Φ_{r_kj} then
                k := v;
                r_ik := r_iu + v;
    push k into UN;
    record k into r_ij;
    update Φ_{r_ik}, ∀k ∈ VN;
```

leverage State Estimation (SE) theory to reduce the overhead. The main idea of SE is that it estimates the state of paths and eliminates the computation and deployment of existing paths if the paths are already in the optimal state. In reality, SE realizes this with graphical and mathematical tools.

First, we assume that paths before failure are in an optimal state. It indicates that the paths, which are out of the range of failure effect, are optimal. SE denotes the set of these paths with V, while set U denotes the paths affected by failures. The status of the path in U is unknown for the reason that it contains the recovery routing and new links are introduced, which leads to changes in path capacity and failure probability, and the increase of routing hops.

SE converts the estimation problem to a graph-proof problem. It views the network topology G as a graph and each path r in set U is a subgraph $G_1 = (V_1, E_1)$ of G. Any other path $\bar{r}$ in R' is also a subgraph $G_2 = (V_2, E_2)$ of G. We use n and m to denote the number of nodes in G_1 and G_2, and use p_1 and p_2 to denote the highest link failure probability of G_1 and the average link failure probability of G_2. We propose Theorem V.1 to provide a basic estimation.

Theorem V.1: For $\forall G_2$, if G_1 satisfies $mp_1 - mnp_2 \leq m - n$, and $c_r \geq c_{\bar{r}}$, path r is under optimal state.

Proof of Theorem V.1 is shown in Appendix B, available online. Theorem V.1 reduces the computation overhead, while it still needs to search the whole network for all the paths in R'. The computation overhead of acquiring these paths is unacceptable as well. Hence, we propose Theorem V.2, which completes the task by relaxing certain conditions of Theorem V.1. The symbol definition of Theorem V.2 is a bit different. We use $\bar{r}'$ to denote the path with the minimum number of links in R' and use G'_2 to denote the subgraph of $\bar{r}'$. $\bar{m}$ indicates the number of nodes in $\bar{r}'$. Then, we define subgraph $\bar{G_1} = G - G_1$ and denote $\bar{p_2}$ as the average of link failure probability of $\bar{G_1}$. $c'_{\bar{G_1}}$ denotes the maximum link capacity of subgraph $\bar{G_1}$. Finally, we have Theorem V.2 as below:

Theorem V.2: If $\bar{m}p_1 - \bar{m}n\bar{p_2} \leq m - n$ and $c_r \geq c'_{\bar{G_1}}$, path r is under optimal state.

TABLE I
DETAILS OF THE INTER-DC NETWORK TOPOLOGIES

Topology Name	#. of DCs	#. of links
ATT North America	25	112
IBM	18	48
GlobalCenter	9	72
AGIS	25	60

Proof of Theorem V.2 is shown in Appendix C, available online. Through Theorem V.2, SE just needs to search the path with the minimum number of links and the state of the existing path r is clearly known. Once SE detects that existing path r is under optimal state, it erases all the paths that share the same recovery routing with r from set U, which sharply cuts down computation and deployment overhead of the routing optimization process.

VI. Evaluation

In this section, we evaluate the performance of FLAIR and the experiment results show that:

1) FLAIR reduces the failure recovery time by 70.2% on average compared with the state-of-the-art failure recovery solutions.
2) While reducing the failure recovery time and availability risk, FLAIR also maintains comparable performance on network throughput and achieves up to 36.0% utilization improvement.
3) FLAIR minimizes the influence of recovery on traffic and reduces the scale of shuffle flows by 29.8% on average compared with other evaluated solutions. In extreme scenarios, the outperformance is up to 90.7%.
4) ROSE optimizes the routing with up to 46.4% utilization improvement and SE can achieve a 48.2% deployment overhead reduction at most.

A. Experiment Setting

We conduct a series of simulation experiments to evaluate the performance of FLAIR. For simulations, we use Gurobi [21] as the optimization framework. We then develop a data-driven simulation software, which runs on a Windows 10 platform with an AMD Ryzen5 2600 processor. It consists of three major components:

Simulation data source: The following describes the data used in our simulations which includes:

- *Topology:* Our simulations use four real Inter-DC network topologies: ATT, IBM, GlobalCenter, and AGIS, which are real Inter-DC network topologies extracted from the internet topology zoo [22]. These networks cover situations from looseness to compactness. Table I shows the size of the networks and Fig. 5 shows the abstract structure of each topology. Among these topologies, ATT is the largest network with 112 Inter-DC links while IBM is the smallest

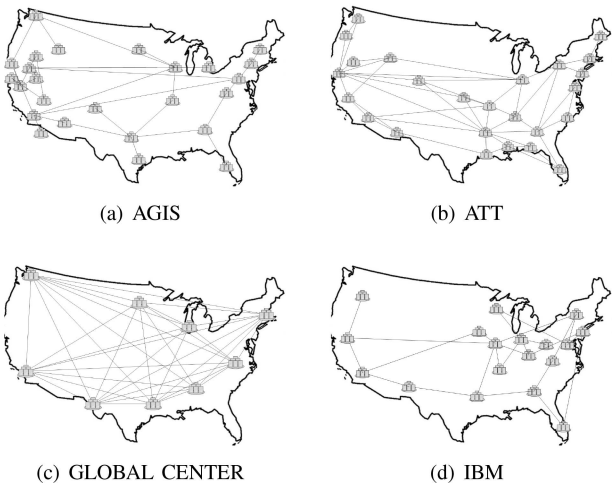


Fig. 5. Geographic distribution of the typical inter-DC networks.

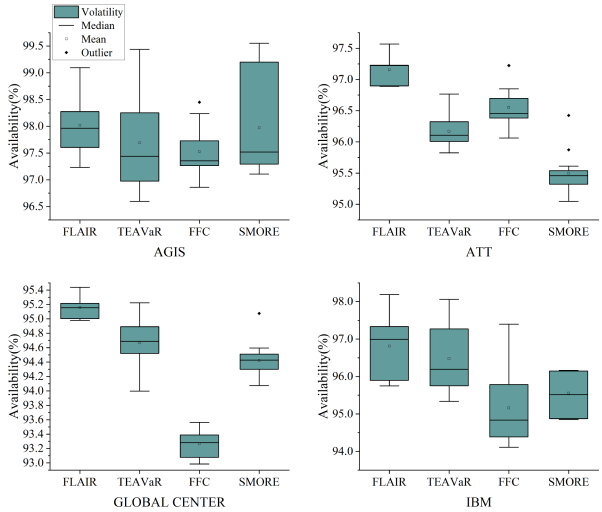


Fig. 6. Availability performance over four algorithms on four Inter-DC networks.

with 48 Inter-DC links. Global Center and ATT are denser than the other two topologies.

- *Traffic profile:* The online traffic volume on each link for every interval was monitored and recorded from a big ISP's Inter-DC network in real time. Simulated offline traffic requirements were derived from the capacities and utilization of the links, with each request delineated as a set of six elements: demand identification, time of arrival, time due, volume of traffic, originating DC, and target DC. In order to align with practical scenarios, the generated offline traffic demands slightly exceeding the available capacity of the remaining link. Additionally, the arrival times, deadlines, originating and destination DC for offline traffic were determined in a random manner. Due to the need for confidentiality and privacy, the traffic dataset will not be made public.
- *Failure Information:* We generate failure information for each network topology in chronological order. For each link e, we build a failure probability array Ψ_e following Weibull distribution and in each epoch, we generate one random decimal $\psi_e \in [0, max\Psi]$ for e, and track e whether its failure probability becomes greater than ψ_e. Then, if this happens, the link e would fail (i.e., when $\Psi_e > \psi_e$). Note, in this mode, the larger network will be given more failures, which is in line with reality.

Algorithm library: We divide the failure recovery process into two phases: 1) reroute; 2) flow shuffle. In the first step, we develop the rerouting algorithms and start them whenever the network detects link failures. Right after that, we shuffle flows to recovery routes by solving the problem via linear programming.

Tracers: We designed a monitoring program written in C++ language, including a thread pool for evaluation. Each time failures occur, several tracer threads are triggered to log the performance of the algorithms, which includes time consumption, utilization, availability, and scale of traffic impacted by rerouting. Since tracer threads are parallel to the main process, it is promised to pose little overhead on the performance of the recovery algorithms.

B. Methodology

We compare FLAIR with three other solutions: TEAVaR, FFC_2, and SMORE. The recovery process of FLAIR (namely FRAVaR) introduces Value-at-Risk (VaR) in its TE mechanism, and strikes balance between throughput and availability. FFC_2 recovers the network from failures by assuming there are at most k failures occurring at once and reserving part of link capacity to be ready for rescheduling. In the experiments, FFC_2 refers to FFC with the assumption of at most 2 failures happening at the same time. SMORE prepares resilient route sets with Räcke's oblivious routing algorithm for traffic schedule and potential failures, which to some extent is similar to the idea of IR.

C. The Performance of FLAIR Framework

First, we evaluate the FLAIR framework as a whole, integrating both design components (i.e., FRAVaR and ROSE). We compare FLAIR with the aforementioned three other algorithms on four topologies. We evaluate each algorithm twenty times and record availability and utilization, during which we introduce maximum link utilization (MLU) to measure utilization. Under the same traffic demand, MLU reflects the network utilization level, which network is in a state of a higher overall utilization while MLU is less. Here the availability level refers to the proportion of satisfied traffic demand under the same failure frequency. In MLU evaluation, we reduce the traffic demand in case of 100% MLU, in which case the congestion occurs and affects evaluation results. The traffic demand on MLU evaluation absolutely stays the same.

Availability: As shown in Fig. 6, FLAIR achieves the best availability over four recovery algorithms, while the improvement is small. We observe that, in the four Inter-DC networks, FLAIR outperforms the other three algorithms by 1.14% on

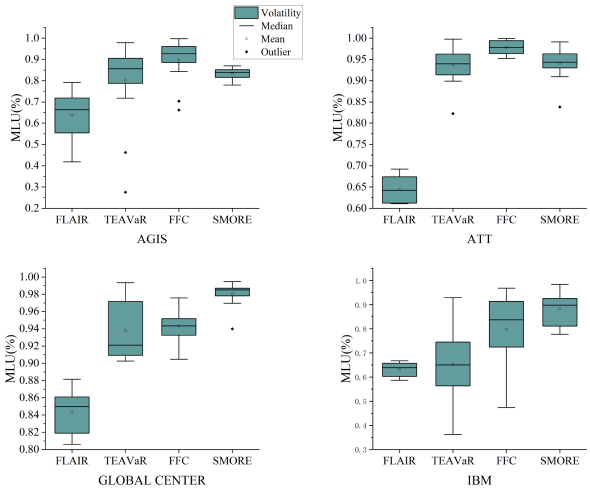
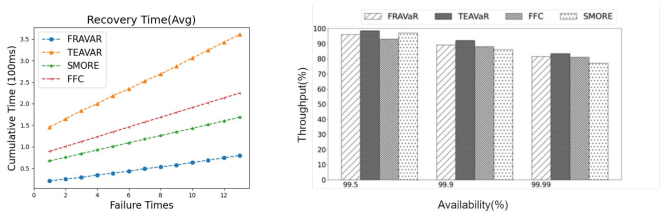


Fig. 7. Utilization performance over four algorithms on four Inter-DC networks.



(a) Comparison on recovery time with different failure times (average value under the 4 network topologies).

(b) Comparison on maximum throughput with different network availability requirement (average value under the 4 network topologies).

Fig. 8. Network throughput after failure occurrence.

average in terms of availability. This indicates that all four algorithms have achieved high performance on availability evenly. Nevertheless, the volatility of FLAIR is minuscule and there is no outlier in FLAIR, which means FLAIR is more stable in availability performance. In smaller Inter-DC networks (AGIS and IBM), the stability of FLAIR is inconspicuous. In larger networks (ATT and Global Center), the availability of FLAIR is significantly higher and more stable. This shows that FLAIR achieves good scalability in terms of availability.

Utilization: Fig. 7 shows the MLU evaluation results. As discussed below, MLU reflects the average utilization level of the network and a lower MLU indicates a better utilization performance. FLAIR reduces the MLU by 36.0% at most compared to the other algorithms on the four Inter-DC network simulations. Simultaneously, FLAIR has 0 outlier points showing high stability on performance, whereas other algorithms all have outliers (TEAVaR and FFC in AGIS, SMORE in Global Center). In AGIS and IBM simulations, the MLU performance of four algorithms is close, while in ATT and Global Center, the upsides of FLAIR are still significant, which also shows excellent scalability. Moreover, in larger Inter-DC networks, FLAIR distributes traffic more evenly to each link.

D. The Performance of FRAVaR

Recovery time: As mentioned above, we start the traffic flow in the Inter-DC network and then generate the link failure events following a Weibull distribution for each network topology. Each time failures occur, we record the consumed time between failure occurrences until all rescheduled tasks are complete. We also start a tracer to follow these rescheduled traffic flows. Then, we accumulate the consumed time by the order of failures and show the average time consumption in Fig. 8(a). The changing amplitude of the gradient of accumulation is not large in the four algorithms so it is obvious that the time consumption of each epoch is broadly consistent. TEAVaR is the most time-consuming algorithm overall and after TEAVaR, FFC_2 and SMORE are separately the second and third time-consuming algorithms, while FRAVaR consumes the least time, which is attributed to the effectiveness of FRAVaR's incremental routing algorithm.

Throughput and availability: We calculate the network availability with a post-processing simulation [8] from which we generate scenarios of the current network state. For each scenario, we try to send the entire demands through the network and record whether the network can satisfy the demands. The ratio of the unsatisfied demand with respect to the total demand reflects the availability level of the network. We present the average throughput and availability level into one plot and show them in Fig. 8(b). With the increase in availability level requirements, the throughput of all four methods evaluated dropped. TEAVaR achieves the best balance between throughput and availability while the network throughput in the case of SMORE drops significantly. In contrast, the performance of both FRAVaR and FFC_2 are relatively stable and comparable to TEAVaR. Generally, we note that with minor trade off of the throughput and availability, FRAVaR achieves significantly faster recovery process.

Traffic shuffle scale: Next, we use a micro-benchmark to evaluate the scale of shuffle traffic among the algorithms. Upon failure, we track and note the variation in traffic for each link pre and post-rescheduling. Consider link e, where traffic spikes from 5 Gbps at failure to 10 Gbps post-recovery, reflecting a 5GB increment. We accumulate the change on each link by order of epoch and refer to it as the rerouted traffic. See Fig. 9 for the experimental results, where the x-axis is the failure times order by epoch and the y-axis is the cumulative scale of rerouted traffic. Over the four network topologies, FRAVaR affects the least amount of traffic during the rescheduling while the impact of TEAVaR and FFC_2 is noticeably more than the other two algorithms. The results support our view that the recovery time positively correlates with the rerouted traffic scale.

In networks of large topologies like ATT and AGIS, the superiority of FRAVaR is highlighted. In these topologies, the three algorithms other than FRAVaR shuffle more traffic while FRAVaR can maintain low levels of rescheduling scale. In contrast, in small topologies, like GlobalCenter, the advantage of FRAVaR is not significantly obvious. These observations demonstrate that a recovery algorithm is more likely to impact more traffic during rescheduling when it covers large Inter-DC

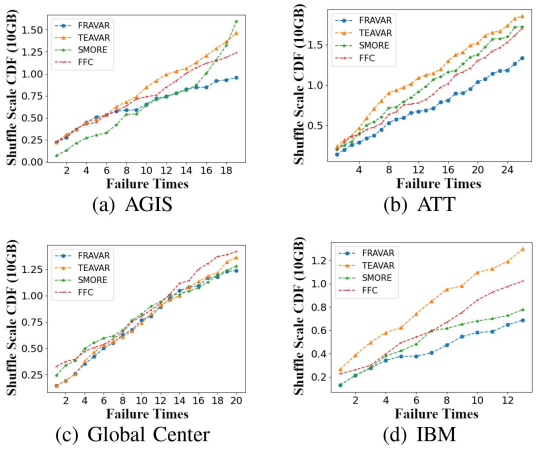


Fig. 9. Comparison of the scale of traffic rerouted by four failure recovery algorithms over four network topologies. The x-axis is the failure times order by epoch and the y-axis is the cumulative scale of traffic rerouted (metric: 10 GB).

networks and so in larger networks, the impact will be more pronounced. In such cases, FRAVaR maintains stable degrees of impact thanks to IR, which prunes reroute space into a stable scale regardless of network size. The network size impact on rerouted traffic scale is universal. Therefore, the difference in shuffle traffic scale over the four algorithms, in small topologies, is smaller than that of the large topologies.

E. The Performance of ROSE

We use another micro-benchmark to evaluate the long-term impact of failure recovery and the benefits of ROSE in reducing the failure impact. We set FRAVaR as the baseline to evaluate the performance of ROSE and analyze the role of routing optimization after failure recovery. Throughout the micro-benchmark, we set a failure frequency to trigger ROSE and optimize the routing of the whole network. We use a counter to record the number of failure occurrences and whenever it reaches the predetermined frequency (or threshold), the counter clears the record and starts ROSE. We adjust the preset failure frequency trigger over the four network topologies to evaluate the performance of ROSE in different environments. Finally, we take the average value as the performance result under the corresponding failure frequency and network topology.

Availability: Similar to Section VI-D, we generate all possible scenarios from the current network state when it completes either the failure recovery or the operations of ROSE. The availability level is the proportion of satisfied traffic demands. Fig. 10(a) shows the improvements in availability from ROSE, where x-axis is the preset failure frequency trigger and y-axis is the evaluated network topology. The availability performance of ROSE is represented as the improvement compared to the baseline. We observe that the routing optimization of ROSE provides at most 13.3% availability improvement and at least 2.1%, while the improvement is 3.7% on average. In general, FRAVaR as a baseline already provides a high-level of availability which is close to the availability level of the global optima. Nevertheless, there are still interesting results behind these evaluations. We note that, with the increase of trigger failure frequency, the improvement of ROSE increase as well. It reflects the fact that routing redundancy resulting from failure recovery keeps accumulating and ROSE's benefits become prominent. With the increase of link/node ratio (as shown in Table I), ROSE provides a better performance, which means that ROSE achieves good scalability because it brings higher gains in larger Inter-DC networks.

Utilization: We introduce maximum link utilization (MLU) as the network utilization indicator and evaluate the MLU difference before ROSE is invoked and after ROSE finishes its task. Fig. 10(b) shows the MLU reduction from ROSE. ROSE provides up to 46.4% utilization improvements and 37.3% on average. With the increase in failure frequency, the improvements in the utilization performance from ROSE increases. This indicates that routing redundancy caused by failure recovery results in an additive effect and ROSE is able to eliminate this impact. Similarly, with the increase in the link-to-node ratio, ROSE provides a higher improvement in terms of the MLU. We note two vital results: 1) in larger and denser networks, failure recovery would cause more severe routing redundancy, and 2) with the decrease of available links, the performance of ROSE becomes closer to that of the baseline failure recovery (i.e., FRAVaR).

Overhead: There is still an important part of ROSE. SE theoretically reduces routing deployment overhead by reducing unnecessary routing updates. We set a tracker to evaluate the proportion of routing rules that SE rejects to update during the ROSE process and show it in Fig. 10(c). We find that SE reduces routing deployment overhead by up to 48.2% and by 25.4% on average. Meanwhile, with the ascending link-to-node ratio, SE provides a higher reduction in deployment overhead, which means improved scalability. This indicates that in a large network topology, the difference between the routing of failure recovery and the globally optimized routing is minor. With the ascending of preset failure frequency, the improvement in deployment overhead from SE declines. This shows that even though, routing redundancy accumulates with the increase in failure recovery, there is a decline in the best-state routing.

VII. Related Work

In this section, we introduce and discuss the related work.

A. Failure Recovery

Existing failure recovery work can be roughly divided into proactive and reactive solutions. Proactive work implements recovery mechanisms before failure by reserving network resources. SMORE [6], which comprehensively considers two aspects of traffic engineering. Specifically, it uses R $\ddot{a}$ cke's oblivious routing algorithm [23], [24] to select a low extension, diversified, and load-balanced path set and dynamically adjusts the transmission rate [25], [26], [27], [28]. Considering only the availability of the network, TI-LFA [18] proposes an ideal backup and recovery scheme by preparing backup routes and switching to the backup link directly in case of failures, which undoubtedly makes the failures recovery very fast. However, the

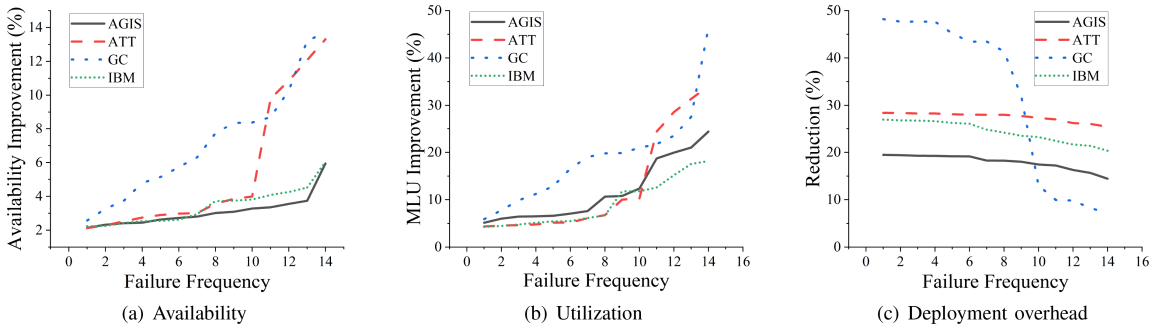


Fig. 10. Improvement of ROSE compared with FRAVaR.

main bottleneck is the time cost of backup computation which could cost several hours. R3 [19] adds a virtual demand on each available link and reserves bandwidth resources for recovery which may be used in case of link failures. FFC [5] ensures the stability of failure recovery through redundant links to increase efficiency. CFR-RL [29] considers the negative effects of rerouting and from the perspective of minimizing the rerouting probability, it leverages Reinforcement Learning (RL) method to identify key flows. Sentinel [30] introduces an optimized LP process [31], [32] into recovery but it introduces undesirable redundancy. Suchara et al. [33] flexibly split traffic flows over multiple backup routes.

Reactive work makes recovery decisions right after failure and it guarantees a higher transmission performance when the network is normal. TEAVaR [8] focuses on the balance of network utilization and availability. It reduces the rerouting probability by leveraging VaR [13] to formulate a loss function for evaluating the failure risk. TEAVaR also uses the decision tree pruning method to simplify the rerouting. Its traffic scheduling method inspired our algorithm in this paper. BlastSheild [4] slices a large network into tidy domains and recovers within a small range, which is somehwat similar to IR but the size of domains in BlastSheild is fixed. LFA-FRR [34], [35] avoids the formation of loops during routing recovery. MPLS TE FRR [36] introduces MPLS in failure recovery in order to accelerate the recovery process.

B. Routing Optimization

Routing optimization, a long-standing topic with various custom solutions, falls into non-heuristic, heuristic, or machine learning categories. Traditional non-heuristic methods, such as BFS, DFS, and Dijkstra [36], though straightforward, do not adequately address redundancy minimization or failure recovery.

Similarly, some of the existing heuristic methods are based on path planning algorithms, like A-Star and D-Star [37] which inherit the same spirit of non-heuristic methods (e.g., Dijkstra). Follow-up efforts were dedicated to overcoming their drawbacks. For instance, to achieve load balancing and high availability with limited TCAM space, Xu et al. [38] considered TCAM capacity as an LP constraint and proposed approximate methods for path-finding. Another work leveraged genetic algorithms to collect network information and proposed GA-TBR [39] to select routes which guarantee the Quality of Service (QoS) of wireless sensor networks. Moreover, the minimum energy-cost problem was used as a proxy and Lu et al. [40] proposed a two-stage linear programming solution based on a min-cost multi-commodity flow model.

Finally, most machine Learning methods aim to support routing algorithms by reducing the deployment overhead. For example, Deep Extreme Learning Machine (DELM), commonly used in classification, is leveraged to propose ISDN [41] which reduces the discreteness in Cognitive Routing Optimization (CRO). Another work leveraged the multi-agent meta-reinforcement learning method and proposed MAPPO [42] to achieve an adaptive routing optimization problem optimization for traffic engineering under fixed and variable demands.

These works focus mainly on distributing the traffic efficiently across the network, reducing the likelihood of congestion and overcoming potential bottlenecks and overheads. While these works may not directly address route redundancy nor failure recovery, they partly contribute to solving these issues and provide a building block and inspiration for algorithms such as FLAIR which can help efficiently address these issues.

VIII. Conclusion

In this paper, we discuss the importance and challenges of fast recovery with low deployment overhead in Inter-DC network. To this end, we introduce a two-phase solution FLAIR comprised of two key components, namely FRAVaR and ROSE. FRAVaR cleverly selects fewer links to reroute with an incremental rerouting algorithm IR, and then it formulates a flow shuffle schedule by converting it into a traffic scale reduction problem with Value-at-Risk theory. ROSE aims to reduce the redundancy introduced by FRAVaR. To achieve this, State Estimation (SE) theory is used to perform routing optimization and reduce routing updates. The evaluation results show that FLAIR not only reduces the deployment overhead by 25.4% but also outperforms the state-of-the-art algorithms by reducing recovery time by 70% on average while saving 29.8% shuffle flow scale. FLAIR is also shown to achieve comparable performance on both network throughput and availability.

REFERENCES

[1] C.-Y. Hong et al., "B4 and after: Managing hierarchy, partitioning, and asymmetry for availability and scale in Google's software-defined WAN," in *Proc. Conf. ACM Special Int. Group Data Commun.*, 2018, pp. 74–87.

[2] S. Hu et al., "Aeolus: A building block for proactive transport in datacenters," in *Proc. Annu. Conf. ACM Special Int. Group Data Commun. Appl. Technol. Architectures Protoc. Comput. Commun.*, 2020, pp. 422–434.

[3] S. Jain et al., "B4: Experience with a globally-deployed software defined WAN," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 3–14, 2013.

[4] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized cloud wide-area network traffic engineering with BLASTSHIELD," in *Proc. 19th USENIX Symp. Netw. Syst. Des. Implementation*, 2022, pp. 325–338.

[5] H. H. Liu, S. Kandula, R. Mahajan, M. Zhang, and D. Gelernter, "Traffic engineering with forward fault correction," in *Proc. ACM Conf. SIGCOMM*, 2014, pp. 527–538.

[6] P. Kumar et al., "Semi-oblivious traffic engineering: The road not taken," in *Proc. 15th USENIX Symp. Netw. Syst. Des. Implementation*, Renton, WA: USENIX Association, 2018, pp. 157–170. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/kumar

[7] M. Englert and H. Räcke, "Oblivious routing for the lp-norm," in *Proc. 50th Annu. IEEE Symp. Found. Comput. Sci.*, 2009, pp. 32–40.

[8] J. Bogle et al., "TEAVAR: Striking the right utilization-availability balance in WAN traffic engineering," in *Proc. ACM Special Int. Group Data Commun.*, 2019, pp. 29–43.

[9] P. He, W. Zhang, H. Guan, K. Salamatian, and G. Xie, "Partial order theory for fast TCAM updates," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 217–230, Feb. 2018.

[10] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 123–137.

[11] G. Zhou et al., "Primus: Fast and robust centralized routing for large-scale data center networks," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[12] H. Huang et al., "FRAVaR: A fast failure recovery framework for inter-DC network," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2023, pp. 1–6.

[13] Rockafellar, Journal of risk. Uryasev S, ch. Optimization of conditional value-at-risk, 2000, pp. 21–42.

[14] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 523–536, 2015.

[15] K. He et al., "Measuring control plane latency in SDN-enabled switches," in *Proc. 1st ACM SIGCOMM Symp. Softw. Defined Netw. Res.*, 2015, pp. 1–6.

[16] X. Wen et al., "RuleTris: Minimizing rule update latency for TCAM-based SDN switches," in *Proc. IEEE 36th Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 179–188.

[17] Y. Wan et al., "T-cache: Dependency-free ternary rule cache for policy-based forwarding," in *Proc. IEEE Conf. Comput. Commun.*, 2020, pp. 536–545.

[18] C. Filsfils, K. Michielsen, and K. Talaulikar, *Segment Routing Part I*. Scotts Valley, CA, USA: CreateSpace Independent Publishing Platform, 2017.

[19] Y. Wang et al., "R3: Resilient routing reconfiguration," in *Proc. ACM SIGCOMM Conf.*, New York, NY, USA, 2010, pp. 291–302, doi: 10.1145/1851182.1851218.

[20] J. Zhou et al., "WCMP: Weighted cost multipathing for improved fairness in data centers," in *Proc. 9th Eur. Conf. Comput. Syst.*, 2014, pp. 1–14.

[21] I. G. Optimization et al., "Gurobi optimizer reference manual, 2018," 2018. [Online]. Available: http://www.gurobi.com

[22] T. U. of Adelaide, "The internet topology zoo," 2011. [Online]. Available: http://www.topology-zoo.org/

[23] H. Racke, "Minimizing congestion in general networks," in *Proc. IEEE 43rd Annu. Symp. Found. Comput. Sci.*, 2002, pp. 43–52.

[24] H. Räcke, "Optimal hierarchical decompositions for congestion minimization in networks," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, 2008, pp. 255–264.

[25] A. Kumar et al., "BWE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proc. ACM Conf. Special Int. Group Data Commun.*, 2015, pp. 1–14.

[26] C.-Y. Hong et al., "Achieving high utilization with software-driven WAN," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, 2013, pp. 15–26.

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

[28] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 253–264, 2005.

[29] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "CFR-RL: Traffic engineering with reinforcement learning in SDN," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 10, pp. 2249–2259, Oct. 2020.

[30] J. Zheng, H. Xu, X. Zhu, G. Chen, and Y. Geng, "We've got you covered: Failure recovery with backup tunnels in traffic engineering," in *Proc. IEEE 24th Int. Conf. Netw. Protoc.*, 2016, pp. 1–10.

[31] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discrete Math.*, vol. 13, no. 4, pp. 505–520, 2000.

[32] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," *J. Comput. Syst. Sci.*, vol. 50, no. 2, pp. 228–243, 1995.

[33] M. Suchara, D. Xu, R. Doverspike, D. Johnson, and J. Rexford, "Network architecture for joint failure recovery and traffic engineering," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 39, no. 1, pp. 97–108, 2011.

[34] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote loop-free alternate (LFA) fast reroute (FRR)," IETF RFC 7490, Apr. 2015. [Online]. Available: www.rfc-editor.org/rfc/rfc7490

[35] C. Filsfils et al., "Loop-free alternate (LFA) applicability in service provider (SP) networks," IETF RFC 6571, Jun. 2012. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6571.txt

[36] O. Lemeshko and O. Yeremenko, "Linear optimization model of MPLS traffic engineering fast reroute for link, node, and bandwidth protection," in *Proc. 14th Int. Conf. Adv. Trends Radioelecrtronics Telecommun. Comput. Eng.*, 2018, pp. 1009–1013.

[37] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 1, pp. 36–56, First Quart., 2008.

[38] S. Xu, X. Wang, G. Yang, J. Ren, and S. Wang, "Routing optimization for cloud services in SDN-based Internet of Things with TCAM capacity constraint," *J. Commun. Netw.*, vol. 22, no. 2, pp. 145–158, 2020.

[39] U. Baroudi, M. Bin-Yahya, M. Alshammari, and U. Yaqoub, "Ticket-based QoS routing optimization using genetic algorithm for WSN applications in smart grid," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 4, pp. 1325–1338, 2019.

[40] X. Lu, F. Kong, X. Liu, J. Yin, Q. Xiang, and H. Yu, "Bulk savings for bulk transfers: Minimizing the energy-cost for geo-distributed data centers," *IEEE Trans. Cloud Comput.*, vol. 8, no. 1, pp. 73–85, First Quarter, 2020.

[41] F. Alhaidari et al., "Intelligent software-defined network for cognitive routing optimization using deep extreme learning machine approach," *Comput. Materials Continua*, vol. 67, no. 1, pp. 1269–1285, 2021.

[42] L. Chen, B. Hu, Z.-H. Guan, L. Zhao, and X. Shen, "Multiagent meta-reinforcement learning for adaptive multipath routing optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5374–5386, Oct. 2022.

Yuchao Zhang (Member, IEEE) received the BS degree in computer science and technology from Jilin University, in 2012, and the PhD degree from the Department of Computer Science, Tsinghua University, in 2017. She is currently an associate professor with the Beijing University of Posts and Telecommunications, Beijing, China. Her research interests include large scale datacenter networks, blockchain, federated learning, data privacy, and edge computing. She is a member of ACM.

Haoqiang Huang (Member, IEEE) received the BE and MSc degrees from the Beijing University of Posts and Telecommunications, China. He is currently engaged in research and development in the financial industry, focusing on networks and data. His current research interests include private cloud data centers, ERP systems, and trusted computing technologies.

Ahmed M. Abdelmoniem (Member, IEEE) received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2017. He is an assistant professor with the Queen Mary University of London, U.K. and Assuit University, Egypt. Formerly, he was a research scientist with KAUST, Saudi Arabia and a senior researcher with Huawei's Future Networks Lab, Hong Kong. He is an investigator on projects totalling USD 1.5mil in funding. His research interests lie in the intersection of distributed systems, networks and machine learning. His work appears in top-tier conferences and journals including NeurIPS, AAAI, MLSys, ACM EuroSys, IEEE INFOCOM and ICDCS, *IEEE/ACM Transactions on Networking*, *IEEE Internet of Things Journal*, *IEEE Transactions on Information Forensics and Security* and *Elsevier Computer Networks*, *Future Generation Computer Systems*, and *Computer Communications*. He is a member of ACM and USENIX.

Gaoxiong Zeng (Member, IEEE) received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology, under the supervision of Prof. Kai Chen. He is a scientific researcher with Network Technology Lab of Huawei. His research focuses on computer networks and systems. His work has been published on various top venues such as SIGCOMM, NSDI, INFOCOM, ICNP, and *IEEE/ACM Transactions on Networking*, etc.

Chenyue Zheng (Student Member, IEEE) received the BE degree from the Beijing University of Posts and Telecommunications, China. He is a currently working toward the master's degree with the State Key Laboratory of Networking and Switching Technology. His current research interests include next generation network architecture and network systems.

Xirong Que (Member, IEEE) received the BE and ME degrees from the Beijing University of Posts and Telecommunications, China, in 1993 and 1998, respectively. She is currently an associate professor with the State Key Laboratory of Networking and Switching Technology. She has published more than 60 papers in various journals and conference proceedings. She is current research interests include next generation network architecture, network resources management and QoS.

Wendong Wang (Member, IEEE) received the BE and ME degrees from the Beijing University of Posts and Telecommunications, China, in 1985 and 1991, respectively. He is currently a full professor with the State Key Laboratory of Networking and Switching Technology. He has published more than 200 papers in various journals and conference proceedings. His current research interests include next generation network architecture, network resources management and QoS, and mobile internet.

Ke Xu (Fellow, IEEE) received the PhD degree from the Department of Computer Science and Technology, Tsinghua University. He is currently a full professor with Tsinghua University. His research interests include next generation internet, P2P systems, the Internet of Things, network virtualization, and network economics. He is a member of ACM. He serves as an associate editor for *IEEE Internet of Things Journal*. He has guest edited several special issues in IEEE and Springer journals.