FRAVaR: A Fast Failure Recovery Framework for Inter-DC Network

Haoqiang Huang[†*], Yuchao Zhang✉[†*], Ran Wang[†*], Qiao Xiang[††], Wendong Wang[†*], Xirong Que[†*], and Ke Xu[‡]

[†]Beijing University of Posts and Telecommunications, Beijing, China
[*]With the State Key Laboratory of Networking and Switching Technology
[††]School of Informatics, Xiamen University, China
[‡]Department of Computer Science and Technology, Tsinghua University, China

***Abstract*—Along with the development of 5G and IoT technologies in recent years, Inter Data Center (Inter-DC) network is facing an explosive growth of geographically distributed user data, which needs to be duplicated among DCs in a real-time manner. Transmission-based applications require high availability that is going beyond 99.99%. However, with the expansion of Inter-DC network scale, link failures are also growing, which seriously affects data transmission efficiency, so fast link failure recovery is then urgently needed. Many previous works have been done to achieve fast failure recovery, but most of them ignore two key points, 1) the cost of deploying recovery strategies, and 2) the side-effect of re-transmission to network availability. These two factors make the existing failure recovery process too slow to be practical in real-time online industrial environments. To achieve realistic fast recovery from Inter-DC network failures, we propose a failure recovery framework FRAVaR, which achieves high network availability with very little deployment overhead. Particularly, FRAVaR reduces the deployment overhead by a novel incremental routing strategy to isolate link failures. In other words, it only needs to shuffle a tiny amount of traffic within a small failure isolation domain. On this base, FRAVaR further adopts a risk assessment theory named Value-at-Risk (VaR) to control flow re-transmission. We implement a prototype of FRAVaR and conduct a series of experiments on 4 real Inter-DC network topologies (ATT North America, IBM, GlobalCenter, AGIS). Experiment results show that FRAVaR outperforms state-of-the-art solutions on the recovery speed by 70.2%.** [1]

***Index Terms*—Inter data center network, Failure recovery, Value at Risk**

I. Introduction

In recent years, user data generated from edge access networks has explosively grown and needs to be transmitted among geographically distributed data centers (DCs) in a nearly real-time manner. In order to keep pace with the 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 the data traffic which has increased by 100 times in five years [2]. At the same time, transmissions of this large amount of data require extremely high availability that goes beyond 99.99% [1], because even one second of network breakdown would result in a loss of 100GB-level network traffic and would be instantly observed in a global range [3]. In order to achieve high availability in such large Inter-DC networks, large ISPs (e.g. Google [1], Microsoft [4]) leverage Software-Defined Networking (SDN) in management and optimization.

With the continuous growth of the Inter-DC network scale, link failures occur more frequently in actual networks. 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%.

Inter-DC networks need a fast link failure recovery framework to improve availability by coping with such high failure frequency. Many previous works have already made considerable achievements in recovering from link failures. [6]–[13] Most recovery solutions (e.g. FFC [5], SMORE [9] and TEAVaR [6]) prepare backup network resources for recovery while sacrificing part of bandwidth. Others devote to reducing traffic loss in failures and re-transmit flows after failures. Nevertheless, the previous work neglects the cost of deploying rerouting strategy. While a measurement in [14] shows that updating routing rules do take a certain amount of time that cannot be ignored. For example, updating only 20K rules introduces millisecond-level latency on average. In nowadays 200Gbps Inter-DC network, such an update would delay the transmission of megabyte traffic. Meanwhile, the deployment of rerouting strategies always results in a large number of re-transmissions, which finally leads to slow failure recovery.

Designing a fast recovery framework is never an easy task due to the following two challenges. First, recovery solutions with globally optimal rerouting strategies are usually accompanied by a large amount of routing rules updates [14], which makes those solutions suffer from long deployment time on routers. Second, the failure recovery process would bring a lot of flow re-transmissions [8] (we call these flows "shuffled" flows), especially in large Inter-DC networks. During the rerouting process, the controller needs to carefully decide

✉ Corresponding author, E-Mail: yczhang@bupt.edu.cn

[1]The work was supported in part by the National Key R&D Program of China under Grant 2019YFB1802603, the National Natural Science Foundation of China(NSFC) under Grant 62172054 and 62072047, the Key Project of Beijing Natural Science Foundation under M21030, and the ByteDance Grant under S2022054.

the update order of routers to avoid out-of-order packets, congestion, and packet loss [15], while calculating order also consumes time. What's worse, connection rebuilding, transmission, and congestion control, all of these processes consume so much time that some works reroute in free time [16], and consequently, recovery methods should be able to solve all the above problems.

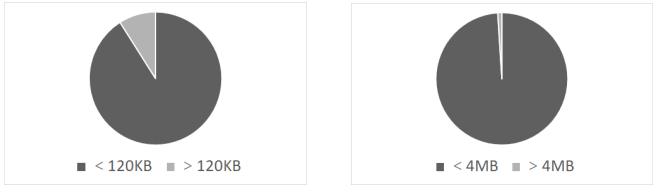
In this paper, we propose *FRAVaR*, a framework that achieves fast failure recovery while still maintaining comparable throughput and link utilization performance with state-of-the-art failure recovery solutions. To speed up the implementation of rescheduling strategy and shorten the time of router reconfiguration, we propose an incremental reroute algorithm *IR*, which reduces the amount of re-transmission flows by isolating link failures within a limited range of network and carefully selecting available links to replace the failed ones. Thus, IR substantially reduces the number of routers that need to be reconfigured and the number of routing rules to update, and therefore accelerates the deployment of our rerouting strategy. On this base, we further introduce the Value at Risk theory (*VaR*) [7] to guide flow shuffles, which makes FRAVaR more inclined to shuffle flows to low failure probability links. Through VaR, we formulate the traffic loss in rerouting paths and minimize it through linear programming (LP). Finally, VaR reduces the amount of failed flows and re-transmissions and further accelerates the recovery process.

The **main contributions** of this paper are as follows: 1) We disclose two essential factors to fast failure recovery: the deployment overhead of rerouting strategy and the affected scale of flow retransmissions. 2) We propose a fast failure recovery framework FRAVaR, where an incremental reroute algorithm IR isolates link failures within a small domain and a VaR-based flow shuffle algorithm shrinks re-transmission scale. 3) We implement a prototype of FRAVaR on four real Inter-DC topologies and the results show that it reduces about 70.2% recovery time compared with existing solutions.

The remainder of this paper is organized as follows. We discuss the situation and challenges of recovery in Inter-DC network and give a motivating example in Section II to show the necessity of designing FRAVaR. We introduce the specific algorithm model in Section III. We then show experimental results of FRAVaR and other recovery solutions in Section IV. Finally, we conclude the paper in Section VI.

II. Motivation

We discover, although a little bit counterintuitive, that deployment overhead becomes one of the bottlenecks to fast link failure recovery. Traditional deployment of routing strategy runs like this: SDN controller sends out new routing rules to routers that new routing paths traverse, and these routers update new routing rules into their routing table. But at the same moment, there are thousands of flows on the paths and they have to wait for updating. When failure occurs, the network stops transmitting these flows, deploys a new routing strategy, rebuilds connections, and restarts transmission. Over this process, the problem is that updating is no more fast



(a) FBHadoop (b) WebSearch

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

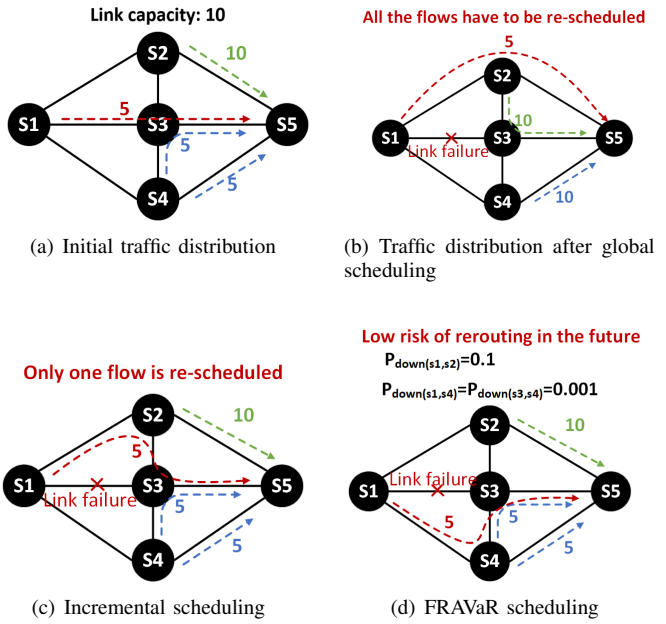
compared to the high speed of Inter-DC networks and re-transmission consumes too much time. To locate the next hop in routing tables within a fraction of a millisecond, Ternary Content Addressable Memory (TCAM) is broadly installed in routers as the proprietary memory to store routing tables for its concurrent and efficient searching structure. 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 [17] in Figure1(a) and WebSearch [18] in Figure1(b). More than 90% of flows are less than 120KB in FBHadoop and approximately 99% of flows in WebSearch are within 4MB. That is to say, in 400G Inter-DC networks, almost every flow finishes transmission within 0.1 milliseconds, while the time cost of updating 20K routing rules is a hundredfold. And that is what hinders a fast link failure recovery.

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 RTT and also CPU resources. In this case, CPU resources that assign to transmission sharply rise up, and CPU resources for other works are occupied. As a result, the performance of both network and those applications dependent on the hosts, like distribution machine learning, drops down. And that is the reason why the scale of re-transmission flows matters a lot in network performance.

A. A Motivating Example

We illustrate our motivation in the example shown in Figure2. Figure 2(a) shows an initial traffic distribution with four flows, f_1: s_1-s_3-s_5, f_2: s_2-s_5, f_3: s_4-s_3-s_5, and f_4: s_4-s_5. The dashed curves represent flows and numbers represent the traffic volume they carry. When link s_1-s_3 fails, a globally optimal recovery strategy adjusts flow assignment like Figure 2(b), i.e. f_1 to s_1-s_2-s_5. This strategy requires updating all the flows in this network to deploy its rescheduling result, which increases the amount of re-transmission flows.

However, if we adjust f_1 to s_1-s_2-s_3-s_5, as shown in Figure 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 Figure2(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. And it isolates these link failures within only one path s_1-s_3-s_5, making it unable to affect other flows outside this range.



(a) Initial traffic distribution

(b) Traffic distribution after global scheduling

(c) Incremental scheduling

(d) FRAVaR scheduling

Figure 2. Network Example of Recovery Rerouting

While Figure2(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 Figure2(d), if the failure probability of link $s1$-$s2$ is 0.1, while that of link $s1$-$s4$ and $s4$-$s3$ are both 0.001, there would be a great risk that the scheduling results in Figure2(c) is unstable and f_1 is more likely to fail again and has to be rescheduled in the future again. But see the assignment in Figure2(d), which enjoys a more stable path, it shuffles f_1 to s_1-s_4-s_3-s_5. This recovery schedule enjoys higher availability.

From the slightly artificial example, we illustrate that it is efficient for only part of network to participate in failure recovery while sacrificing a little bit of negligible performance impact. An incremental reroute brings: 1) fewer changes of routing rules and 2) less scale of shuffle traffic, which greatly speeds up the process the network recovers from failures and returns to a normal operation state.

B. Our Approach

Motivated by the above example, FRAVaR 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. FRAVaR tries to adjust the minimum number of transmissions that are already built up. Following this principle, we narrow the range of recovery to a local part of the network while still maintain the performance. Along the way, we propose IR, our incremental reroute algorithm. IR adjusts routing rules only on a short route bypassing the failed link and carefully selects rerouting paths by breadth-first-searching (BFS) algorithm. BFS promises completeness of recovery routes and we use visited tags to guarantee all the routes are disjoint. In this way, IR isolates link failures into a small domain of network. Based on IR, we then strengthen FRAVaR by introducing a minimum loss flow shuffle algorithm based on 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 then introduce the whole framework of FRAVaR in the next section.

III. FRAVaR DESIGN

In this section, we firstly introduce a network model and then FRAVaR's two-phase design.

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 that traverses node a, b, c and d is expressed as $r = (ab, bc, cd)$. 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 optimal performance in utilization and availability, it also slows down the processing speed of recovery and elongates the time delay. A little sacrifice of performance would extremely improve recovery speed. Here comes two design principles behind IR.

A trick to efficiency. In traditional TE, recovery algorithms are running on the whole network. Global failure recovery solutions can achieve good performance in many ways, but not all. To improve recovery efficiency, the scale of links chosen to recover the network needs to be pruned. 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 key point of our fast recovery solution is to reduce the exploring space while maintaining network performance.

Avoid SRLG. Shared Risk Link Group (SRLG) is one of the problems that reroute algorithms try to avoid. SRLGs are link groups that in a group, multiple routes go through the same link. We call them "share" the failure risk when they share the same link and whenever this shared link fails, each route that traverses this SRLG goes down and it is one of the most critical situations. The network has to recover multiple routes at the moment. As a solution, we enhance IR by visited tag and achieve a SRLG-disjoint reroute algorithm.

Here we introduce an incremental reroute algorithm of our fast recovery framework, IR, and show it in Algorithm 1. The input n is the amount of links of network G. IR scans the Inter-DC network G, collects failure links (e.g. ij), and designates a subset of network P as the rerouting paths. Firstly, IR puts source node i as an incomplete path into heap Q. Next, in every iteration, IR gets "top" path T from Q, adds an adjacent

node of T to acquire a new path T', and puts T' into Q if it has not reached destination node j. Whenever j is visited, IR pushes this path into result set P. IR ends when the amount of recovery paths has reached the limit n.

Moreover, IR introduces a visited tag array vis to mark the nodes when they are visited so that searches in later iterations refuse to select visited nodes. 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. 2) SRLG rerouting is avoided. Through visited tag and BFS, IR offers a rerouting path set P with mutually disjoint and shortest paths.

Here, IR introduces minimum heap Q to accelerate the searching process. We define a cost function f_k to sort heap, while the top of the heap is the node with minimal cost. Here we introduce a cost function to describe f_k, which considers both capacity and resilience.

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

Where h is the current node being traveled 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 of the selected paths.

Algorithm 1: Incremental Reroute

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;
        else
            push T' into Q;
            vis[lg] := true, ∀lg ∈ SRLG_hk
```

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 flow shuffle problem within this function. A natural solution to evaluate the expectation of flow loss is to simulate every possible scenario that every subset of surviving links fails, which time complexity is $O(2^N)$ where N is the number of links in network G. Obviously, it is unacceptable. Now, with IR narrowing the scale of network, the time complexity is reduced to $O(2^K)$ where K is the number of links in sub-network, and in most cases, we have $K \ll N$ and thus make a reactive computation of scenarios into practice.

Here we introduce the probability of scenarios. Firstly 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:

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

The probability p_q of scenario q to occur is naturally the probability of paths combinations as follows:

$$p_q = \prod_{ij \in E} (p_{ij} * y_{ij}^q) + (1 - p_{ij}) * (1 - y_{ij}^q) \tag{3}$$

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. 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:

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

Constraint (6) specifies that the whole flow allocated is not greater than the minimal link capacity of the path and it provides a minimum congestion control for network. Constraint (7) indicates that all the shuffle data of a flow is not less than its demand. So far, we have built up a fast failure recovery framework. Next we analyze the performance with evaluation.

IV. Evaluation

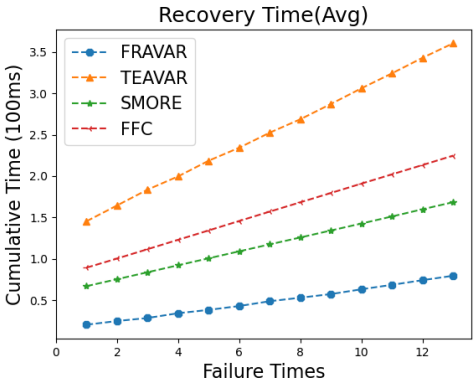
We design a simulation to evaluate the performance of FRAVaR and the other three algorithms on failure recovery: TEAVaR [6], FFC [5], and SMORE [9]. We adopt Gurobi [19] as the optimization framework.We develop a data-driven simulation software and it contains three components:

Topology. Our simulation includes four real Inter-DC network topologies: ATT, IBM, GlobalCenter, and AGIS, which are real Inter-DC network topologies from the internet topology zoo [20]. These networks cover situations from looseness to compactness. Table I shows the size of the networks.

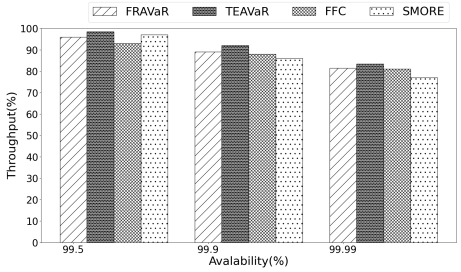
Traffic Information. We use the real traffic demand matrices from a big ISP's Inter-DC network and replay it on the above four Inter-DC topologies.

Table I
INFORMATION OF NETWORK TOPOLOGY.

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



(a) Cumulative summation of recovery time.



(b) Comparison on maximum throughput with different network availability requirement.

Figure 3. Average recovery performance under four topologies.

Failure Information. For each link, we build a failure probability array following Weibull distribution. It is noted that in this mode, the larger network will be given more failures, which is in line with reality.

A. FRAVaR Over Existing Solutions

Methodology. We compare FRAVaR with TEAVaR, FFC_2, and SMORE, where FFC_2 refers to FFC with the assumption that at most 2 failures happen simultaneously.

Recovery Time. We show the average consumption in Figure 3(a). The changing amplitude of 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 the attribution of IR.

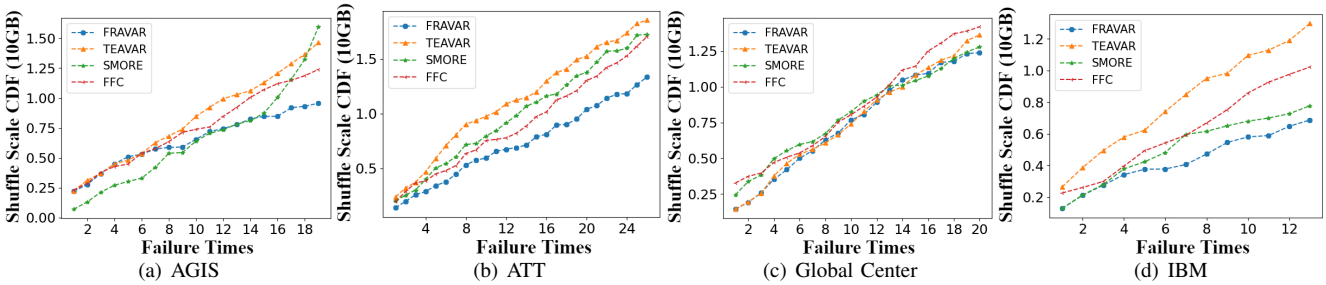
Throughput and availability. We calculate the network availability with a post-processing simulation [6] that we generate scenarios from the current network state and for each scenario we try to send the entire demands through the network and record whether the network can satisfy the demands. The ratio of unsatisfied scenarios reflects the availability. Then, we put the average throughput and availability into one graph and show it in Figure 3(b). With the improvement of availability requirement, throughput of the four methods all drop. TEAVaR achieves the best balance between throughput and availability while the throughput of network applying SMORE drops more. The performance of FRAVaR and FFC_2 is relatively stable. Generally, with a little sacrifice in the balance between throughput and availability, FRAVaR achieves a much faster recovery process.

B. Shuffle scale

Next, we use a micro-benchmark to evaluate the scale of shuffle traffic among the algorithms. We trace the traffic on each link when failures occur and record the difference between the traffic before rescheduling and after. For example, the traffic on link e is 5GB when failures occur and then the recovery algorithm starts to reschedule traffic. When reschedule finishes and the tasks have not been transmitted yet, the traffic on link e is 10GB. Then, the difference on link e in this recovery is 5GB. We accumulate the change on each link by order of epoch and refer to it as the rerouted traffic. See Figure 4 for the experiment results. In the network of large topology like ATT and AGIS, the superiority of FRAVaR is highlighted. In this topology, the other three algorithms similarly shuffle much traffic when FRAVaR still maintains a low-level scale of rescheduling. While in a small topology, like GlobalCenter, the advantage of FRAVaR is not especially obvious. The phenomena shows a plain fact that a recovery algorithm is more likely to impact more traffic during rescheduling when it covers a larger range of Inter-DC network and when in a larger network, the impact will be enlarged. While in such cases, FRAVaR holds the degree of influence thanks to IR, which prunes reroute space into a stable scale whatever how large the network is. The impact of the size of network on rerouted traffic scale is universal. The difference of shuffle traffic scale over four algorithms in small topology is smaller than of the larger topology.

V. RELATED WORK

Advanced TE work notice the importance of failure recovery and tried to address link failures from different aspects. TEAVaR [6] focuses on the balance of network utilization and availability and introduces VaR [7] to formulate the loss function to evaluate the failure risk, so as to reduce the rerouting probability. TEAVaR uses the decision tree pruning method to simplify rerouting process. Its traffic scheduling method inspired our algorithm in this paper. Similarly, SMORE [9], which comprehensively considers two aspects of traffic engineering, uses Räcke's oblivious routing algorithm to select a low extension, diversified and load-balanced path set, and dynamically adjusts the transmission rate.R3 [11] adds a virtual demand on each available link which may be rerouted on it when failures occur and reserves bandwidth resources for recovery. FFC ensures the stability of failure recovery through redundant links to increase efficiency. CFR-RL [10]



(a) AGIS (b) ATT (c) Global Center (d) IBM

Figure 4. Comparison of the scale of traffic rerouted by four failure recovery algorithms over four network topologies.

considers the negative effects of rerouting, like packet disorder and packet loss, and from the perspective of minimizing the rerouting probability, CFR-RL uses RL method to identify key flows. The fast failure recovery problem has been discussed for a long time. However, most of the work is on the network layer, like LFA-FRR [12] and MPLS TE FRR [13], which limits the probability to achieve the optimal performance over the network and reduces the risk of meeting overlapping failures.

VI. Conclusion

In this paper, we discuss the importance and challenges of fast recovery in Inter-DC network, and to reduce the deployment overhead, we introduce a two-phase solution FRAVaR, in which the first step is to cleverly select little links to reroute with an incremental rerouting algorithm IR, and then FRAVaR formulates flow shuffle by converting it into a traffic scale reduction problem with Value-at-Risk theory. Evaluation results show that FRAVaR outperforms the state-of-the-art algorithms in recovery time consumption down to 70% on average and save 29.8% shuffle flow scale, and at the same time, FRAVaR also achieves comparable performance on both network throughput and availability.

References

[1] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev *et al.*, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 74–87.

[2] S. Hu, W. Bai, G. Zeng, Z. Wang, B. Qiao, K. Chen, K. Tan, and Y. Wang, "Aeolus: A building block for proactive transport in datacenters," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 422–434.

[3] U. Krishnaswamy, R. Singh, N. Bjørner, and H. Raj, "Decentralized cloud wide-area network traffic engineering with {BLASTSHIELD}," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 325–338.

[4] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, "Achieving high utilization with software-driven wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 15–26.

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

[6] J. Bogle, N. Bhatia, M. Ghobadi, I. Menache, N. Bjørner, A. Valadarsky, and M. Schapira, "Teavar: striking the right utilization-availability balance in wan traffic engineering," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 29–43.

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

[8] Y. Zhang, X. Nie, J. Jiang, W. Wang, K. Xu, Y. Zhao, M. J. Reed, K. Chen, H. Wang, and G. Yao, "Bds+: An inter-datacenter data replication system with dynamic bandwidth separation," *IEEE/ACM Transactions on Networking*, vol. 29, no. 2, pp. 918–934, 2021.

[9] P. Kumar, Y. Yuan, C. Yu, N. Foster, R. Kleinberg, P. Lapukhov, C. L. Lim, and R. Soulé, "Semi-oblivious traffic engineering: The road not taken," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. Renton, WA: USENIX Association, Apr. 2018, pp. 157–170. [Online]. Available: https://www.usenix.org/conference/nsdi18/presentation/kumar

[10] J. Zhang, M. Ye, Z. Guo, C.-Y. Yen, and H. J. Chao, "Cfr-rl: Traffic engineering with reinforcement learning in sdn," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2249–2259, 2020.

[11] Y. Wang, H. Wang, A. Mahimkar, R. Alimi, Y. Zhang, L. Qiu, and Y. R. Yang, "R3: Resilient routing reconfiguration," in *Proceedings of the ACM SIGCOMM 2010 Conference*, ser. SIGCOMM '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 291–302. [Online]. Available: https://doi.org/10.1145/1851182.1851218

[12] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote loop-free alternate (lfa) fast reroute (frr)," *Internet Requests for Comments, RFC Editor, RFC*, vol. 7490, 2015.

[13] O. Lemeshko and O. Yeremenko, "Linear optimization model of mpls traffic engineering fast reroute for link, node, and bandwidth protection," in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*. IEEE, 2018, pp. 1009–1013.

[14] P. He, W. Zhang, H. Guan, K. Salamatian, and G. Xie, "Partial order theory for fast tcam updates," *IEEE/ACM Transactions on Networking*, vol. 26, no. 1, pp. 217–230, 2017.

[15] A. D. Ferguson, S. Gribble, C.-Y. Hong, C. Killian, W. Mohsin, H. Muehe, J. Ong, L. Poutievski, A. Singh, L. Vicisano *et al.*, "Orion: Google's {Software-Defined} networking control plane," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, 2021, pp. 83–98.

[16] M. A. Qureshi, Y. Cheng, Q. Yin, Q. Fu, G. Kumar, M. Moshref, J. Yan, V. Jacobson, D. Wetherall, and A. Kabbani, "Plb: congestion signals are simple and effective for network load balancing," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 207–218.

[17] A. Roy, H. Zeng, J. Bagga, G. Porter, and A. C. Snoeren, "Inside the social network's (datacenter) network," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 123–137.

[18] Y. Zhu, H. Eran, D. Firestone, C. Guo, M. Lipshteyn, Y. Liron, J. Padhye, S. Raindel, M. H. Yahia, and M. Zhang, "Congestion control for large-scale rdma deployments," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 523–536, 2015.

[19] I. G. Optimization *et al.*, "Gurobi optimizer reference manual, 2018," *URL http://www. gurobi. com*, 2018.

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