Analyzing Network Routing Resilience: A Hybrid Approach of Face and Tree Routing

Georgios Karamoussanlis
TU Dortmund
Dortmund, Germany
georgios.karamoussanlis@udo.edu

Stephanie Althoff
TU Dortmund
Dortmund, Germany
stephanie.althoff@udo.edu

Erik van den Akker
TU Dortmund
Dortmund, Germany
erik.vandenakker@udo.edu

Klaus-Tycho Foerster
TU Dortmund
Dortmund, Germany
klaus-tycho.foerster@udo.edu

***Abstract*—With the aim of ensuring reliable and consistent communication, network topologies need to be able to correctly and rapidly respond to potential errors. Through these responses, the connection within the network is maintained, even in the presence of faults. This is achieved through various mechanisms, known as fast failover algorithms, which are implemented in the data plane, opposed to the slower control plane.**

In this work, an established fast failover approach, which routes through trees, is extended. This extension is achieved by dividing the routing path into two parts and using both face routing and tree routing for forwarding packets. This is done against the backdrop that Tree Routing yields suboptimal results when the source of the routing is highly error-prone. To counteract this, this work proposes a modification of the Tree Routing by first routing through a waypoint using face routing and then utilizing tree routing. As such, the routing is resilient to errors both at the source and destination.

Results of this work showed in a setting of clustered failures using real world graphs, resilience can be enhanced, with a higher hop count due to utilizing the faces to circumvent early failures. Furthermore, it has been shown that for random errors in random graphs, the resilience is also increased compared to the original tree routing algorithm.

***Index Terms*—fast failover routing, routing, face routing**

I. Introduction and Background

In today's digital society, we increasingly rely on communication networks. Many applications, such as i.e. financial transactions or medical software, require a high availability of service. Network outages are causing a degradation in service quality [1] and can lead to high costs [2]. However, networks have been expanded significantly in size over the years, which leads to a corresponding increase of data traffic. Consequently, the likelihood of packets encountering failures during transmission has grown [3]. In the modern era, managing failures of communication networks has, therefore, become increasingly critical.

It is essential for networks to be equipped to handle node failures and to provide alternative paths ensuring that packets reach their intended destinations. Due to stringent availability requirements, local fast rerouting algorithms are used within the data plane in many communication networks [4], which are known to operate at timescales several orders of magnitude faster than the control plane [5].

Current methods for local fast failover algorithms, which are widely used in many networks today, primarily utilize edge-disjoint paths to create alternative routes [6], [7]. We refer to the survey of Chiesa et al. [4] for a general overview. Further research has shown that the use of tree routing can further enhance resilience [8]. In this approach, the edge-disjoint paths are extended into trees, which are then used for routing packets through the network. Trees are especially suited to this approach, as one can implement local fast failover routing in the nodes by traversing them in a depth-first search (DFS), where failed edges are simply *skipped* [9] in the local DFS traversal. Notwithstanding, using trees brings the disadvantage that routing may be prematurely terminated when frequent errors occur at the root, as the path diversity is concentrated at the destination neighboring the leaf nodes.

However, this approach cannot be directly turned into a double-tree structure, protecting both source and destination: when routing from a leaf to the destination, the packet never needs to return, as the packet has reached its intended target. But, when returning to the source, using the local DFS-traversal approach is problematic, as connecting the source to the tree turns the routing structure into a non-tree planar graph, where resilience to failures is hard to achieve [9], [10].

Still, upon taking a closer look, one does not need to perform arbitrary routing in this double-tree, as we are only interested in reaching the destination from the source, and hence we can take inspiration from classic face routing [11], [12]. In this setting, the graph is divided into so-called "faces" and routing is based on their geometric and geographic properties. Even though face routing usually requires non-trvial state [13] in the packets or nodes, which is undesirable in high-performance networks, we in this paper propose to simply route along the outer face to realize such a double-tree aproach, as routing along the outer face is compatible with zero-state local fast failover algorithms [9], [10].

A. Contributions

The present work analyzes whether the integration of face routing can improve fast failover routing in the form of tree routing. In this context, a demand first traverses from its source to a checkpoint using face routing and is then routed to the destination using tree routing. We denote this hybrid approach as $OneTreeCheckpoint$.

To this end, we present a formal concept how face routing can be integrated into the context of fast failover routing for the example of tree routing.

We show in Python-based simulations that such a hybrid approach can increase resilience to link failures, for both clustered failures in real-world graphs and random failures in random graphs. Nonetheless, the higher resilience comes at the cost of increased hop count, as the increased path diversity, leading to better routing success, comes at the cost of longer routes post-failures.

B. Organization

The work is structured as follows: Section II introduces the model and Section III discusses the concepts, including both the precomputation and routing processes. Section IV presents the evaluation, analyzing the resilience and hops of the various approaches. Finally, Section V summarizes the work and Section VI provides an outlook on further improvements that could be made to the hybrid use of face and tree routing.

II. Model

We consider networks modeled as undirected graphs $G = (V, E)$. For routing packets in the network, static local forwarding rules are implemented at each node for every possible combination of source $s \in V$ and destination $d \in V$. These forwarding rules decide about the used outgoing edge based on the information in the packet header, as well as the incoming port the packet was sent from. It is not allowed to modify the packet header to add additional information. The forwarding rules have to be installed at each node in advance (while the failures are unknown yet). After implementing the forwarding rules, a random set of edges $F \subseteq E$ fails, hence the packet has to be routed through the Graph $G' = (V, E \setminus F)$ only utilizing the previously installed forwarding rules.

Generally a forwarding pattern is called r-resilient, if for any set $F \subseteq E$ with $|F| = r$ and any source-destination pair $s, d \in V$, where s and d are still connected in G', packets from s can still reach the destination d while only following the rules of the forwarding pattern. In this work, we heuristically check if a packet still reaches the destination node under random edge failures, applying our strategy, and using resilience to describe the probability of the packet reaching its destination under a given failure set. A more detailed description of the used metrics is given in Section IV.

III. Concept and Model

In this section the concept of connecting face routing with tree routing is described in detail. Each step of Fig. 1 is referenced in the corresponding following sections. In the figure, arrows are shown under each step to indicate the direction in which each respective phase operates. The framework [14], which is based on previous work [15], has been expanded in this work. This is a corresponding model to [15], which describes the graph structures. A comparable model can also be found in the following work [8]. Therefore, the model of the trees used for routing in Sections III-A2 and III-B2 consists of unidirectional edges to maintain the internal structure of the trees and preserve parent-child relationships. The concept is encapsulated in the implementation of the

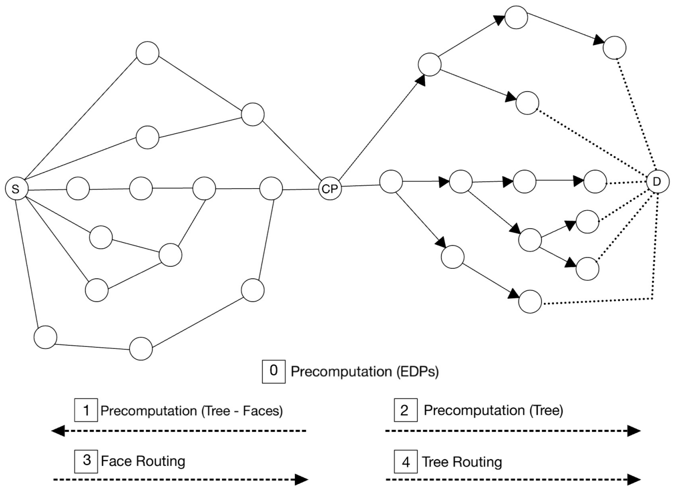

Fig. 1. Example graph demonstrating the step-wise implementation of the Hybrid Approach of Face and Tree Routing

$OneTreeCheckpoint$ algorithm, which will be detailed further in the following sections. In the other steps of the concept, the deployed structure consists of bidirectional edges.

A. Precomputation

In the precomputation phase, preparations are made and the underlying structures are created, which form the basis for the routers forwarding rules. This precomputation consists of three steps, which are shown in Fig. 1 as $\boxed{0}$, $\boxed{1}$, and $\boxed{2}$. The precomputation fills the entries of the routing tables by building the structures of each step for every source (s) and destination (d) pair.

1) EDPs: Before face and tree structures are created, the edge-disjoint paths (EDPs) need to be found in step $\boxed{0}$. For a given (s,d)-pair the edge-disjoint paths describe paths from s to d such that no two paths share an edge. First the longest EDP is found in order to choose the checkpoint (cp) as a safe node for routing through. In the current state of the algorithm, the cp is selected as the node that lies halfway through the longest EDP. This decision is based on the observation that the midpoint of the longest EDP often provides a balanced splitting point for further computations, particularly in dense network topologies, where it is critical to evenly distribute the workload between the subsequent EDPs. While the choice of cp at this midpoint is heuristic and determined ad hoc in this version of the algorithm, it serves to minimize the complexity of the sub-problems created during the search for the remaining two EDPs. Specifically, it ensures that the two remaining paths, which are crucial for constructing the structures in Sections III-A2 and III-A3, are reasonably balanced in length. This heuristic aims to prevent situations where one sub-problem becomes disproportionately more complex than the others.

After choosing cp, it is used to find the remaining two EDPs in order to use them as the foundation for the structures in Section III-A2 and III-A3. This is done by finding the longest

EDP between s and cp for step $\boxed{1}$. Similar for step $\boxed{1}$ the longest EDP between cp and d is used for step $\boxed{2}$.

2) Tree: The underlying structures of both steps $\boxed{1}$ and $\boxed{2}$ are trees, that are generated by expanding the longest EDP. In contrast to the structure of $\boxed{1}$, the tree in $\boxed{2}$ is expanded to a planar structure with faces, which is described in Chapter III-A3. The first tree build, expands the EDP in the direction of cp to s. The direction was chosen to address the issue of having too many faulty edges at the source. This leads to a multitude of leaves at the source, all of which serve as entry points into the routing structure. The second tree expands the EDP in the direction of cp to d. At the beginning of the tree building algorithm, the tree consists of the longest EDP as previously described. After that, the algorithm iterates over the nodes of the tree and adds their neighboring nodes, which are not part of the tree yet, along with the corresponding edges to the tree. These iterations will be terminated once no further edges can be inserted into the tree. Next comes the pruning of the tree, as it contains paths that are not leading to the destination. The pruning function will be called repeatedly until no more nodes can be removed from the tree. In every call of this pruning function, leaves are removed if they don't have a connection to d in the topology. At the last part of the algorithm, each node gets a rank, which describes the order in which the routing has to choose its next hop. For this part a ranking function assigns ranks to the nodes of the tree, starting from the leaves. A rank reveals the information about how many edges need to be traversed from the respective node to reach the destination. A special ranking is needed for the rankings of the EDP nodes, since they need to be traversed first.

3) Faces: Since the routing of step $\boxed{3}$ starts at s and ends at cp, the reversed tree structure build in step $\boxed{1}$ needs to be changed. This adjustment is needed due to the intricate nature of routing within a tree structure, particularly when navigating from leaf nodes towards the root, relying solely on localized information available at each node. In order to route on this tree starting from the leaves, face routing is chosen as the concept of routing to the checkpoint. Being able to use face routing and find the faces of a graph, implies that the graph is planar. This is given by the definition of a graph represented as a tree [16]. Conducive to route from s to cp, the source gets connected to the leaves of the structure and coordinates get added to each node of the structure, representing a planar topology. Following the coordinates of each node, the faces of the structure need to be determined. The algorithm determines the faces of a planar graph by traversing each node and examining the half-edges emanating from that node. New faces are identified by visiting neighboring nodes starting from a node until a closed path is formed.

B. Routing

The routing is divided into 2 sections (Sections III-B2 and III-B1), as shown in Fig. 1. The routing scheme changes as soon as the packet to be sent has arrived at cp. This is due to the change in the underlying structure, which was described in Chapter III-A. Before using the structure provided by the precomputation, packets are routed along the EDPs, which span from s to d. In case the routing via the EDPs fails, the packet then first passes through a planar structure with faces $\boxed{3}$ and then through a tree structure $\boxed{4}$.

1) Face Routing: Face routing describes the first part that the packet traverses on its complete route to the destination. It is first routed from s to cp. The idea of the face routing algorithm is similar to [12]. We first explain it in general before showing how it can be applied to our setting.

Since the structure created in Chapter III-A3 is planar and the faces have been determined, this paragraph explains how to route in any face in order to get to the next face in general or to find the destination directly. An example of the face routing algorithm is shown in Fig. 2. Three variables are given in this Figure, which include the Entry Point (EP), from where the routing starts in the current face, the Destination (D), and the Intersection Point (IP), which represents the intersection point between the imaginary edge (EP, D). In the first phase of routing, the demand follows the green arrows. At each edge, it is checked whether it intersects with the imaginary edge. If such an intersection exists, it is further checked if this intersection is the one closest to the destination and is stored. This process is repeated until an invalid edge is encountered, causing the routing to backtrack, or a loop within the face is completed. Next, the packet follows the blue path to the node of the previously determined nearest IP and transitions to the next face. At this point, this process is repeated until the packet encounters D in the first pass.

We note however that this face routing as described requires some header modification in general, as we move from face to face, and can no longer rely on the incoming port to uniquely identifiy our current routing state – opposed to edge disjoint paths. Notwithstanding, in our proposal, we do not need to change faces, as we start on the outer face, which contains both the source and the checkpoint. As long as source and checkpoint remain connected in the pre-built subgraph-structure post-failures, they will both be on the outer face, and we can hence route from source to the checkpoint without header modification, as we switch from face to tree routing once we hit the checkpoint.

2) Tree Routing: Tree routing describes the second part that the packet goes through on its complete path to the destination. In this part, it is ultimately routed from cp to d.

First, routing follows the previously edge-disjoint paths. This is ensured by prioritized ranking. The lower the rank of a node, the higher its priority. Assuming that all EDPs contain a faulty edge, the routing begins via the tree. This is divided into sections, which correspond to the type of port through which the packet arrives at any node (A) of Fig. 3.

When the packet arrives through the incoming edge of the parent node, it indicates that node A is being visited for the first time. Therefore, the optimal child (B) with the lowest rank is selected first, the packet is forwarded there, and the process is repeated.

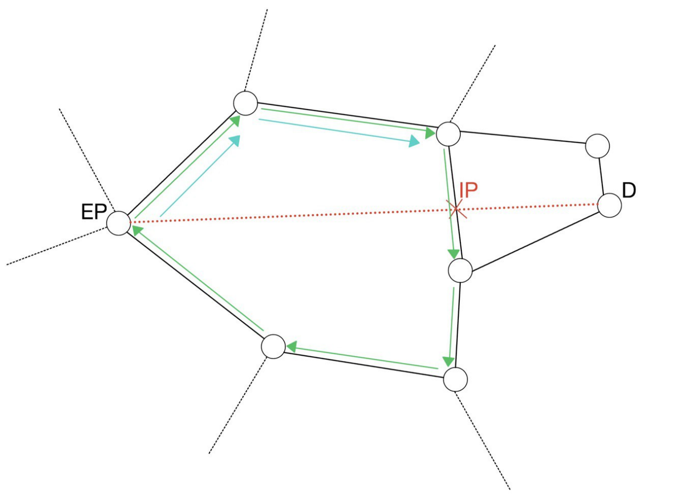

Fig. 2. Example of Face Routing used as a motivation for step $\boxed{3}$. The first path is searching for the intersection point (IP) and second path is traversing to the closest intersection from the entry point (EP) of the face. This figure serves solely as an illustrative example of the conceptual workings of face routing and employs different nodes than those depicted in Fig. 1.

If the packet returns through the child node B because all outgoing edges there might have failed, it means that another child must be selected. For this, all outgoing edges from A are sorted by their ranks, and it is checked which child has the next higher rank compared to B. In this case, it would be C. This process is similar if the packet is sent back from node C to node A.

In the last case, the packet arrives back at A from D. This means that the child with the highest rank also failed in routing, and there are no other children to which the packet can be forwarded. Therefore, the packet is forwarded from A to its parent node.

Here, the special case to be considered is when there are no further children at the source, the routing has failed.

3) Theoretical Guarantees: As noted in the precomputation in §III-A, we start with as many EDPs as possible, i.e., if the source and destination are r-edge-connected, we obtain r such paths, and we can hence obtain $(r-1)$-resilience by successively routing on the paths as shown in *CASA* [15].

In our proposal, we leave the first $(r-1)$ EDPs untouched and hence directly obtain $(r-2)$-resilience. For the last EDP, after the checkpoint, our proposal is guaranteed to reach the destination if routing on the last EDP also reaches the destination, as we simply potentially offer more options past the checkpoint, see the original *TREE* paper [8]. For the last EDP before the checkpoint, the original part of the EDP is contained in the planar "face" subgraph built in the preprocessing, which is edge-disjoint from the first $(r-1)$ EDPs and the tree structure past the checkpoint. If the r-th EDP still connects source and checkpoint, then the planar "face" subgraph also connects source and checkpoint, as the r-th EDP is contained in it, and as thus we can route from the source to the checkpoint analogously as in [9, §6.2].

Hence, overall, we retain the $(r-1)$-resilience of the original

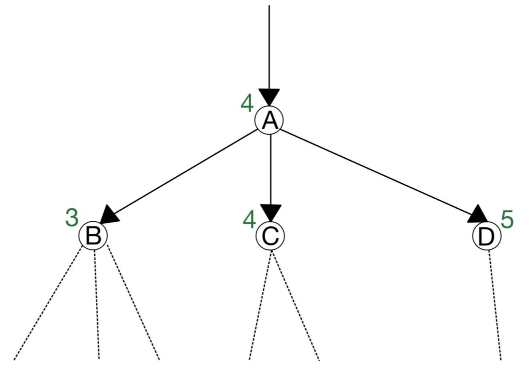

Fig. 3. Example of Tree Routing used in step $\boxed{4}$. Ranks of the nodes.

r EDPs, but possible induce longer paths/stretch due to the extension of the last EDP into a larger subgraph. Furthermore, the latter hopefully allows for more resilience to failures, which we next evaluate heuristically.

IV. Evaluation

The evaluation is divided into two sections. First, in Fig. 4, randomly generated graphs with randomized failures are evaluated. Subsequently, TopologyZoo [17] graphs with clustered failures are evaluated in Fig. 5. We use Python to perform these graph-based simulations. The metrics used in all figures are listed below:

$Hops$: the number of edges needed to reach a specific destination from a source. The average number of hops is calculated by summing the hops of all repetitions of a failure rate (or a graph in the case of Fig. 5) that did not fail, and then dividing by the number of repetitions of a failure rate that did not fail.

$Resilience$: The resilience R is defined as the probability of success with which a packet reaches its destination from a source. To quantify this, the resilience is computed as the ratio of successful transmissions m to the total number of transmission attempts n. Mathematically, the resilience of a path P is expressed as: $R = \frac{m}{n}$ where m is the number of successful transmissions from the source s to the destination d and n is the total number of transmission attempts. To determine the average resilience $\bar{R}$ across multiple repetitions or for a given failure rate, we compute the mean resilience across all tested paths or graph topologies. If R_i is the resilience of the i-th path, the average resilience is calculated as: $\bar{R} = \frac{1}{k}\sum_{i=1}^{k} R_i$ where k is the number of paths or repetitions considered, and R_i is the resilience of the i-th path. For each failure rate or graph configuration (as shown in Fig. 5), the sum of all path resiliences is divided by the total number of repetitions to yield the average resilience.

In the following Section IV-A, each step on the X-axis represents the average of the results from 100 experimental runs for that value. In Section IV-B, each step on the X-axis represents the average of the results from 5 experimental runs for that value. In each experimental run, the s from which the packet is routed to d changes.

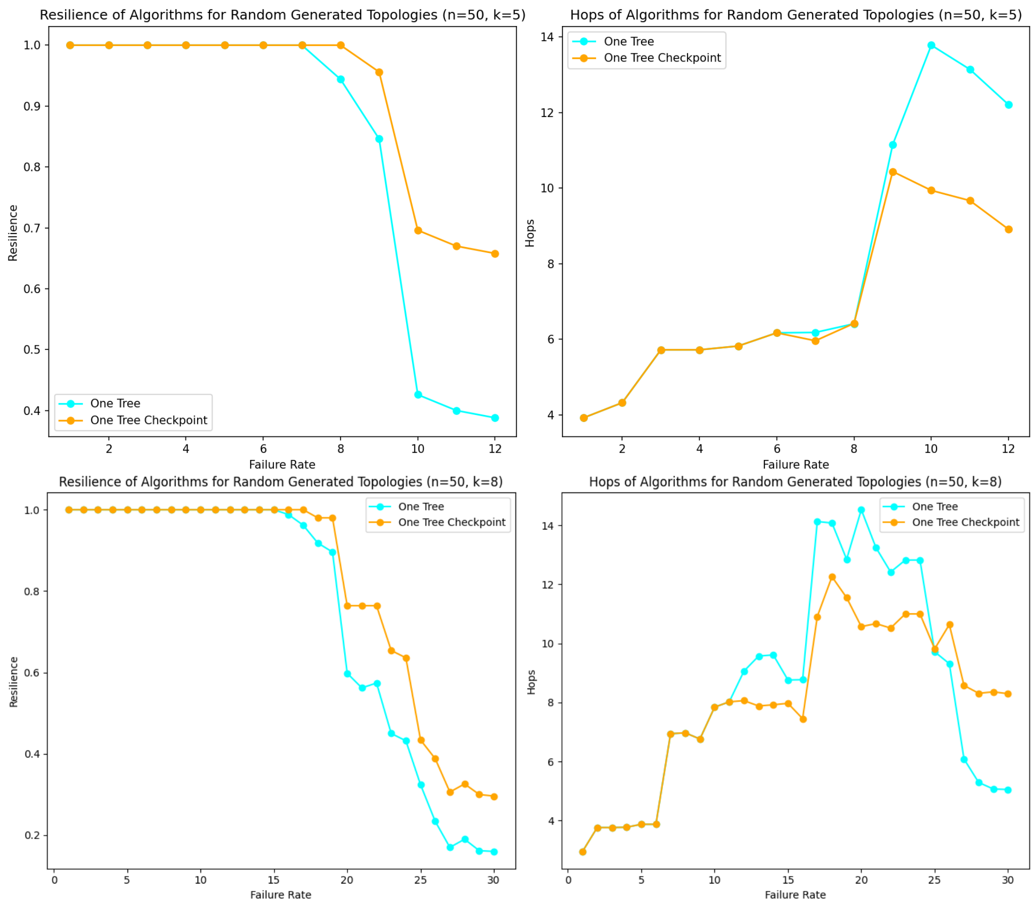

Fig. 4. Comparison of the resilience and hops of different algorithms on randomly generated 5-regular (upper) and 8-regular (lower) graphs. Each Failure Rate step represents an increment of 5 failures added to the network topology. The left plot displays the resilience (success rate) of each algorithm, while the right plot shows the average number of hops required.

For each step on the X-axis, five different source nodes s are randomly chosen to evaluate the performance of the algorithm when routing to the same destination d. The source nodes are selected as follows: prior to each run, the list of all nodes in the graph is shuffled using a pseudo-random number generator initialized with a fixed seed to ensure reproducibility. From the shuffled list, the first five nodes are selected as source nodes s. The destination node d remains the same across all runs for consistency in comparison. The random selection of source nodes enhances the diversity of test scenarios by varying the sources across runs, allowing the algorithm to be tested against different parts of the network. This approach improves robustness and ensures more generalizable results.

A. Randomly Generated Graphs

First, the results of the randomized graphs from Fig. 4 are evaluated. These graphs consist of 50 nodes, each with a connectivity of 5 or 8 respectively. In both plots, the respective metrics in relation to the failure rate can be seen. The first failure rate (FR) step involves selecting 5 edges uniformly at random from the total set of edges and marking them as faulty. A FR step indicates that 5 faulty edges are randomly selected, in the same manner as the first step, and added to the existing faulty edges. The randomly generated graphs of this chapter are created by using the $random\_regular\_graph(n, k, seed)$ algorithm of NetworkX [18], which generates a random k-regular graph on n nodes. For this experiment 100 5-regular and 100 8-regular graphs were generated. For each graph a destination and five different starting points are chosen at random.

Additionally, one other algorithm from a recent paper [8] is compared to the algorithm of this work. $OneTree$ is the original form, which operates without face routing and the checkpoint.

Regarding resilience, both algorithms offer maximum resilience up to FR=7 on 5-regular graphs (FR=15 on 8-regular), as can be seen in the left plot of Fig. 4. From FR=8 on 5-regular graphs (FR=16 on 8-regular), $OneTreeCheckpoint$ consistently achieves the highest resilience values.

Upon examining the hop counts in the right plot of Fig. 4, it becomes apparent that $OneTree$ achieves an identical number of hops up to FR=6 on 5-regular graphs (FR=11 on 8-regular graphs). Starting from FR=7 on 5-regular graphs, the hop counts of $OneTree$ exceed those of $OneTreeCheckpoint$. On 8-regular graphs between FR=11 and FR=24, the $OneTree$ Algorithm generally has a higher hop count than the $OneTreeCheckpoint$, from FR=25 to FR=30, the $OneTree$ needs a smaller number of hops.

In summary, this means that, with regard to randomized regular graphs, the resilience is generally improved by the hybrid approach of this work but also requires a comparatively higher number of hops in some cases.

B. Real World Graphs

The experiments with the $OneTree$ and $OneTreeCheckpoint$ algorithms on real-world graphs [17] are shown in Fig. 5. There, four graphs are listed with their number of nodes and measured hops/resilience. The used topologies consist of: $BtEurope$ with $n = 24$, $Geant2001$ with $n = 27$, $Bics$ with $n = 33$ and $Garr201105$ with $n = 59$. The chosen model for the clustered selection of faulty edges is taken from [19]. This targeted attacks failure model simulates adversarial failures, such as those caused by attacks, by focusing on well-connected regions or clusters in the network. These clusters could represent strategically important areas, like groups of cities. An adversary might target links within these clusters to disrupt the network's fast-recovery capabilities. In this model, links are selected for failure based on the clustering coefficient of nodes, with a predefined number of links either chosen randomly from a candidate set or fully disabled if the set contains fewer links than the failure parameter.

In the upper plot of Fig. 5, it can be seen that $OneTreeCheckpoint$ achieves higher resilience in two out of the four experiments. This is due to the fact that this algorithm provides multiple entry points from the source, even when influenced by several faulty edges. However, it can also be observed that $OneTree$ delivers increased resilience values in the experiments with graphs of $n = 24$ and $n = 59$. This can be attributed to the fact that, in selected cases, expanding a single EDP is more effective than creating a broader hybrid structure. Considering the hops in the lower plot of Fig. 5, both algorithms achieve similar numbers in the first three experiments. In the fourth experiment, a significant increase in hops is observed for $OneTreeCheckpoint$, which can be attributed to the structure of the hybrid architecture having a larger number of edges. However, in selected cases, this also leads to substantial overhead regarding the hop count.

It is also noteworthy that the precomputation time of the algorithms was measured in the experiments, although not graphically represented in this work. From these results, it can be concluded that $OneTreeCheckpoint$ usually requires three times as much precomputation time compared to the $OneTree$ algorithm.

In summary, it can be concluded that $OneTreeCheckpoint$ enhances resilience by approximately 10.64%, making it more adept at withstanding clustered failures. However, in some cases, the increased hops due to its larger structure may lead to unnecessary overhead, potentially compromising its resilience despite these improvements.

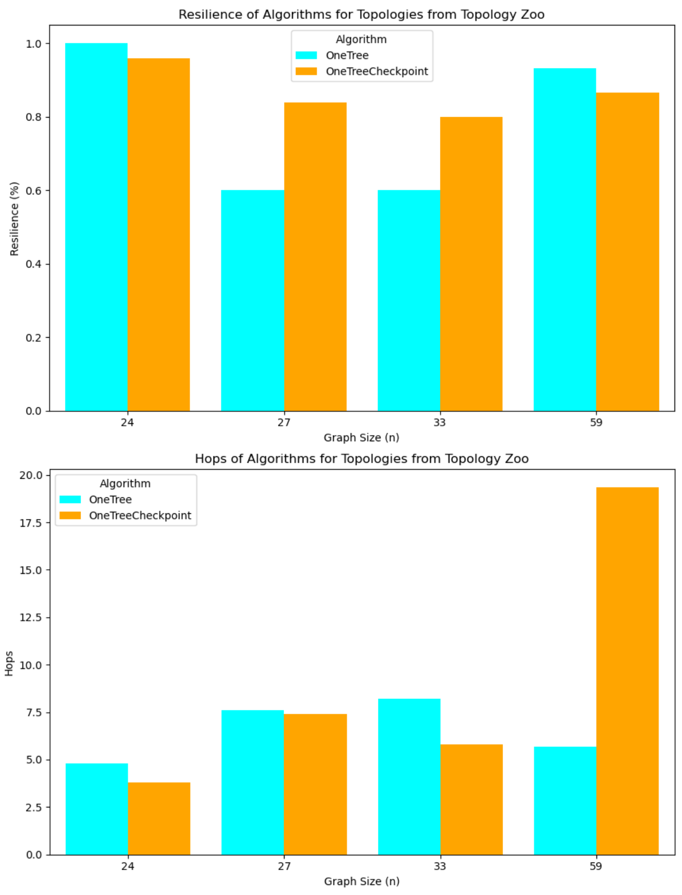

Fig. 5. Comparison of the resilience and hops of the two algorithms on various graph sizes using topologies from the Topology Zoo. The upper plot shows the resilience (success rate) of each algorithm, while the lower plot illustrates the average number of hops. Each bar represents the aggregated results from 5 experimental runs.

V. Conclusion

This work presents an initial approach to combine the advantages of face routing with those of tree routing. Initially, packets are forwarded from the source to a checkpoint cp via face routing. Next, the packets reach their destination with the help of tree routing. This method was chosen to counteract clustered failures at the source and to provide a variety of entry points into the routing for the packets. It has been observed that the hybrid approach can enhance resilience not only in the case of clustered failures but also in comparison to similar algorithms in scenarios involving random failures.

VI. Outlook

Subsequently, further possibilities are described for improving the combination of face routing and tree routing. One possibility is to modify the choice of cp by, for example, already globally knowing the ”fault tolerance” of a node and thus highlighting it in the choice of cp. This choice can also be randomized in other ways. Another approach could be to modify the tree formation. Modifications could be incorporated, such as building multiple trees or restricting the width of the tree. However, in this case, it would be necessary to evaluate whether the graph is large enough to fill multiple trees. Furthermore, in some scenarios, it would be beneficial to use multiple cp, through which routing should be performed. To this end, we also plan to run larger-scale evaluations on more data sets and topologies.

Lastly, it could also be interesting to extend the ideas presented in this paper to other concepts and methodologies in the fast failover setting, such as randomization [6], header modification [7], [20], dynamic failures due to link flapping [21], shortcutting failures to reduce the detour stretch overhead [22], [23] or even to directed graphs [24], [25].

References

[1] E. Katz-Bassett, H. V. Madhyastha, J. P. John, A. Krishnamurthy, D. Wetherall, and T. E. Anderson, "Studying black holes in the internet with hubble," in *5th USENIX Symposium on Networked Systems Design & Implementation, NSDI 2008, April 16-18, 2008, San Francisco, CA, USA, Proceedings*, J. Crowcroft and M. Dahlin, Eds. USENIX Association, 2008, pp. 247–262. [Online]. Available: http://www.usenix.org/events/nsdi08/tech/full_papers/katz-bassett/katz-bassett.pdf

[2] G. Aceto, A. Botta, P. Marchetta, V. Persico, and A. Pescapè, "A comprehensive survey on internet outages," *J. Netw. Comput. Appl.*, vol. 113, pp. 36–63, 2018. [Online]. Available: https://doi.org/10.1016/j.jnca.2018.03.026

[3] P. Gill, N. Jain, and N. Nagappan, "Understanding network failures in data centers: measurement, analysis, and implications," in *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011*, S. Keshav, J. Liebeherr, J. W. Byers, and J. C. Mogul, Eds. ACM, 2011, pp. 350–361. [Online]. Available: https://doi.org/10.1145/2018436.2018477

[4] M. Chiesa, A. Kamisinski, J. Rak, G. Rétvári, and S. Schmid, "A survey of fast-recovery mechanisms in packet-switched networks," *IEEE Commun. Surv. Tutorials*, vol. 23, no. 2, pp. 1253–1301, 2021. [Online]. Available: https://doi.org/10.1109/COMST.2021.3063980

[5] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via data plane mechanisms," in *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013, Lombard, IL, USA, April 2-5, 2013*, N. Feamster and J. C. Mogul, Eds. USENIX Association, 2013, pp. 113–126. [Online]. Available: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/liu_junda

[6] M. Chiesa, A. Gurtov, A. Madry, S. Mitrovic, I. Nikolaevskiy, M. Shapira, and S. Shenker, "On the Resiliency of Randomized Routing Against Multiple Edge Failures," in *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), I. Chatzigiannakis, M. Mitzenmacher, Y. Rabani, and D. Sangiorgi, Eds., vol. 55. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 134:1–134:15. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/6269

[7] M. Chiesa, I. Nikolaevskiy, S. Mitrovic, A. V. Gurtov, A. Madry, M. Schapira, and S. Shenker, "On the resiliency of static forwarding tables," *IEEE/ACM Trans. Netw.*, vol. 25, no. 2, pp. 1133–1146, 2017. [Online]. Available: https://doi.org/10.1109/TNET.2016.2619398

[8] O. Schweiger, K.-T. Foerster, and S. Schmid, "Improving the resilience of fast failover routing: Tree (tree routing to extend edge disjoint paths)," in *Proceedings of the Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '21. New York, NY, USA: Association for Computing Machinery, 2022, p. 1–7. [Online]. Available: https://doi.org/10.1145/3493425.3502747

[9] K. Foerster, J. Hirvonen, Y. Pignolet, S. Schmid, and G. Trédan, "On the feasibility of perfect resilience with local fast failover," in *2nd Symposium on Algorithmic Principles of Computer Systems, APOCS 2020, Virtual Conference, January 13, 2021*, M. Schapira, Ed. SIAM, 2021, pp. 55–69. [Online]. Available: https://doi.org/10.1137/1.9781611976489.5

[10] ——, "On the price of locality in static fast rerouting," in *52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2022, Baltimore, MD, USA, June 27-30, 2022*. IEEE, 2022, pp. 215–226. [Online]. Available: https://doi.org/10.1109/DSN53405.2022.00032

[11] B. Leong, S. Mitra, and B. Liskov, "Path vector face routing: Geographic routing with local face information," in *13th IEEE International Conference on Network Protocols (ICNP 2005), 6-9 November 2005, Boston, MA, USA*. IEEE Computer Society, 2005, pp. 147–158. [Online]. Available: https://doi.org/10.1109/ICNP.2005.32

[12] I. Stojmenovic, "Position-based routing in ad hoc networks," *IEEE Commun. Mag.*, vol. 40, no. 7, pp. 128–134, 2002. [Online]. Available: https://doi.org/10.1109/MCOM.2002.1018018

[13] F. Cadger, K. Curran, J. A. Santos, and S. Moffett, "A survey of geographical routing in wireless ad-hoc networks," *IEEE Commun. Surv. Tutorials*, vol. 15, no. 2, pp. 621–653, 2013. [Online]. Available: https://doi.org/10.1109/SURV.2012.062612.00109

[14] K.-T. Foerster, A. Kamisinski, Y.-A. Pignolet, S. Schmid, and G. Trédan, "fast-failover," https://gitlab.cs.univie.ac.at/ct-papers/fast-failover, 2022.

[15] K.-T. Foerster, Y.-A. Pignolet, S. Schmid, and G. Tredan, "Casa: Congestion and stretch aware static fast rerouting," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. IEEE Press, 2019, p. 469–477. [Online]. Available: https://doi.org/10.1109/INFOCOM.2019.8737438

[16] D. B. West, *Introduction to Graph Theory*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 2001.

[17] S. Knight, H. X. Nguyen, N. Falkner, R. A. Bowden, and M. Roughan, "The internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, 2011. [Online]. Available: https://doi.org/10.1109/JSAC.2011.111002

[18] A. Hagberg, P. Swart, and D. S Chult, "Exploring network structure, dynamics, and function using networkx," Los Alamos National Lab.(LANL), Los Alamos, NM (United States), Tech. Rep., 2008.

[19] K. Foerster, A. Kamisinski, Y. Pignolet, S. Schmid, and G. Trédan, "Grafting arborescences for extra resilience of fast rerouting schemes," in *40th IEEE Conference on Computer Communications, INFOCOM 2021, Vancouver, BC, Canada, May 10-13, 2021*. IEEE, 2021, pp. 1–10. [Online]. Available: https://doi.org/10.1109/INFOCOM42981.2021.9488782

[20] E. van den Akker and K. Foerster, "Short paper: Towards 2-resilient local failover in destination-based routing," in *Algorithmic Aspects of Cloud Computing -9th International Symposium, ALGOCLOUD 2024*. Springer, 2024.

[21] W. Dai, K. Foerster, and S. Schmid, "On the resilience of fast failover routing against dynamic link failures," *CoRR*, vol. abs/2410.02021, 2024. [Online]. Available: https://arxiv.org/abs/2410.02021

[22] A. Shukla and K. Foerster, "Shortcutting fast failover routes in the data plane," in *ANCS '21: Symposium on Architectures for Networking and Communications Systems, Layfette, IN, USA, December 13 - 16, 2021*. ACM, 2021, pp. 15–22. [Online]. Available: https://doi.org/10.1145/3493425.3502751

[23] S. Althoff, F. Maassen, M. Weiler, A. Shukla, and K. Foerster, "Towards local shortcutting of fast failover routes," in *Proceedings of the on CoNEXT Student Workshop 2023, CoNEXT-SW 2023, Paris, France, 8 December 2023*, G. Tyson and M. Varvello, Eds. ACM, 2023, pp. 1–2. [Online]. Available: https://doi.org/10.1145/3630202.3630223

[24] J. Grobe, S. Althoff, and K. Foerster, "Analyzing network routing resilience: A hybrid approach of face and tree routing," in *14th International Workshop on Resilient Networks Design and Modeling, RNDM 2024*. IEEE, 2024.

[25] E. van den Akker and K. Foerster, "Brief announcement: On the feasibility of local failover routing on directed graphs," in *Stabilization, Safety, and Security of Distributed Systems - 26th International Symposium, SSS 2024*, ser. Lecture Notes in Computer Science. Springer, 2024.