Increasing Network Resilience Through Dynamic Routing with Disjoint Paths

Konrad Altenhofen, Julian Zobel and Björn Scheuermann
Communication Networks Lab
Technical University of Darmstadt
Darmstadt, Germany
Email: {konrad.altenhofen, julian.zobel, scheuermann}@kom.tu-darmstadt.de

***Abstract*—Network resilience and routing adaptivity are essential for communication in state-of-the-art wired IP networks. While dynamic routing approaches consider the network topology when making routing decisions, they cannot guarantee that a functional path is provided during failures. However, by combining the well-known mechanism of dynamic routing with the possibility to use disjoint paths, uninterrupted communication can be guaranteed during single failures, while retaining the adaptivity to the current topology of dynamic routing. For that, we present On-Demand Disjoint Dynamic Routing (OD³R), a novel mechanism for dynamic routing with disjoint paths. Our simulation results demonstrate that OD³R guarantees at least one working path during any single link failures and an increased reliability in multi-link failure scenarios compared to single shortest path routing with reasonable overhead.**

***Index Terms*—disjoint paths, dynamic routing, network layer, network resilience, redundancy, OD3R, wired IP**

I. Introduction

Networked communication for operating critical infrastructures, such as railway, water or power systems, and Information and Communication Technology (ICT), often has high constraints, especially for timeliness and reliability. To this day, specialized systems and networks are employed to fulfill individual requirements and constraints specifically tailored for individual use cases. However, aspects like scalability, maintainability, interoperability, and cost instigated the increasing replacement of such specialized systems by (proprietary) IP-based network solutions. The modernization and digitalization of the German railway control infrastructure[1] is one example, that – on its downside – also highlights the increasing impact of outages and failures that come with simultaneously increasing network size and demand[2]. Therefore, network resilience is an essential requirement for future-proof IP networks.

Today, dynamic routing protocols like the *Routing Information Protocol* (RIP) [1], the *Enhanced Interior Gateway Routing Protocol* (EIGRP) [2], *Open Shortest Path First* (OSPF) [3], or *Intermediate System to Intermediate System* (IS-IS) [4] are widely used in interior gateway networks. These protocols exchange topology information between internal routers, allowing them to make routing decisions based on the current topology and adapt their routing decisions if the topology changes. Specifically, this adaptivity increases the resilience of a network by considering changes in the topology, e.g., additions or failures of nodes and links. Although allowing a disrupted network to restore itself to a functional state, the communication between network nodes might be disrupted for a longer time – that is until the failure is discovered, distributed through the network, and routing decisions are adapted. An alternative is the use of routes that are disjointed from each other. These routes may be used as backup paths [5], [6] or in a source-selectable manner [7], [8]. Since disjoint routes do not share any common link or node, depending on the diversity criteria, at least one working path is guaranteed if only a single network component fails. However, if multiple links fail at the same time, communication is not necessarily protected despite possible additional redundancy capacities of the network. Moreover, many mechanisms for routing on disjoint paths are slow to incorporate topology changes or cannot adapt to changes at all, as they rely on pre-computation or require a lot of coordination.

Both the usage of disjoint paths and dynamic routing provide viable solutions to mitigate failures in the network, but each option also has its individual problems. To mitigate some of the problems each approach brings and to increase the resilience of a network in the face of link failures, we propose to combine both options. For this, we determine and discuss the challenges of a combined approach for dynamic routing with disjoint paths and investigate possible solutions. Based on our findings, we developed *On-Demand Disjoint Dynamic Routing* (OD³R), a mechanism specifically designed for dynamic routing with disjoint paths in wired IP networks. This combination guarantees at least one functioning path in the event of individual link failures through the use of disjoint paths, while preserving adaptivity to the current topology and increasing resilience in the face of multiple simultaneous link failures. OD³R is evaluated through simulation and theoretical analysis of different characteristics, such as overhead, quality of paths, and routing stability.

Specifically, we make the following contributions:

- We discuss the key challenges and requirements for dynamic routing with disjoint paths.
- We present OD³R (*On-Demand Disjoint Dynamic Routing*), a novel mechanism enabling routing on disjoint paths that dynamically adapt to the network topology distributed by any link-state protocol, e.g., IS-IS or OSPF.

[1] https://digitale-schiene-deutschland.de/en/digital-rail
[2] https://www.dw.com/en/a-63377385

ISBN 978-3-903176-63-8 © 2024 IFIP

- We evaluate the benefits, potential penalties, and overhead of OD³R with simulations in OMNeT++[3] using real-world topologies from the Internet Topology Zoo [9].

This paper is structured as follows. Section II presents the scope and assumptions of the paper, followed by a discussion of related work in Section III. Section IV describes the challenges of dynamic routing with disjoint paths. Section V presents *On-Demand Disjoint Dynamic Routing* (OD³R), followed by its evaluation through simulation and a theoretical analysis in Section VI. Section VII concludes this paper.

II. Scope and Assumptions

The focus of this work and the proposed mechanisms are wired IP networks, consisting of End Devices (EDs) and routers. Each ED is a leaf node in the network and can generate traffic, while being connected to one router. Each router itself is part of a larger interior network of connected routers. Within the scope of this paper, we specifically differentiate between EDs and the first router. In a practical deployment, of course, both could be implemented in the same system. We assume that the cut degree of the interior network between each pair of routers is at least two; hence, no single link failure can cut off parts of the topology. Furthermore, the network provides sufficient capacity for all traffic demands. Additionally, all communication is assumed to take place within a single administrative domain, i.e., multi-domain networks or networks using transit domains are not considered. Finally, we assume full control over the interior network, including programmability of routers, i.e., we can change their routing and forwarding behavior without limitations, and we assume the absence of malicious nodes in our network.

Disjointness is explicitly regarded w.r.t. the interior network, starting from the first interior router up to the router responsible for the destination. An ED can decide how the provided paths are used for each packet, using either a specific path or redundantly multiple paths. We do not consider a specific transport protocol.

III. Related Work

Approaches for disjoint routing can be grouped into two major classes: end-to-end disjoint paths and disjoint trees. Trees are commonly used for many different routing applications. Their usage simplifies path indication as well as route selection and reduces storage overhead compared to a naive end-to-end path-based solution. To employ the concept of tree-based routing for disjoint path routing, multiple disjoint trees need to be computed. However, minimizing the length of disjoint trees is *NP-complete* [10] and, thus, heuristic approaches are often chosen. Furthermore, even if the trees are optimal, they typically have higher costs than end-to-end paths.

Disjoint trees have proven their usability for network recovery after link failures [5]. Various approaches to finding disjoint trees exist, e.g., addressing demands for Quality-of-Service [11]–[13] or relying only on local information [14], [15]. Other approaches specifically find disjoint trees rooted at each node in a network [6] and maximally disjoint trees to support non-two-connected networks [16].

In contrast to disjoint trees, segment-routed robustly disjoint paths, i.e., paths that remain disjoint from each other even after a single link failure, can be found [8]. Comparable concepts are also available using mixed integer linear programming [17]. However, these approaches have a significant computational overhead and require extensive pre-computations.

The low path stretch of end-to-end paths can also be combined with the reduced number of routing table entries for disjoint trees. Babarczi et al. [7] propose a tag-based path indication and selection solution. Their mechanism creates tags uniquely identifying the disjoint paths to a destination based on previously computed end-to-end disjoint paths between each node and a destination. This does not require rewriting packet headers during routing and reduces both the number of required routing table entries and the required tag size. However, the computation and distribution of the tags and corresponding routing table entries require extensive computations and considerable coordination effort.

Within the vast realm of related work, individual approaches focus on specific aspects involved in routing with disjoint paths, e.g., path computation, path optimization, or routing table entry reduction. While some approaches mention dynamic or changing topologies as a possible use case, there are neither specifically dynamic routing approaches nor approaches with a concrete explanation of route handling and maintenance in link failure scenarios. Especially disjoint path routing approaches for wired networks only consider singular aspects, such as optimizing path identification, and neglect dynamic topology adaptations. In contrast, our presented approach considers all required aspects to enable dynamic routing with the usage of disjoint paths in wired IP networks, increasing network resilience by facilitating quick reactions to topology changes.

IV. Challenges

A mechanism that combines dynamic routing with the utilization of disjoint paths retains the challenges of each individual approach and, thus, must consider each of the following four main challenges:

- *Route Computation* — How to find disjoint paths?
- *Routing Protocol* — Which information needs to be exchanged and how?
- *Route Pinning and Forwarding* — How to indicate chosen paths to other routers and how to forward packets?
- *Route Maintenance* — How should a mechanism maintain routes and how to react to failures?

In addition, further complexity is introduced through the potential interdependency of challenges, as decisions for individual challenges can impact the design options for others. E.g., different route computation approaches require different amounts and types of information, which in turn have a strong impact on the requirements of the routing protocol. In the following, we detail each of these challenges. Their applied solutions are discussed in the subsequent section.

[3] https://github.com/konivik/inet-od3r

A. Route Computation

To allow routing with disjoint paths, a route computation algorithm is required to determine such disjoint routes. There are a number of specific algorithms for finding disjoint paths, e.g., [18]–[20]. They can be grouped according to (i) their type of output paths, namely, end-to-end paths or disjoint trees, and (ii) their mode of operation, i.e., distributed or centralized. Furthermore, the algorithms differ in their optimization goal, the quality of paths, and the required information.

B. Routing Protocol

To enable dynamic routing with disjoint paths, we require different types of information for different components to be available: (i) The computation of routes requires topology information, (ii) to facilitate forwarding, information on these routes needs to be distributed in the network, and (iii) route maintenance may need to exchange additional control information. The choice of the routing protocol combines all these issues. In interior networks, typically either link-state (cf. [3], [4]) or distance-vector (cf. [1], [2]) approaches are used to exchange topology information. This exchange might happen between multiple routers, or routers and a central instance like an SDN controller. Especially when routers do not compute all possible paths by themselves, the distribution of the calculated paths is required. Distribution approaches differ in (i) their operational mode, i.e., proactive and reactive, (ii) the initiator of the exchange, and (iii) an implicit or explicit exchange. This part of the routing protocol needs to be considered together with the route pinning and forwarding mechanism due to their close relation.

C. Route Pinning and Forwarding

To enable routing on a specific path, packets must hold a path indicator to declare the desired path. Thus, generating nodes must insert this indicator and intermediate nodes must be able to correctly interpret it. Approaches for Route Pinning and Forwarding differ in the associated overhead. This includes additional routing table overhead or increased packet sizes, as well as overhead for computation of identifiers, coordination of routers, and packet processing. The simplest approach to indicate the path of a packet is to send the complete path as part of the packet, which makes route indication more independent of the routing protocol but also increases the packet size the most. Smaller impact can be achieved by adapting route pinning to the specific routing approach. When using tree-based hop-by-hop routing, the destination address, together with the number of the tree, clearly identifies the path. For end-to-end paths, the combination of source and destination address together with the number of the path can be used instead. Other alternatives encompass individual tags or labels to identify a desired path, possibly reducing the packet size compared to full path indication, but increasing the computation and coordination overhead. They can be computed for the complete or only for the remaining path; the latter requiring updates in each forwarding step.

D. Route Maintenance

Since one of our main goals is to use dynamic routing to allow for the adaptation to topology changes, special emphasis is required to address route maintenance and the handling of failure states, which is typically an unknown path. Route maintenance mechanisms can be grouped into soft-state and hard-state approaches, i.e., deleting old routes explicitly via deletion request or implicitly by enforcing expiration of unused routes, respectively. In case of handling packets with an unknown path, these packets can either be dropped, sent via an alternate path, or buffered until the indicated path can be resolved. Naturally, the used approach must consider requirements and limitations for traffic in the network as well as computational costs, overhead, and storage on each node. Furthermore, methods for active path recovery, i.e., requesting path information from other nodes, or passive path recovery, e.g., timer-based or change-based redistribution of path information, can be used to recover to a stable state in which packets can be forwarded successfully.

V. On-Demand Disjoint Dynamic Routing (OD³R)

We highlighted the possible benefits of a dynamic routing protocol using disjoint paths, effectively combining both ideas to increase network resilience. Subsequently, we discussed the individual challenges resulting from this combination that need to be addressed by such a mechanism. Based on this, we present *On-Demand Disjoint Dynamic Routing* (OD³R), a mechanism specifically designed for dynamic routing with disjoint paths in wired IP networks. With OD³R, at least one functioning path can be provided in the event of individual link failures through the use of disjoint paths, while preserving adaptivity to the current topology and increasing resilience in the face of multiple simultaneous link failures. In this section, we detail each of the discussed individual challenges and provide further information on the logic behind OD³R. Note that OD³R is optimized for IPv4 in its current state, but can be adapted to IPv6 with minor changes. This, however, is not within the scope of this paper.

A. Route Computation Algorithm and Topology Exchange

OD³R uses a centralized algorithm for finding disjoint paths. It minimizes the sum in each set of end-to-end disjoint paths between a source and a destination. While tree-based paths have a reduced storage and path identification overhead, optimizing end-to-end paths is significantly less computationally expensive, and optimized end-to-end paths typically have a lower cost than tree-based disjoint paths. The usage of a centralized algorithm also increases stability, due to a reduced risk of meta-stability and guaranteed loop-free paths. It furthermore requires no additional communication or coordination effort compared to a distributed approach. Minimizing the sum of the cost of disjoint paths within each set provides a compromise between the different optimization goals, as both paths in the set are considered in the optimization while the computational complexity remains appropriate. Specifically, OD³R uses the Suurballe and Tarjan algorithm [18] for the computation of

disjoint paths. The computation is done on-demand by the first interior router after receiving a packet requesting disjoint path usage. This requires to have a graph of the complete network topology for the computation. Hence, the full topology is exchanged to the point of computation, i.e., each router, using a link-state protocol. OD³R is not limited to a specific link-state protocol. Every link-state protocol that exchanges the full topology and has fast convergence properties could be used, including well-known protocols like OSPF [3] or IS-IS [4].

B. Route Distribution, Route Pinning, and Forwarding

As path information after the computation is only present in the node computing the paths, i.e., the first interior router, the computed path needs to be indicated to intermediate nodes. Either the complete path needs to be included in each packet or the route needs to be distributed to intermediate routers on the path and to be set up. OD³R uses a hybrid approach. The complete source route is added to a packet to allow its routing along the intended path and to distribute the computed path for following packets. Based on the complete source route, intermediate nodes use the information of the remaining path to add a routing table entry including the residual in- and output path identifiers. The input path identifier is the hash of the residual path including the current intermediate node. It is used to select the correct routing table entry for incoming packets. OD³R uses the Jenkins hash function [21] for hash computation, allowing fast and low-cost computation while being fairly collision-resistant. The hash of the remaining path after this router is stored as *output path identifier*. This reflects the routing progress with its incremental route shortening. Subsequent packets, sent after path distribution, do not require the full path to be included. Instead, they are routed using only the residual path identifier, which is updated by intermediate nodes along the path (cf. [22], [23]). For that, intermediate routers replace the ID in the packet header with the output path ID of the matching routing table entry.

This procedure allows on-demand disjoint route computation and route exchange while reducing the communication overhead compared to a full path inclusion in each packet. Moreover, it reduces the routing table overhead at intermediate nodes compared to path identification based on a combination of the source and destination address.

C. Maintenance and Error Handling

OD³R uses a soft-state approach for route entries, i.e., routes which remain unused for a certain time expire. This allows the deletion of unused route entries without the need for explicit deletion requests. Routes are refreshed from time to time by including the full source route in the packet to ensure that routes are not wrongfully deleted and allow nodes to recover after a crash. Similarly, routes are refreshed after a topology change is detected by the link-state protocol.

However, it is still possible that a node receives a packet with an input path ID with no matching routing table entry. In this case, the path identifier option is removed from the packet and the packet is forwarded to the destination using the shortest path. No active path recovery is initiated by a node receiving a packet without a matching routing table entry for this path ID. Paths are passively restored due to path refreshment. Similarly, if no valid paths can be found during route computation, the request is omitted and the shortest path is used instead as the graph is not two-connected or the paths would be too long.

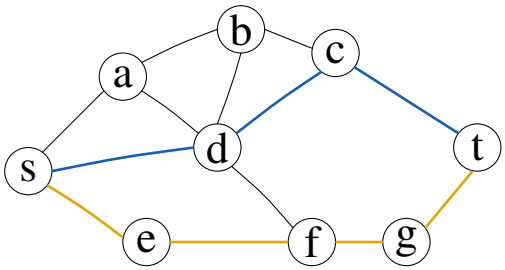


Fig. 1: Example topology of interior routers. Two possible disjoint paths between routers s and t are marked in blue and orange, respectively.

D. Packet Indicators

OD³R does not require additional packets besides normal traffic and those required by the link-state protocol. Instead, it includes computed paths and pinned routes in the options field of the network header of normal traffic packets, which provides compatibility and flexibility for adding and removing extra information. OD³R requires three option types to be included in the network header: (i) Disjoint Path Request, (ii) Source Route Exchange, and (iii) Residual Path ID.

The *Disjoint Path Request* option allows an End Device (ED) to indicate the desire for disjoint paths to the first interior router. Further, the ED can specify which path should be used through a path selector.

Distribution of computed disjoint paths is done via the *Source Route Exchange* option. The complete source route is added to the packet, both for immediate routing and path information exchange. This option is similar to the normal source route option but instructs intermediate nodes to add or update the corresponding routing table entry.

The *Residual Path ID* option is used to route packets along set-up disjoint paths. The included residual path identifier, together with the destination address, allows an intermediate node to determine the desired path to the destination. While the packet is forwarded, each node updates the residual path identifier.

E. OD³R Routing

To allow a better understanding, we illustrate the behavior of OD³R using an example: Figure 1 shows a mock topology consisting of interior routers, connected by links with unit costs. The topology is already known by each router and the default shortest path routes are computed using a link-state protocol. Initially, no OD³R routes are computed or installed.

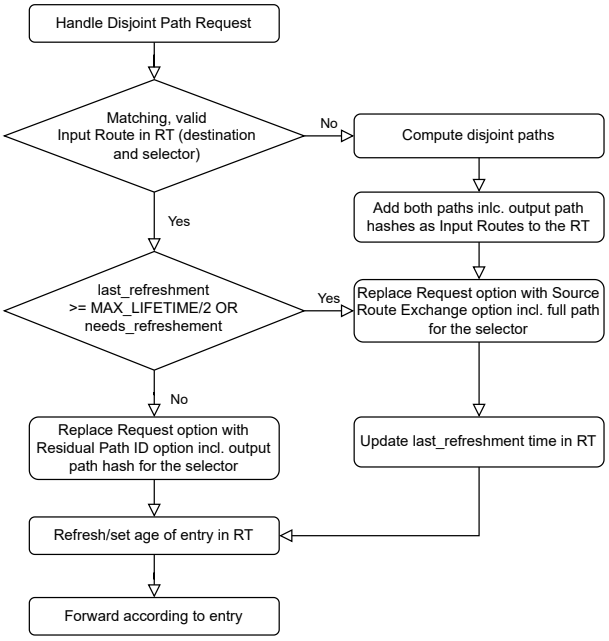


Fig. 2: Handling incoming packets with disjoint path requests.

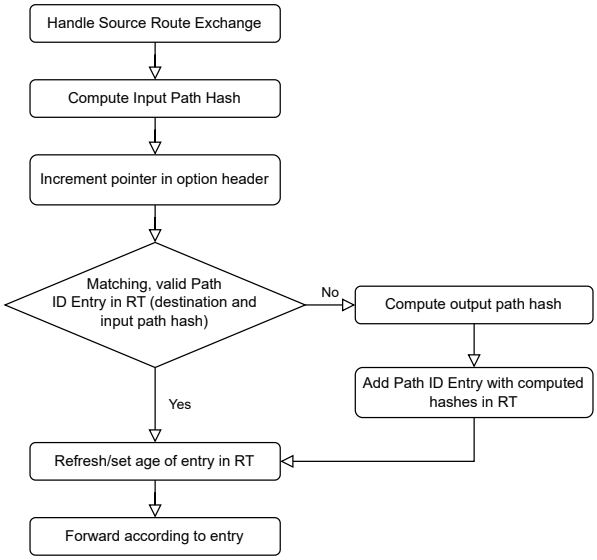


Fig. 3: Handling incoming packets for route exchange.

1) Handling a new Disjoint Path Request: Assume router $\mathbf{s}$ receives a disjoint path request with the path selector 0 from one of its connected EDs with the destination ED being connected to router $\mathbf{t}$. As detailed in Figure 2, router $\mathbf{s}$ handles the disjoint path request performing a routing table lookup using the destination $\mathbf{t}$ together with the path selector 0 as keys. As no routing table entry is present for this request, router $\mathbf{s}$ starts the route computation procedure [18] using the topology information extracted from the link-state database. This results in the two disjoint paths $0 : (\mathbf{s} \rightarrow \mathbf{d} \rightarrow \mathbf{c} \rightarrow \mathbf{t})$ and $1 : (\mathbf{s} \rightarrow \mathbf{e} \rightarrow \mathbf{f} \rightarrow \mathbf{g} \rightarrow \mathbf{t})$. Both paths are added to the routing table. Specifically, the entry includes (i) the entire computed path, (ii) its expiration time, (iii) the last refreshment time of the path, as well as (iv) a hash of the residual path (*output path hash*). Next, the disjoint path request is removed from the packet header and replaced with the complete path as a source route for path 0 as specified by the indicator inside the disjoint path request. The router sets the last refreshment time for path 0 and the age for both new paths in the respective routing table entries. Finally, router $\mathbf{s}$ forwards the packet to router $\mathbf{d}$.

2) Source Route Exchange: Router $\mathbf{d}$ receives the packet for destination $\mathbf{t}$ with the source route exchange option from router $\mathbf{s}$ including the path $(\rightarrow \mathbf{c} \rightarrow \mathbf{t})$. Figure 3 details how the packet is handled by router $\mathbf{d}$. After receiving the packet, router $\mathbf{d}$ computes the input hash $hash(c - t)$ and performs a routing table lookup using the destination address $\mathbf{t}$ and the computed hash. As no route is found, router $\mathbf{d}$ computes the output hash $hash(t)$ and adds a new routing table entry to the routing table. Next, the age of the entry is set and the packet is forwarded with an incremented source route pointer to router $\mathbf{c}$. Similar to router $\mathbf{d}$, router $\mathbf{c}$ computes the input hash, performs a routing table lookup, and adds a new route before forwarding the packet to its final destination router.

3) Following Disjoint Path Requests: When router $\mathbf{s}$ receives a disjoint path request for destination $\mathbf{t}$ with path selector 1, the distribution and packet handling performs similar as before, because this route is not set up. Similarly, if router $\mathbf{s}$ receives another disjoint path request for destination $\mathbf{t}$ with path selector 0 and the time since the last full path exchange exceeds half the route lifetime, the same procedure is executed. Only if router $\mathbf{s}$ receives another disjoint path request for destination $\mathbf{t}$ with path selector 0 before path refreshment is needed, the behavior of OD³R slightly differs: Router $\mathbf{s}$ replaces the disjoint path request with the residual path ID option, which includes the output path ID stored in the routing table entry $(hash(c - t))$. Then, the packet is forwarded to router $\mathbf{d}$, where the router performs a routing table lookup using the destination address together with the path ID as the key, as detailed in Figure 4. Since a valid routing table entry is available, router $\mathbf{d}$ replaces the ID in the path ID option with the output path ID from the routing table entry, updates the age in the entry, and forwards the packet to router $\mathbf{c}$. The packet is handled the same way at router $\mathbf{c}$ and forwarded to the destination router $\mathbf{t}$.

VI. Evaluation

To evaluate OD³R regarding its routing reliability as well as the resulting costs – i.e., additional routing table entries, communication and computation overhead – we analyze OD³R theoretically and evaluate it using simulations. OD³R is implemented using the INET framework[4] for OMNeT++[5]. The simulations were conducted on a system with an Intel Xeon Gold 6130 processor and 256 GB of RAM. All experiments were repeated five times with different random seeds.

[4]https://inet.omnetpp.org/ (version 4.4.1)
[5]https://omnetpp.org/ (version 6.0.1)

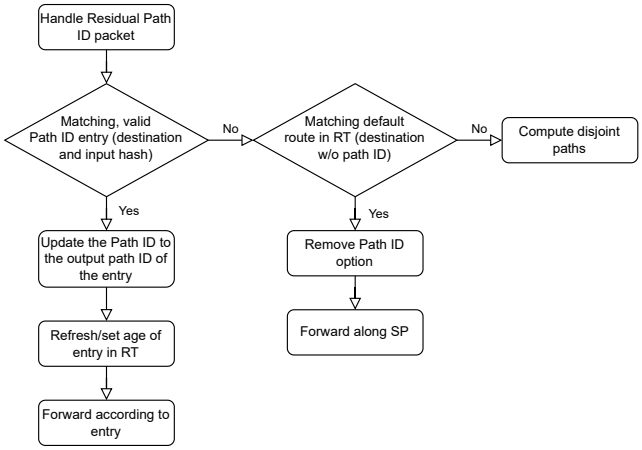


Fig. 4: Handling incoming packets with a path ID.

We utilize different topologies with varying size and connectivity to evaluate OD³R under different topological conditions, summarized in Table I. These include (i) generalized topologies like grids and fully connected graphs, (ii) large randomly generated topologies, and (iii) real-world topologies taken from the Internet Topology Zoo [9]. Links have unit costs and a link capacity that is much larger than the traffic demand. All nodes represent interior routers. Traffic is generated by a Constant Data Function (CDF) in each node, representing traffic requests originating from connected End Devices (EDs). Depending on the experiment, the packet destination is either chosen once for all generated packets per router or individually for each generated packet at each router. For the evaluation, we assume that the ED uses the provided paths redundantly, i.e., whenever traffic is routed using disjoint paths, packets are duplicated to include a path request for each path.

Instead of a specific link-state protocol, which would influence the results of our evaluation of OD³R, we use a generic global routing implementation to minimize its footprint. The implementation allows to specify a fixed delay until a change in the network is detect and propagated to all nodes. We examined OD³R in different link failure scenarios, including a maximum of total failures, a maximum of total concurrent failures, and different delays until the failure was repaired.

A. Route Reliability

The main objective of OD³R is to increase route reliability, that is, to guarantee a working path or at least increase the probability of a working path depending on the failure scenario. To measure the route reliability of OD³R, we investigate the share of missing packets compared to the unique packets that are received at least once. This allows to determine if at least one of the provided paths is working.

The simulation results show that OD³R generally increases the route reliability, as the rate of missing packets decreases compared to single shortest path routing (cf. Figure 5). In all experiments with a single link failure, we did not observe any packet loss when the packets are transmitted redundantly using OD³R, indicating that at least one of the provided paths was

TABLE I: Examined topologies and their properties. We use generalized topologies, real-world topologies from the Internet Topology Zoo [9], and randomly generated topologies.

	Topology	Nds	Edges	Min Cut	Avg SSP	Max SSP
Generalized	Grid 4x4	16	24	2	2.667	6
	Grid 5x5	25	40	2	3.333	8
	Grid 6x6	36	60	2	4.0	10
	Fullgraph 16	16	120	15	1.0	1
	Fullgraph 25	25	300	24	1.0	1
	Fullgraph 64	64	2016	63	1.0	1
ITZ [9]	Abilene	11	14	2	2.418	5
	BT Europe	17	30	2	1.963	3
	BT N.A.	25	35	2	2.696	6
	DFN	51	80	2	3.191	6
Random	N100E200	100	200	3	3.538	6
	N100E350	100	350	6	2.568	4
	N200E375	200	375	2	4.250	7
	N200E580	200	580	5	3.084	5

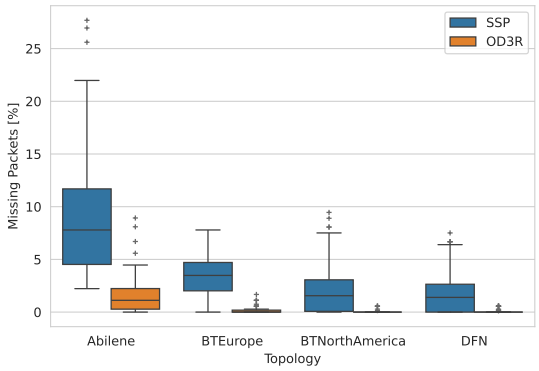
functional. In contrast, packet loss is observed with the same simulation configuration using Single Shortest Path (SSP) routing instead. Although expected, this behavior demonstrates the benefit of using disjoint paths and the indented behavior of OD³R.

In multi-failure scenarios, on the other hand, OD³R still shows an improved route reliability but packets are lost in less-connected topologies. For better-connected random and fullgraph topologies, however, a working path is provided and all unique packets are received at least once. Figure 5 shows the percentage of missing packets in the ITZ topologies for multi-failure scenarios, comparing OD³R with SSP routing. The rate of missing packets is significantly decreased through the use OD³R, as the redundant usage of disjoint paths increases the chance of at least one functional path. But in contrast to the single failure scenarios, OD³R experiences some packet loss, as the topology change detection and propagation is delayed by $40\,\mathrm{s}$, which might cause that an invalid path is chosen or a new path is not considered immediately.

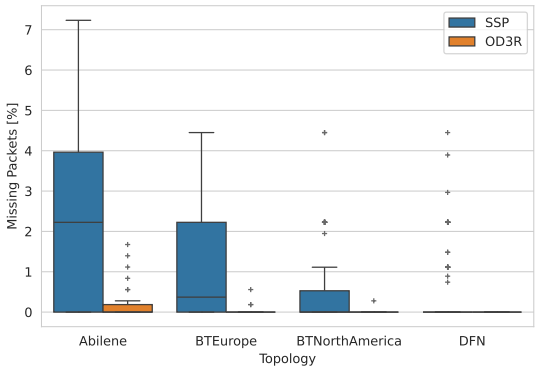
In Figure 5a and 5b, the number of concurrent failures is limited to one and the number of total failures is limited by the time to repair of $15\,\mathrm{s}$ and $240\,\mathrm{s}$, respectively. We see that the packet reception rate heavily depends on the size and connectivity of a topology: for example, more missing packets can be observed in the small Abilene topology than in the larger DFN topology, since each failure has a larger impact in smaller topologies for the same number of total failures.

In Figure 5c and 5d, multiple concurrent failures are possible. In contrast to the single concurrent failure scenario, the total number of failures increases with a larger time to repair and results in an increased packet loss. While OD³R still decreases packet loss compared to SSP routing noticeably, this gain reduces for smaller topologies like Abilene (cf. Fig. 5d).

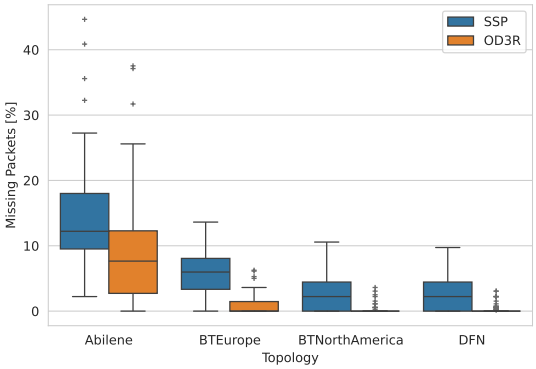
Throughout all simulations, OD³R improves the packet delivery ratio compared to SSP and provides a stable path under any single link-failure conditions. Even under more adverse multiple concurrent link failure conditions, OD³R is



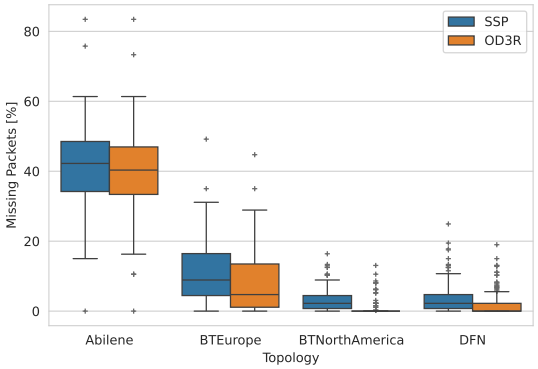
(a) 15 s time to repair. This single concurrent failure scenario has the most total link failures.



(b) 240 s time to repair. This single concurrent failure scenario has the least total link failures.



(c) 60 s time to repair. This multiple concurrent failure scenario has the least total link failures.



(d) 240 s time to repair. This multiple concurrent failure scenario has the most total link failures.

Fig. 5: Missing packets for OD³R compared to single shortest path (SSP) routing in the ITZ topologies. Multi-failure scenario with and without limitation on concurrent failures, respectively, with different time to repair.

still able to decrease packet loss, thus, increasing the network resilience compared to SSP. If paths are link-disjoint from each other, they do not share any common links. Any failed link — viewed on its own — can impact at most one of the paths. As such, one path is guaranteed to remain functional if only one link fails. For multiple failures, the probability of a functional path is still increased, since the probability a failure disrupts both paths is decreased. Furthermore, OD3R dynamically adapts its routing decisions based on the current topology. This means that as long as the topology allows it, another path is chosen after a link failure causes one path to break. Through this method, another failure can be absorbed.

B. Routing Table Size

OD³R uses the routing table to store disjoint routes computed on a node together with routes that are exchanged by other nodes. Thus, extra routing table entries need to be stored in addition to the entries needed for the shortest path routing. In the worst case, OD³R requires similar routing table entries than a source-destination-based approach. On average, these are $2 \cdot L \cdot dests$ routing table entries per node, where L is the average path length and $dests$ is the average number of destinations. However, OD³R uses the residual path for path identification. As paths from multiple sources are likely to converge at some point, we expect the number of routing table entries to be much lower.

The simulation results endorse our expectations for OD³R. Clearly, the number of routing table entries depends on the number of destinations and the average path length. As illustrated in Figure 6, showing the real-world and random topologies without any failure and randomly chosen destinations, routing table sizes generally increase for larger topologies. In general, OD³R adds 3 to 4 times the number of destinations as additional routing table entries compared to SSP. The maximum table size throughout all simulation runs was 947 entries for a single node in the *RandomN200E375* network. This is the largest topology, resulting in an average of 680 routing table entries per node. Nevertheless, this is much smaller than the worst-case expectation of $2 \cdot L \cdot dests$ entries.

C. Computation Overhead

OD³R introduces additional computation overhead, since additional logic is required for the computation of the disjoint paths, computation of the path hashes, packet handling, and

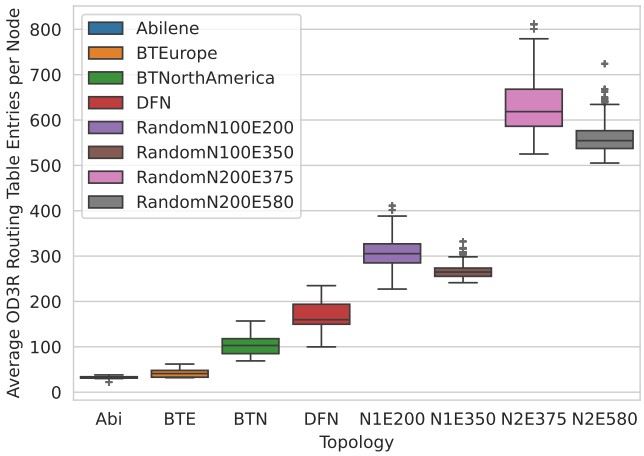


Fig. 6: Distribution of average OD³R routing table entries per node without link failures for different topologies. Packet destinations are chosen randomly.

TABLE II: Simulation wall-clock time per computation.

		Disjoint Path Computation Time		
	Topology	Min	Avg	Max
Generalized	Grid 4x4	0.007 ms	0.016 ms	5.109 ms
	Grid 5x5	0.008 ms	0.024 ms	3.191 ms
	Grid 6x6	0.010 ms	0.036 ms	3.832 ms
	Fullgraph 16	0.013 ms	0.020 ms	2.315 ms
	Fullgraph 25	0.025 ms	0.037 ms	2.709 ms
	Fullgraph 64	0.132 ms	0.242 ms	11.498 ms
ITZ [9]	Abilene	0.005 ms	0.012 ms	0.139 ms
	BT Europe	0.006 ms	0.016 ms	1.676 ms
	BT N.A.	0.009 ms	0.040 ms	4.028 ms
	DFN	0.012 ms	0.046 ms	7.269 ms
Random	N100E200	0.021 ms	0.257 ms	11.169 ms
	N100E350	0.147 ms	0.359 ms	13.375 ms
	N200E375	0.052 ms	1.192 ms	30.267 ms
	N200E580	0.770 ms	1.589 ms	31.350 ms

route maintenance. As the computation of the path hashes as well as the packet handling logic increase the computation overhead only slightly compared to the general packet handling logic, the overhead from these is not further investigated here. Instead, the following part examines the overhead of the route computation in OD³R.

OD³R uses the Suurballe and Tarjan algorithm [18] for route computation. Hence, the computation of a single path has the same runtime complexity as Dijkstra's algorithm (cf. [18]). The reactive, on-demand behavior of OD³R limits the route computations to actually used routes. However, compared to a traditional single-path routing mechanism, OD³R needs to compute the disjoint paths for each destination separately instead of computing the shortest path tree for all destinations at once. This means that each time a router receives a disjoint path request without a matching input route entry for this destination, it needs to compute the disjoint paths. The number of disjoint path computations per node depends on the number of destinations times the number of changes in the topology since this triggers a re-computation of the disjoint paths. Additional computations are necessary if already computed routes decay before they are used again.

The simulation results verify the assumption that the number of disjoint path computations per node depends on the number of destinations and topology changes. Furthermore, we observe that the number of path computations in large networks with many destinations is higher, as re-computations for decayed paths are needed.

In addition, we examined the wall-clock time spent in a single disjoint path computation, which is shown in Table II. The observed average runtime for the computation of the disjoint paths was between 10 µs and 2 ms. However, as this computation time is measured in a simulator, which is not concerned with accurate runtime measurements, the measured time only reflects the approximate scale.

D. Communication Overhead

Another important aspect is the communication overhead of OD³R. Obviously, if redundant transmissions are used, twice the traffic is sent through the network. However, as the concrete usage of the paths is left to the ED, here, we only consider the per-packet overhead of OD³R.

The extra communication overhead is caused by the information added to the packets by OD³R through the additional IP options. The main contributors are the *Residual Path ID* option with a size of 8 B including padding and the *source route exchange* option with a size between 4 B and 40 B. Which option is added to a packet depends on the current state, i.e., if the path needs to be exchanged, refreshed, or is still set up. This means that besides the path length, the sending rate and the refreshment rate impact the average per-packet overhead of OD³R. For sending rates much larger than the refreshment rate, the average per-packet converges to 8 B.

In all simulation runs with the destination selection happening once, this behavior was reflected, as the average OD³R per-packet overhead is approximately 8 B independent from the topology or failure scenario. However, in the destination selection mode, in which the destination is chosen per packet, we observed a larger span and an increasing overhead with larger topologies and longer paths. Figure 7 shows the average per-packet overhead observed by each node in the Internet Topology Zoo and random topologies without any failure. The per-packet overhead increases in larger networks since more source route exchanges are required with an increasing number of destinations. Furthermore, we can see that the overhead increases for networks with longer paths since the length of the source exchange option increases for longer paths. We observed a small increase in the overhead for the multi-failure scenarios. Overall, we observed an average OD³R per-packet overhead between 8 B and 14 B. For a single destination per node, the observed per-packet overhead is 8 B across all topologies and different scenarios.

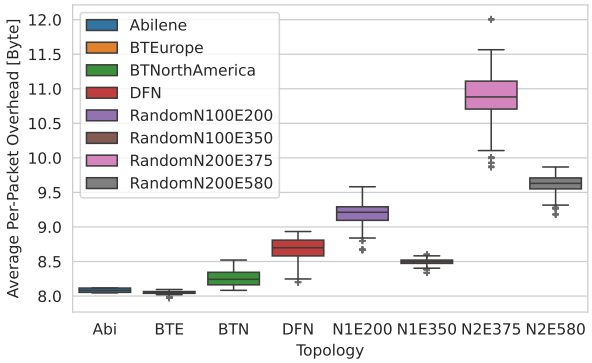


Fig. 7: Additional average per-packet overhead of OD³R, compared to normal protocol overhead, without link failures for different topologies. Packet destinations are chosen randomly. Generally, overhead increases for larger topologies and more destinations.

VII. Conclusion

The resilience of communication networks is increased by combining dynamic routing with disjoint paths, as demonstrated in this paper with *On-Demand Disjoint Dynamic Routing* (OD³R). We identified and addressed key challenges that arise from this combination and demonstrated the feasibility of OD³R in a simulative evaluation using various topologies and network parameters. Specifically, the results highlight that OD³R can guarantee at least one functional path in case of individual link failures while preserving adaptivity to changes in the network topology. Furthermore, OD³R increases network resilience even under multiple, simultaneously occurring link failures. This increase is caused by the usage of disjoint paths, which dynamically adapt to the topology. OD³R comes with the tradeoff of increased routing table entries, mainly influenced by the number of destinations and the average path lengths. Throughout all simulations, we observed that the average number of routing table entries was roughly between 3 and 4 times the number of destinations. This is significantly less than the naive approach but more than other, yet more computationally expensive, approaches, e.g., [7], [20]. The computation overhead introduced by OD³R is dominated by the overhead of the route computation, which is similar to computing shortest paths using Dijkstra's algorithm. However, each node must compute each destination's disjoint paths separately. OD³R requires no coordination effort and enables on-demand dynamic routing using disjoint paths with low computational overhead. This increase in network resilience comes with the tradeoff of higher average per-packet overhead than comparable but non-dynamic approaches [7].

Future research could look into further improvements and adaptations of OD³R, e.g., the possibility of using partially disjoint paths, adaptations for IPv6, or decentralized route computation algorithms. Furthermore, transferring our concepts into Software-Defined Networking (SDN) seems very interesting. Overall, our results highlight that *On-Demand Disjoint Dynamic Routing* (OD³R) provides a suitable approach for dynamic routing with disjoint paths in wired IP networks with reasonable overhead, maintaining routing adaptivity while also increasing network resilience.

Acknowledgment

This work has been partly funded by the LOEWE initiative (Hessen, Germany) within the emergenCITY center and conducted in collaboration with Deutsche Bahn AG as part of the work group CYSIS.

References

[1] G. S. Malkin, "RIP Version 2," Internet RFC, STD 56, November 1998.
[2] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, and R. White, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)," Internet RFC, RFC 7868, May 2016.
[3] J. Moy, "OSPF Version 2," Internet RFC, STD 54, April 1998.
[4] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," Internet Requests for Comments, RFC 1195, December 1990.
[5] M. Medard, S. Finn, R. Barry, and R. Gallager, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Transactions on Networking*, vol. 7, no. 5, 1999.
[6] G. Enyedi, G. Rétvári, P. Szilágyi, and A. Császár, "IP fast ReRoute: Lightweight not-via," in *Proc. NETWORKING 2009*. Springer, 2009.
[7] P. Babarczi, G. Retvari, L. Ronyai, and J. Tapolcai, "Routing on the Shortest Pairs of Disjoint Paths," in *IFIP Networking Conf, IEEE*, 2022.
[8] F. Aubry, S. Vissicchio, O. Bonaventure, and Y. Deville, "Robustly disjoint paths with segment routing," in *14th Intl. Conf. on emerging Networking EXperiments and Technologies (CoNEXT)*. ACM, 2018.
[9] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.
[10] J. Tapolcai, G. Retvari, P. Babarczi, E. R. Berczi-Kovacs, P. Kristof, and G. Enyedi, "Scalable and Efficient Multipath Routing: Complexity and Algorithms," in *Intl. Conf. on Network Protocols*. IEEE, 2015.
[11] G. Xue, L. Chen, and K. Thulasiraman, "QoS issues in redundant trees for protection in vertex-redundant or edge-redundant graphs," in *Proc. International Conference on Communications (ICC)*. IEEE, 2002.
[12] ——, "Delay reduction in redundant trees for preplanned protection against single link/node failure in 2-connected graphs," in *Proc. Global Telecommunications Conference (GLOBECOM)*. IEEE, 2002.
[13] W. Zhang, G. Xue, J. Tang, and K. Thulasiraman, "Faster Algorithms for Construction of Recovery Trees Enhancing QoP and QoS," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 642–655, June 2008.
[14] S. Ramasubramanian, M. Harkara, and M. Krunz, "Linear time distributed construction of colored trees for disjoint multipath routing," *Computer Networks*, vol. 51, no. 10, pp. 2854–2866, July 2007.
[15] G. Jayavelu, S. Ramasubramanian, and O. Younis, "Maintaining colored trees for disjoint multipath routing under node failures," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 346–359, Feb 2009.
[16] G. Enyedi and G. Rétvári, "Finding multiple maximally redundant trees in linear time," *Periodica Polytechnica Electrical Engineering*, 2010.
[17] Y. A. Mtawa, A. Haque, and G. Sidebottom, "Disjoint-path segment routing: Network reliability perspective," in *2022 International Wireless Communications and Mobile Computing (IWCMC)*. IEEE, May 2022.
[18] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Networks*, vol. 14, no. 2, pp. 325–336, 1984.
[19] R. Ogier, V. Rutenburg, and N. Shacham, "Distributed algorithms for computing shortest pairs of disjoint paths," *IEEE Transactions on Information Theory*, vol. 39, no. 2, pp. 443–455, March 1993.
[20] A. Itai and M. Rodeh, "The multi-tree approach to reliability in distributed networks," *Information and Computation*, 1988.
[21] B. Jenkins, "Hash functions for hash table lookup," Nov 2013, [Accessed 01-Feb-2023]. [Online]. Available: http://www.burtleburtle.net/bob/hash/doobs.html
[22] H. T. Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi, "BANANAS: An Evolutionary Framework for Explicit and Multipath Routing in the Internet," in *ACM SIGCOMM Workshop on FDNA*, 2003.
[23] R. Bless, M. Zitterbart, Z. Despotovic, and A. Hecker, "KIRA: Distributed Scalable ID-based Routing with Fast Forwarding," in *IFIP Networking Conf, IEEE*, 2022.