Destination-Specific Maximally Redundant Trees: Design, Performance Comparison, and Applications

Wolfgang Braun, Daniel Merling, and Michael Menth
University of Tübingen, Department of Computer Science, Germany

***Abstract*—Resilient networks react to a failure by reconfiguring routing tables in a network-wide manner and utilizing fast reroute (FRR) in the meanwhile. FRR uses pre-computed backup paths to avoid the failure locally. The existing FRR approaches for IP networks and Software-Defined Networking (SDN) differ in regard to computational complexity, backup path length, additional forwarding state and required capacities.**

Maximally Redundant Trees (MRTs) are standardized by the IETF and are already available for IP and MPLS networks. In this paper, we propose destination-specific MRTs (dMRTs), an improved variant of MRTs, for protection of IP networks. This mechanism reduces the generated path lengths but increases the computational complexity of the MRT algorithm. We analyze and compare our proposal with simple MRTs and MPLS FRR. We show on a large and publicly available network data base that dMRTs provide significant shorter backup paths compared to simple MRTs and require less forwarding state and network capacities than MPLS FRR. Adapting an existing FRR mechanism for IP networks to SDN enables its deployment in hybrid-SDN networks. Thus, we outline implementation options of dMRTs in IP and MPLS networks and (d)MRTs in SDN. Finally, we discuss the computational overhead of our proposal in the context of (decentralized) IP networks and (centralized) SDNs.

***Index Terms*—Resilience, Software-Defined Networking, IP networks, Scalability, Resource Management**

I. Introduction

In decentralized networks such as OSPF or MPLS networks, traffic is routed along shortest paths. When failures occur, the devices inform each other about the malfunction and recompute the forwarding tables in a network-wide consistent fashion. This reconfiguration process can be slow in large networks and meanwhile traffic may be lost. Fast reroute (FRR) techniques have been introduced so that failure detecting nodes (points of local repair) are able to immediately reroute traffic to alternate pre-computed paths. This ensures successful delivery from the detection of the failure until shortest paths have converged throughout the network. Failures are generally detected sub 50 ms due to the usage of bidirectional forwarding detection (BFD) components. Thus, FRR minimize the traffic loss that occurs when network failures occur.

Maximally Redundant Trees [1], [2] constitute a FRR mechanism for IP and MPLS networks. They are standardized by the IETF and provide full protection against single link and single node failures in 2-node-connected networks. MRTs define two backup forwarding structures (red and blue) such that at least one of them is working in any single-component failure scenario. If the next hop fails, the packet is colored red or blue, depending on which structure is still working, and forwarded accordingly. Computational overhead of MRTs is optimized for IP networks by leveraging an Almost Directed Acyclic Graph (ADAG). It utilizes a root node to effectively calculate the backup paths. However, in an earlier work [3] we found that MRTs may lead to excessively long backup paths. Therefore, we propose to adapt the MRT calculation to reduce path lengths and generate a more intuitive path layout. We propose to use destination-specific ADAGs to determine the red and blue backup forwarding entries by using the corresponding destination as the root node for ADAG computation. Therefore, we call this method destination-specific MRTs (dMRTs). This approach increases computational complexity but does not cause changes to the forwarding mechanism or state requirements. We compare dMRTs to simple MRTs and additionally to another well-known existing resilience mechanism: multi-protocol label switching (MPLS) FRR. We analyze these mechanisms on a publicly available large network data set and compare them with regard to path lengths, required network capacities, and number of additional forwarding entries. As every node within a decentralized network needs to perform its own independent calculations, the algorithmic complexity of a FRR mechanism is relevant. Thus, we discuss dMRTs in terms of computational overhead in such networks. Furthermore, we provide suggestions for the implementation of dMRTs in IP and MPLS networks.

For SDN, failure handling differs in comparison to decentralized networks. If a malfunction occurs, the detecting switch informs the controller which in turn computes new forwarding entries and installs them on the switches. In [4], the authors measured on a real testbed that their controller reacts within 80 – 100 ms. They also identified that the restoration time depends on the path lengths, number of flows to be restored and network size. In our opinion, a FRR mechanism with full coverage is crucial for SDN to mitigate the effects of failures, especially when control elements observe high load and might not be able to repair failures quickly. Adapting an existing IP FRR mechanism to SDN grants the advantage of its possible deployment in hybrid-SDN networks. This allows introducing SDN devices more easily in existing networks where resilience is an important aspect. Therefore we provide suggestions how (d)MRTs can be implemented OpenFlow. Finally, we discuss computational overhead of MRTs and dMRTs in SDN networks.

The authors acknowledge the funding by the Deutsche Forschungsgemeinschaft (DFG) under grant ME2727/1-1. The authors alone are responsible for the content of the paper.

The remainder of the paper is structured as follows. Section II explains MRTs and introduces the novel dMRTs. We explain MPLS and MPLS FRR in Section III. Our methodology and results are presented in Section IV. Section V discusses the applicability of dMRTs in IP and SDN networks. Section VI presents related work. Finally, Section VII summarizes this work and gives conclusions.

II. Destination-based Maximally Redundant Trees

In this section, we first explain MRTs and their computational principles. We provide insight about the disadvantages of this approach and propose our improvement to MRTs called dMRTs.

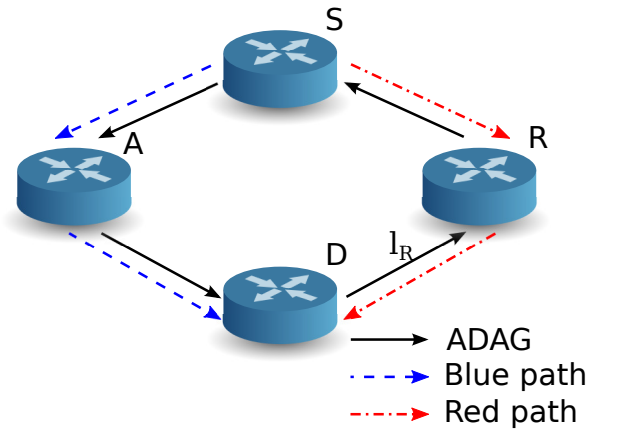
A. Maximally Redundant Trees

MRTs are standardized in the IETF [1], [2], and can be implemented in IP and MPLS networks to protect against single link and single node failures. The main motivations for MRTs are to achieve 100% failure coverage without introducing too much forwarding state as with previously discussed approaches such as not-via addresses [5] in the IETF. In addition, MRTs should not require much computational complexity since IP routers generally provide limited computational resources.

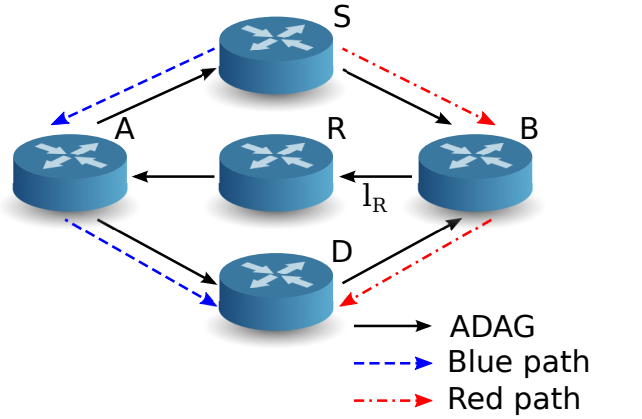
MRTs achieve these requirements by calculating two additional routing topologies per source-destination pair that are referred to as the red and blue topologies. Each red-blue pair leads to maximally redundant paths, i.e., a packet sent on the blue topology from a source S towards destination D traverses different links and nodes than a packet sent on the red topology from S to D if the network topology is redundant enough. If the network topology does not allow for redundant paths, MRTs ensure that only a minimum number of elements are shared by each red and the blue path pair. The backup paths are generally implemented in such a way that only a single additional entry is required in a node per destination and per routing topology. Thus, MRTs scale linear with the number of nodes in the network and require 200% additional forwarding entries which is a desired property of MRTs in the IETF.

1) Computation of Backup Paths: MRTs leverage a special structure to keep computational complexity low. The structure is called ADAG, which is described in great detail in the IETF draft [1], and is computed in linear time [6]. In the following, we briefly outline the ADAG structure and how all next-hops for the red and blue topologies can be inferred from it. An arbitrary node of the network will be the root node R of the ADAG. All (bidirectional) links of the network topology are directed in such a way that every link is part of at least one cycle that contains the root R. Moreover, all cycles must enter the root node through a single link l_R. Removing the link l_R removes all cycles from the structure converting the ADAG in an directed acyclic graph (DAG), hence its name.

The ADAG structure is used to compute two disjoint backup paths. Consider a point of local repair (PLR) S and a destination D. Figure 1a illustrates the case that both S and D are



(a) Source and destination are part of a single cycle in the ADAG structure.



(b) Source and destination are on two different cycles in the ADAG structure.

Fig. 1: Construction of red and blue path using the ADAG structure.

located on the same cycle. The blue backup path follows the ADAG direction and the red backup path follows the reverse direction. If several cycles exist where S and D are part of, the shortest cycle in terms of link costs is considered for backup path construction. Figure 1b shows the case when S and D are not part of the same cycle. Then, they must be part of two different cycles that intersect at least in the root node R and link l_R due to the ADAG properties. The blue backup path first goes against the ADAG direction towards an intersection node with the cycle that D is part of. From this intersection node, it follows the ADAG towards D. The red backup path is constructed using the reverse direction. At first, it follows the ADAG to a (different) intersection node and from there it goes against the ADAG towards D. These two backup paths for the PLR S are computed using two modified shortest path first (SPF) computations rooted at S. This may not lead to the overall shortest backup paths but red and blue hops can be inferred for all destinations using both cases shown in Figure 1.

2) Selection of Backup Paths in Failure Cases: Traffic is normally forwarded on the primary path. When a failure occurs, the PLR determines whether the red or blue backup

avoids the failure and sends it along the corresponding path. The computation of this decision is based on the information gathered during backup path construction. The ADAG provides a partial topological order of the nodes and the SPF computations for the backup paths contain information about reachability of nodes by red or blue paths. [1, Figure 25] shows that the construction method utilizing an ADAG ensures that at least one of the backup paths cannot contain the failed element. This path is selected and the traffic is sent to the appropriate color (Section V-B1) using tunneling mechanisms.

B. Destination-specific MRTs

In a previous work [3], we found MRTs can cause excessive path lengths, i.e., up to 16 hops in a 16 node network, depending on the network topology and the selected root node of the ADAG. Therefore, appropriate root node selection is highly significant for MRTs. We think that the reason for long backup paths with MRTs is caused by the way the paths are constructed. The shortest cycle that S and D are part of may be very long when both nodes are quite distant to R. This property depends on the initially chosen root node for the ADAG and affects every source destination pair. If the failure is on the same side of the cycle as S and D, the packet has to be redirected over the other side of the cycle traversing the root node causing a long backup path. The same logic applies to the second case, when S and D are not part of the same cycle.

Therefore, for each destination D we propose to use a separate ADAG rooted at D. Then, the PLR and D always are part of the same cycle due to the ADAG construction. In most cases this will lead to short cycles and backup paths because instead of three nodes (R, S, D) only two nodes (R/D, S) have to part of the cycle. This optimization of MRTs seems kind of obvious to make. However, this has significant implications on the computational complexity on the algorithm. For each of the n possible destinations an ADAG and two backup paths have to be calculated because the root node and thus the structure of the ADAG changes. Note that a high number of SPF calculations is very undesirable for IP networks. However, we think that SDN justifies higher algorithmic complexity due to higher computational resources resulting in shorter backup paths.

III. MPLS and Fast Reroute

In this section, we briefly explain MPLS and present MPLS-FRR and its two variants called one-on-one backup and facility backup. Then, we discuss their state requirements.

A. MPLS Forwarding

MPLS follows a connection-oriented forwarding approach. A connection is established for a forwarding equivalence class (FEC) between two routers. In IP networks, a FEC is generally an IP prefix. When an IP packet enters the MPLS domain, the ingress node identifies the appropriate MPLS path called label switched path (LSP). The node pushes the MPLS label associated with the path on the packet. The label is only locally relevant and has to be changed from hop to hop. Therefore, each MPLS node checks the label and retrieves the next-hop (NH) and its associated label for the destination. It swaps the label to the new one and forwards it to the NH. At the final node, the MPLS label is popped and the packet is forwarded by IP address. Thus, MPLS is useful to create overlay networks, e.g., for BGP, and as lightweight tunnel mechanism.

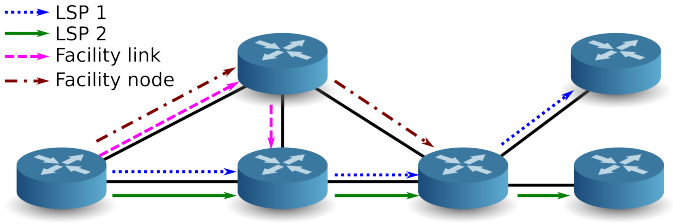


Fig. 2: MPLS facility backup: link and node protection. The backup path bypasses the failed element. The backup tunnel can be leveraged for different LSP simultaneously.

B. MPLS FRR

MPLS fast reroute is based on the local repair principle [7]. A node that detects the failure, called PLR, redirects traffic on backup paths that avoids the failed network element. Backup paths may be constructed to reroute around link or node failures, which is a policy decision made before computation. MPLS-FRR provides two backup path options for the redirection: one-to-one backup (detour) and facility backup (bypass).

Detour backup paths have to be setup for each LSP in the network. For each node, a detour is installed around the failed element. The detour path merges with the MPLS path farther downstream at the node called the merge point (MP). The MP may be any node of the remaining path. Note that, for an LSP of length n, $n-1$ detour LSPs are required. When a failure is detected, the PLR swaps the label of the default path with the label of the detour path. When the packet arrives at the merge point, the labels of the primary paths are used again.

Facility backup requires additional paths for each failed network element, i.e., a tunnel around the failed link to the NH or multiple tunnels to the next-next-hops (NNHs) around failed nodes. The backup paths are illustrated in Figure 2. In case of a failure, the PLR swaps the MPLS label appropriate for the end of the backup path. The PLR pushes the backup label on the MPLS label stack and forwards the packet. The backup label is removed at the last hop of the backup path. Finally, the packet is forwarded normally.

C. MPLS FRR State Requirements

Detour backup paths provide more degrees of freedom since they allow to merge with the primary path farther downstream. Facility backup always tunnels to the NH or NNHs and may result in longer paths. However, detours are LSP specific and cannot be reused by other LSPs as with facility backup as illustrated in Figure 2. In [8], the authors showed that the state required for facility backup is less than for detour backup.

Since forwarding state is of concern especially for SDN, we focus on facility backup in the remainder of the paper. MPLS_l refers to the link and MPLS_n to the node protecting variant.

IV. Results

In this section, we discuss the considered failure scenarios, traffic assumptions, and analyzed metrics. We compare dMRTs with MPLS FRR and with several MRTs variants that are optimized to different metrics.

A. Methodology

We evaluate the flow layout for MRTs and dMRTs in the failure-free case (FF) and in single link failure scenarios (SLF). In FF, (d)MRTs and MPLS are routed both using the shortest path principle. We use unit link costs as input for the routing mechanisms due to the lack of information specified in our data set described below. In failure cases, packets are rerouted using the pre-established backup paths of the selected routing mechanism.

We consider networks from the topology zoo [9] that compromises various commercial and research networks. Nodes that are connected by only a single vertex cannot be protected against a failure of the connecting link. Thus, we modify the topologies by removing such nodes. Furthermore, topologies with a vertex count greater than 200 are omitted. The data set still contains 160 different topologies of various sizes which have 4 to 166 vertices, with an average of 27.6. The number of edges varies from 6 to 212 with an average of approximately 38. We think that the data set is large enough to show the generality of the results presented below.

The topology zoo does not provide traffic matrices. We generate for each topology a homogeneous traffic matrix that consists of traffic demands of equal load for each source-destination pair.

B. Metrics

We consider three metrics for our evaluation: path lengths, required relative capacities, and number of forwarding entries needed. We provide a cumulative distribution function (CDF) for each metric over all networks under study that allows to identify the trends of the different methods with regard to a specific metric on the large data base.

1) Path length prolongation: We measure the path length of a flow in hops in the failure-free case and for each single link failure. The maximum and average path lengths in the failure-free case are $p^{\text{FF}}_{\text{max}}$ and $p^{\text{FF}}_{\text{avg}}$. We count the maximum and average number of hops in SLF of the *rerouted* flows $p^{\text{SLF}}_{\text{max}}$ and $p^{\text{SLF}}_{\text{avg}}$. Finally, we report the average and maximum path prolongations (p_{avg} and p_{max}) for SLF caused by the rerouting process. They are computed by $p_{\text{avg}} = p^{\text{SLF}}_{\text{avg}}/p^{\text{FF}}_{\text{avg}}$ and $p_{\text{max}} = p^{\text{SLF}}_{\text{max}}/p^{\text{FF}}_{\text{max}}$, respectively. Using normalized path prolongations in this evaluation is especially helpful because the topologies under study differ in size and, thus, allow easy generalization of the results in the large analyzed data set.

2) Required Relative Capacities: We determine the utilization of each link in the failure-free case and for every single link failure. For both cases we compute the sum and the maximum of all links as network capacity and maximum link capacity. The relative network capacity C_n and the relative maximum link capacity C_l is reported relative to the failure-free case.

3) Number of additional forwarding entries: To provide connectivity among all nodes in a network with n nodes, $n-1$ forwarding entries are required per node. MPLS and MRTs install additional forwarding entries to implement the backup paths. MRTs require two additional entries per destination. For MPLS facility backup, the number of additional entries varies for each node. We provide both the average and maximum percentage of additional entries for all nodes.

C. MRT Variants

While dMRTs leverage a single ADAG structure for each destination, MRTs are based on a single ADAG using a single root node. We already showed in [3] that the MRTs performance of path lengths and link utilization is heavily dependent on the selection of the root node. Therefore, we compare dMRTs to different MRT root node configurations. We show only MRTs with root nodes in the results that provide the best performance results for a specific metric. We consider MRTs optimized on the maximum path prolongation p_{max} (MRT_p) and relative network capacity C_n (MRT_C).

D. Required Relative Capacities

We present the required relative capacities in this section. We analyze (d)MRTs for each topology and provide C_n and C_l. The required relative network capacity C_n is shown in Figure 3 when single link failures are considered. The figure shows hardly any difference with regard to C_n between MRTs and dMRTs. Network capacities increased by a factor between 2 and 2.4 for most topologies compared to the failure-free case. MRT_C results in marginal less required capacity than dMRT and MRT_p. MPLS_n requires slightly more network capacities than (d)MRTs and MPLS_l the most. The difference between MRT_C and MPLS_l is between about 0.15 and 0.34 for 90% of all topologies. This can be explained by the fact that MPLS_l redirects closely to the failed link which may cause local hotspots more easily than for MPLS_n and (d)MRTs.

Figure 4 shows the CDF for maximum relative link capacity. We observe the same trends as for the relative network capacity. Rerouting using (d)MRTs increases the required maximum link capacity by a factor between 1 and 2 in all topologies. dMRTs results in the least maximum link capacity but the difference between MRTs and dMRTs is only marginal. The difference between dMRTs and MPLS_l is about 0.36 and 0.5 for 90% of all topologies.

We conclude that dMRTs and MRTs do not differentiate much concerning required capacities caused by rerouting. (d)MRTs require approximately 2 – 2.4 more capacity to protect against failures compared to the failure-free case. (d)MRTs require less capacities than MPLS. The difference is more notable for MPLS_l than MPLS_n.

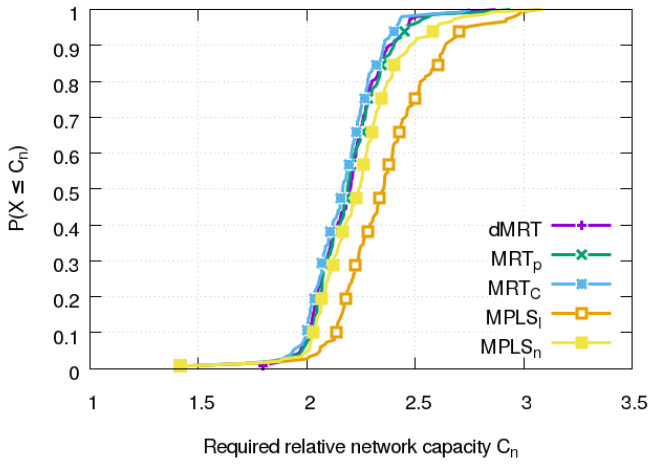


Fig. 3: CDF of required relative network capacity C_n.

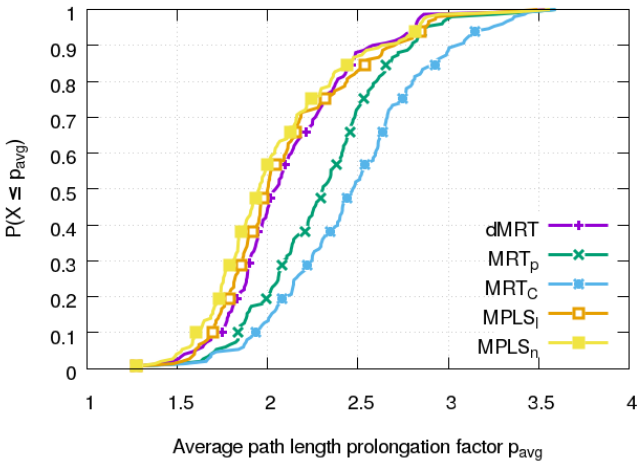


Fig. 5: CDF of average path length prolongation p_{avg}.

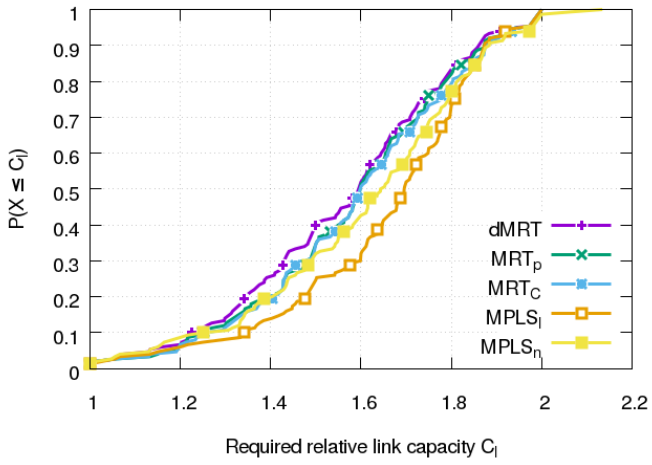


Fig. 4: CDF of required relative maximum link capacity C_l.

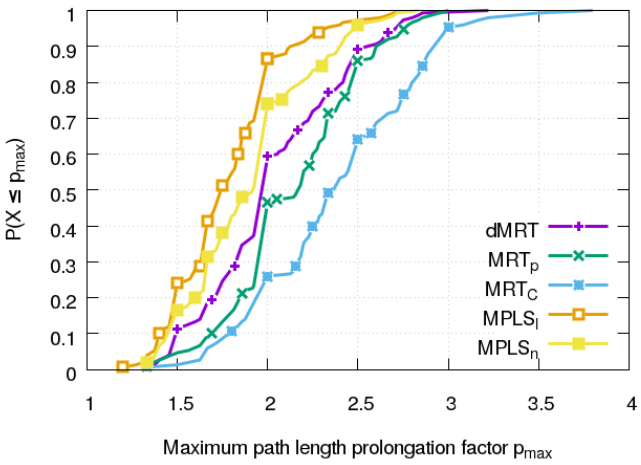


Fig. 6: CDF of maximum path length prolongation p_{max}.

E. Path Length Prolongation

In this section, we present the path length prolongation caused by the rerouting mechanisms. Figure 5 shows the average path length prolongation for all considered topologies as a CDF. MPLS FRR and dMRTs cause the shortest and similar long backup paths with an average path prolongation of approximately 1.75 – 2.7 for most topologies. MRTs result in longer path prolongations. MRT_p is in the range from 1.8 to 3 notably longer than MPLS or dMRTs. MRT_C causes the longest average path lengths with a p_{avg} between 2 up to 3.5 for about 90% of the topologies. The maximum path prolongation p_{max} is illustrated in Figure 6. Notable differences for p_{max} are observed for all methods. MPLS_l has for 87% of the networks a $p_{\text{max}} \leq 2$ but MPLS_n only for 73%. Both can result in a factor up to 2.75. dMRTs and MRT_p have 60% and 46% of the networks with $p_{\text{max}} \leq 2$ and in the worst case $p_{\text{max}} \approx 3.2$. For MRT_C, only 26% of the networks result in $p_{\text{max}} \leq 2$ and a maximum factor of 3.7.

dMRTs clearly lead to shorter backup paths than MRTs even if the root node is optimized on backup path length. We observed that dMRTs have on average similar long backup paths as MPLS FRR. The longest backup paths for dMRTs are clearly longer than for MPLS FRR, especially for MPLS_l. Thus, we expect dMRTs only have longer backup paths for few failure scenarios and be competitive to MPLS with regard to path lengths in general.

F. Number of Additional Forwarding Entries

Figure 7 illustrates the required state for MPLS compared to MRTs as a CDF. The number of additional forwarding entries for MRT and dMRT are fixed by 200%. For MPLS, we differentiate between link and node protection and illustrate the average and the maximum number of forwarding entries. The average number of entries for link-protecting MPLS is below the number for MRTs. For approximately 25% of all networks, the maximum number is larger than 200%. More

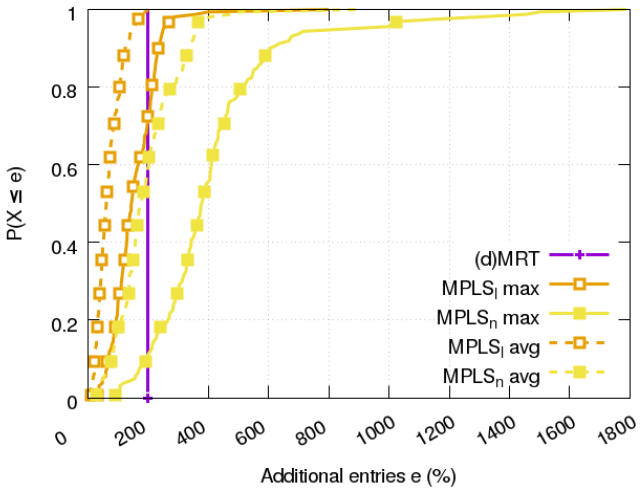


Fig. 7: CDF of additional forwarding entries.

than 95% networks result in less than 300% and in few cases the number of maximum forwarding entries reaches about 500%. Node-protecting MPLS requires significantly more forwarding entries. For 22% of the networks, the average number of forwarding entries is between 200% and 400%. For few networks the number increases up to 900%. We found that only 8% of the networks require less than 200% maximum forwarding entries for MPLS. For about 80% of the networks the number is between 200% and 600%. The remaining 10% of the networks require over 600% up to 1800% additional forwarding entries.

We conclude that (d)MRTs clearly outperform MPLS with regard to state requirements on the basis of two main facts. MRTs do protect against both link and node failures and require significantly less state to achieve node protection. Although link-protecting MPLS requires for about 60% of the networks less state, MRTs are very close to them but achieve higher protection.

G. Summary

We evaluated (d)MRTs for various networks. The results show that both lead to very similar bandwidth requirements when single link failures are considered. We analyzed the path prolongation that is caused when packets are rerouted. We found that dMRTs lead to the shorter backup paths than MRTs. On average dMRTs perform similar to MPLS FRR. When MRTs are optimized on path lengths (MRT$_p$), we observe notable longer backup paths on average and slightly longer for the worst cases compared to dMRTs. Both MRT$_p$ and dMRTs provide significant shorter backup paths compared to MRTs optimized on bandwidth requirements.

Considering additional forwarding entries, (d)MRTs produce clearly better results than MPLS. Although MPLS$_l$ requires slightly fewer entries it only protects against single link failures. (d)MRTs cover every single component failure and the comparable MPLS$_n$ mechanism relies on significantly more forwarding entries for the same degree of protection.

V. Application of dMRTs in IP and SDN

We discuss the computation overhead of this extension for IP and SDN networks. Finally, we review the implementation options for IP networks and outline how (d)MRTs can be implemented in OpenFlow.

A. Algorithmic Complexity of dMRTs: IP vs SDN

In this section, we discuss the computational overhead introduced by dMRTs. In IP networks, every router has to calculate its primary and backup NH to each destination. In an SDN-based network, the computation of forwarding entries for the network devices is performed in one or more control elements. We assume a single centralized controller to estimate the algorithm complexity for an SDN approach. Note that a distributed controller approach can reduce the overall computation of a single controller, e.g., by dividing the network in disjoint parts.

1) Complexity for MRTs: We first outline the complexity for simple MRTs to provide comparability to dMRTs. An IP node has to compute the default hops using a SPF computation. The ADAG computation is assumed here to be in linear time [6] but may be higher for advanced approaches. Backup path calculation requires two modified (l_R is disabled) SPF computations as discussed in Section II-A1. The backup path selection (Section II-A2) is done in linear time leveraging information gathered by the ADAG construction and blue and red hop generation [1], [6]. A SDN controller computes the primary NH for all nodes in the network: n SPF computations. The controller needs to compute only one ADAG as all paths follow the single ADAG structure. The controller has to compute for each node the backup paths; this results in $2n$ modified SPF computations.

2) Complexity for dMRTs: For dMRTs the computation of backup paths in IP nodes has to change significantly. The node has to compute n ADAGs which changes the linear computation in a polynomial one. The red and blue hops must be computed for each destination and its corresponding ADAG resulting in $2n$ SPF computations. Backup path selection can still be performed linear. For SDN, the computational overhead compared to simple MRTs does not change too much. The only difference is that the controller must also compute n ADAGs. Since computation of the controller was already polynomial, only a small overhead is expected for dMRTs.

B. Implementation of (d)MRTs

1) Implementation in IP and MPLS Networks: The proposed improvement dMRT differentiates only with regard to path layout but does not require changes to the existing forwarding techniques. We briefly outline how MRTs are implemented in current networks and leave the details to the IETF draft [2]. Signaling of backup paths is implemented using tunneling: IP-in-IP tunnels and MPLS tunnels are supported. Each node announces its address or label of the primary path as well as the blue and red backup paths in the network. In failure cases, the PLR encapsulates the traffic using the address of the appropriate color of the destination.

2) Implementation in OpenFlow: We propose to implement (d)MRTs in OpenFlow analogous to the implementation existing for IP networks. OpenFlow 1.1 or higher is required to support the implementation of (d)MRTs since it introduces both MPLS support and group tables to implement backup mechanisms.

We explain the structure of the OpenFlow pipeline and how the controller configures the switches for (d)MRTs. The OpenFlow pipeline consists of one or more flow tables. The controller installs an entry for each destination that points to a group table entry of type fast-failover. An entry consists of a list of action buckets which are executed consecutively until the first bucket succeeds. Thus, the controller installs the forwarding next-hop in the first bucket and the encapsulation to the appropriate colored destination in the second bucket.

MPLS is an optional feature for OpenFlow 1.1, and may not be available on all OpenFlow switches. If MPLS is available, the controller simply implements (d)MRTs by pushing the appropriate MPLS label on the packet as action in the group table entry. We recommend MPLS tunneling since it is more lightweight than IP-in-IP tunneling. OpenFlow can also be configured for IP-in-IP tunneling but lacks actions for encapsulation with an additional IP header. We suggest two workarounds for this deficiency. The first one is to create additional interfaces on the switch that perform the appropriate encapsulation. One interface is required for each color and destination pair that the switch may address. The OF-CONFIG protocol [10] can be utilized by the controller to configure the tunnel interfaces on the switches. The second workaround is to extend the OpenFlow protocol by actions for IP encapsulation and decapsulation. This approach was implemented in [11]. Then, the controller can set up the IP tunnels analogously to the MPLS approach.

The controller also installs the blue and red next-hops in the forwarding tables. Such an entry does not point to a group table but simple contains the actions to forward the packet or to decapsulate the packet. This prevents additional redirects in case of multiple failures which may result in forwarding loops. This is the behavior suggested in the MRT drafts.

VI. Related Work

Independent trees (ITrees) can provide resilient multipath routing by sending traffic over one tree and switching in failure cases to the other. Independent Directed Acyclic Graphs (IDAGs) [12] are an extension to independent trees that augment ITrees by directing and using more topology links to increase failure tolerance. This is different to (d)MRTs which forward the traffic normally on shortest paths and only use colored paths in failure cases. The authors analyze the path diversity and the path lengths on four networks for ITrees, IDAGs, and multiple ITrees. In [13], computation of node-resilient colored (independent) trees is investigated and optimized for fast convergence times.

Various approaches for IP FRR were proposed in the IETF. Loop-free alternates (LFAs) [14] redirect traffic to neighbors that avoid loops. They do not require additional forwarding entries but have limited coverage [15]. Remote LFAs (rLFAs) achieve full coverage by redirecting packets to alternate nodes using shortest paths for unit link costs [16]. General coverage guarantees cannot be made by rLFAs. Topology-independent LFAs [17] leverage source routing to guarantee coverage against single link failures independent of the topology. The combination of LFAs and MRTs is discussed in [18]. The main achievement of this work is to reduce the backup path length. Not-via addresses [5] follow the same path layout as MPLS facility backup but are not considered anymore in the IETF because of state and operational complexity. Multiple routing configurations (MRCs) [19] leverage additional routing topologies that are used in failure cases. While (d)MRTs introduce two additional routing topologies, MRCs may require more. A comparison of MPLS and IP resilience methods is given in [20], [21]. Failure insensitive routing (FIR) [22] encodes failed links in an additional packet label before redirecting packets to alternate neighbors. The information is used to forward the rerouted packets without introducing micro loops along precomputed backup paths.

Multiple approaches to increase reliability of the SDN control plane are discussed in [23]. Since (d)MRTs protect the data plane , we focus the related work for SDN around similar methods. The SDN controller reconfigures the network when failures occur. In [4], the authors measures reaction times of 80 – 100 ms in a real testbed. They found that the reconfiguration speed depends on the number of flows affected, path lengths and traffic burst in the control network. Restoration time can be significantly higher in larger networks. Since OpenFlow 1.1, fast-failover was introduced which allows for pre-established backup paths. A protection method based on MPLS-TP was developed in [24] before OpenFlow 1.1. They added a BFD component to the OpenFlow switch to locally detect failures and redirect packets to other interfaces without controller intervention. For the Open vSwitch, link failures can be detected below 30 ms depending on the configuration of the BFD. The authors of [25] encode primary and backup paths in the packet header. They showed the applicability of their approach in a virtual testbed. SPIDER [26] leverages additional state in the OpenFlow pipeline to provide an alternative and more adaptive solution to OpenFlow's fast-failover mechanism. Their path layout is similar to MPLS crankback routing and optimizations are possible. SDN-based LFAs have been proposed in [27]. A novel loop detection mechanism was added to LFAs that prevents micro-loops and increases protection but cannot guarantee full coverage. A similar approach to FIR was developed for SDN in [28]. The failed link or node is encoded in a packet label but the approach only addresses single failures. MRCs have been adopted for SDN in [29].

VII. Conclusion

Maximally Redundant Trees (MRTs) provide failure protection against single link and node failures in IP networks today. They require a number of additional forwarding entries that scales with the number of nodes but are known to cause long backup paths. In this paper, we proposed destination-specific

MRTs (dMRTs): a computational improvement for MRTs that does not require changes to the forwarding mechanism or additional state. We discussed that computational overhead is large in IP but not in SDN networks. We compared dMRTs with simple MRTs and MPLS facility backup on 160 representative topologies. We found that dMRTs provide shorter backup paths than simple MRTs but slightly longer backup paths than MPLS. However, dMRTs require significantly less additional forwarding entries compared to MPLS when the same level of protection is assumed.

Due to its scalability concerning forwarding entries, improvement in path layout, and a straightforward implementation for OpenFlow, we recommend the usage of dMRT in SDN networks. MRTs and dMRTs can be leveraged in hybrid-SDN networks, i.e. consisting of IP and SDN devices, since interoperabiltiy is given due to equal forwarding mechanism. Since computational complexity is significantly increased for IP networks, we suggest that dMRTs are only preferred to plain MRTs when computational overhead is acceptable for the size of the network.

REFERENCES

[1] G. Enyedi, A. Csaszar, A. Atlas, C. Bowers, and A. Gopalan, "RFC7811: Algorithm for Computing IP/LDP Fast Reroute Using Maximally Redundant Trees," Jun. 2016.

[2] A. Atlas, C. Bowers, and G. Enyedi, "RFC7812: An Architecture for IP/LDP Fast Reroute Using Maximally Redundant Trees (MRT-FRR)," Jun. 2016.

[3] M. Menth and W. Braun, "Performance Comparison of Not-Via Addresses and Maximally Redundant Trees (MRTs)," in *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, Apr. 2013.

[4] S. Sharma, D. Staessens, D. Colle, M. Pickavet, and P. Demeester, "OpenFlow: Meeting Carrier-Grade Recovery Requirements," *Computer Communications*, vol. 36, no. 6, pp. 656–665, 2013.

[5] S. Bryant, S. Previdi, and M. Shand, "RFC6981: A Framework for IP and MPLS Fast Reroute Using Not-Via Addresses," Jul. 2013.

[6] G. Enyedi, G. Retvaria, and A. Csaszar, "On Finding Maximally Redundant Trees in Strictly Linear Time," in *IEEE Symposium on Computers and Communications (ISCC)*, July 2009, pp. 206–211.

[7] P. Pan, G. Swallow, and A. Atlas, "RFC4090: Fast Reroute Extensions to RSVP-TE for LSP Tunnels," May 2005.

[8] R. Martin and M. Menth, "Backup Capacity Requirements for MPLS Fast Reroute," in 7^{th} *ITG Workshop on Photonic Networks*, Leipzig, Germany, Apr. 2006, pp. 95–102.

[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, Oct. 2011.

[10] OpenFlow Switch Consortium and others, "OpenFlow Management and Configuration Protocol 1.2 (OF-Config 1.2)," April 2015.

[11] S. Li, Y. Shao, S. Ma, N. Xue, S. Li, D. Hu, and Z. Zhu, "Flexible Traffic Engineering: When OpenFlow Meets Multi-Protocol IP-Forwarding," *IEEE Communications Letters*, vol. 18, no. 10, pp. 1699–1702, Oct. 2014.

[12] S. Cho, T. Elhourani, and S. Ramasubramanian, "Independent Directed Acyclic Graphs for Resilient Multipath Routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 153 –162, Feb. 2012.

[13] 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, 2009.

[14] A. Atlas and A. Zinin, "RFC5286: Basic Specification for IP Fast Reroute: Loop-Free Alternates ," Sep. 2008.

[15] L. Csikor, J. Tapolcai, and G. Retvari, "Optimizing IGP link costs for improving IP-level resilience with Loop-Free Alternates," *Computer Communications*, vol. 36, no. 6, pp. 645 – 655, Mar. 2013.

[16] L. Csikor and G. Retvari, "On Providing Fast Protection with Remote Loop-Free Alternates: Analyzing and Optimizing Unit Cost Networks," *Telecommunications Systems*, 2015.

[17] A. Bashandy, C. Filsfils, B. Decraene, S. Litkowski, and P. Francois, "Topology Independent Fast Reroute using Segment Routing," Jul. 2017.

[18] K. Kuang, S. Wang, and X. Wang, "Discussion on the Combination of Loop-Free Alternates and Maximally Redundant Trees for IP Networks Fast Reroute," in *IEEE International Conference on Communications (ICC)*, June 2014, pp. 1131–1136.

[19] A. Kvalbein, A. F. Hansen, T. Cicic, S. Gjessing, and O. Lysne, "Fast IP Network Recovery Using Multiple Routing Configurations," in *IEEE Infocom*, Barcelona, Spain, Apr. 2006.

[20] S. Rai, B. Mukherjee, and O. Deshpande, "IP Resilience within an Autonomous System: Current Approaches, Challenges, and Future Directions," *IEEE Communications Magazine*, vol. 43, no. 10, pp. 142–149, Oct. 2005.

[21] A. Raj and O. Ibe, "A Survey of IP and Multiprotocol Label Switching Fast Reroute Schemes," *Computer Networks*, vol. 51, no. 8, pp. 1882–1907, 2007.

[22] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast Local Rerouting for Handling Transient Link Failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 359–372, Apr. 2007.

[23] Y. E. Oktian, S.-G. Lee, H.-J. Lee, and J.-H. Lam, "Distributed SDN Controller System: A Survey on Design Choice," *Computer Networks*, vol. 121, pp. 100–111, 2017.

[24] J. Kempf, E. Bellagamba, A. Kern, D. Jocha, A. Takàcs, and P. Sköldström, "Scalable Fault Management for OpenFlow," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 6606–6610.

[25] R. M. Ramos, M. Martinello, and C. E. Rothenberg, "SlickFlow: Resilient Source Routing in Data Center Networks Unlocked by OpenFlow," in *IEEE Conference on Local Computer Networks (LCN)*, Oct. 2013.

[26] C. Cascone, L. Pollini, D. Sanvito, A. Capone, and B. Sansó, "SPIDER: Fault Resilient SDN Pipeline with Recovery Delay Guarantees," in *IEEE Conference on Network Softwarization (NetSoft)*, June 2016, pp. 296–302.

[27] W. Braun and M. Menth, "Loop-Free Alternates with Loop Detection for Fast Reroute in Software-Defined Carrier and Data Center Networks," *Journal of Network and Systems Management (JNSM)*, vol. 24, no. 3, 2016.

[28] N. L. M. van Adrichem, F. Iqbal, and F. A. Kuipers, "Backup Rules in Software-Defined Networks," in *IEEE Conference on Network Function Virtualization and Software-Defined Networking (NFV-SDN)*, Nov 2016, pp. 179–185.

[29] S. Cevher, M. Ulutas, S. Altun, and I. Hokelek, "Multi Topology Routing Based IP Fast Re-Route for Software Defined Networks," in *IEEE Symposium on Computers and Communications (ISCC)*, 2016.