Routing on the Shortest Pairs of Disjoint Paths

Péter Babarczi*, Gábor Rétvári*, Lajos Rónyai†, and János Tapolcai*
*Department of Telecommunications and Media Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
†Institute for Computer Science and Control, Eötvös Loránd Research Network; and Department of Algebra, Institute of Mathematics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
E-mail: {babarczi, retvari, tapolcai}@tmit.bme.hu, ronyai@sztaki.hu

***Abstract*—Recent trends point towards communication networks will be multi-path in nature to increase failure resilience, support load-balancing and provide alternate paths for congestion avoidance. We argue that the transition from single-path to multi-path routing should be as seamless as possible in order to lower the deployability barrier for network operators. Therefore, in this paper we are focusing on the problem of routing along the shortest pairs of disjoint paths between each source-destination pair over the currently deployed link-state routing architecture. We show that the union of disjoint path-pairs towards a given destination has a special structure, and we propose an efficient tag encoding scheme which requires only one extra forwarding table entry per router per destination. Our numerical evaluations demonstrate that in real-world topologies usually only 4 bit tags are sufficient in the packet headers to route on the disjoint path-pairs. Finally, we show that our tags automatically encode additional paths beyond the shortest pair of disjoint paths, including the shortest paths themselves, which enables incremental deployment of the proposed method.**

***Index Terms*—multi-path routing, Suurballe-Tarjan algorithm, data-plane scalability, incremental deployment**

I. Introduction

Traditional link-state routing protocols calculate a Shortest Path Tree (SPT) towards each destination [1], which provides a single minimum-cost/minimum-delay forwarding path between each source-destination pair. *Single-path routing*, however, tends to concentrate load on certain links of the network and even a single link failure may disrupt the flow of traffic between nodes [2]. In contrast, *multi-path routing* leads to more efficient utilization of network capacity by allowing load to be distributed among multiple forwarding paths, and enables instantaneous failure recovery by letting packets to be deflected off the failed path either by any router adjacent to a failure (e.g., IP fast-reroute [3], [4]) or by the source node (i.e., either the ingress router or end-host [5]–[7]). In fact, by making end-hosts fully responsible for load-balancing, failure recovery and traffic optimization, multi-path routing may allow to eliminate dynamic routing all together, yielding a much simpler static forwarding scheme [8].

Our main focus in this paper is *source-selectable multi-path routing* [9], [10], where the goal is to provide multiple pre-calculated paths towards each destination from which sources can select a single path, or simultaneously several paths, for forwarding traffic. For instance, a source may explicitly encode the hops of the selected path in the packet header (by using source-routing, segment-routing, etc. [6]), it may use a *tag* to mark packets by setting some pre-defined bits in the header and then the network would guarantee that marked packets would be forwarded along the path associated with the tag, or it may leave the selection of the forwarding path entirely to the network (e.g., as in equal-cost multi-path where the "tag" is implicit, e.g., the hash of the IP 5-tuple). Explicit routing provides finer control over forwarding paths, whereas tags limit the source to select only from a *pre-defined path set* without being aware of the actual nodes the forwarding path will traverse, which may help mitigate the security concerns associated with source routing [5].

Source-selectable multi-path routing will inevitably lead to an increased use of scarce network resources. At the very minimum the new paths will need to be encoded in the data-plane, consuming extra entries [9]–[11] in the forwarding tables that are already stressed by single-path routing [12], and pinning packets to specific paths will require additional packet header bits or the introduction of extra shim headers [6], [7], [13]. In the control plane, distributing the required routing information may require excessive routing protocol messaging [14], and calculating the extra forwarding paths will unavoidably cause additional CPU and memory consumption in routers. In this paper, we argue that the transition to multi-path routing should incur as small data-plane and control-plane overhead as possible, in order to remove the deployability barrier for multi-path routing.

As a first step towards an incrementally deployable multi-path routing scheme, *we propose to replace the (single) shortest path used in link-state routing with the shortest pairs of disjoint paths (SPDP)* [15], [16]. We argue that this is a small departure from the current practice, which maintains the principle of least-cost/least-delay routing from shortest-path routing but extends it to two *disjoint* paths per source-destination pair, and hence would entail minimal change in the management of operational networks. At the same time, using two disjoint paths already admits guaranteed fast recovery from any single-link/single-node failure and unlocks traffic optimization in an end-to-end fashion, by enabling sources to dynamically load-balance traffic among their paths in concert with congestion feedback. In fact, the idea of routing on disjoint path-pairs was already proposed (e.g., using independent trees [17], [18]), and specifically SPDP has also been considered in [6], [16]. However, these methods are either not backward compatible with the least-cost/least-delay policy

[17], [18] or impose considerable data-plane and/or control-plane overhead [6], [16] (see Table I).

While efficient algorithms exist to calculate the SPDP routes requiring only the graph of network and link costs as input [15] – which mitigates the concerns related to control-plane overhead –, the data-plane encoding of SPDP routes is by far not trivial. This is because, in contrast to the (single) shortest paths, the shortest *pairs* of disjoint paths do not line up into a single destination-rooted tree. Correspondingly, a naive SPDP encoding would require source-specific routing at each intermediate router, yielding quadratic-size routing tables (one entry per source-destination pair). This would not scale, not even in the context of a single autonomous system. Our main contribution in this paper is *the first demonstration that an efficient data-plane encoding scheme exists for implementing the shortest-pairs of disjoint paths* with just one additional forwarding table entry per destination (on top of the default entry required by shortest path routing) and small tags. What is more, *a trivial extension of our encoding scheme makes further high-quality paths available for sources, including the shortest paths themselves*, and this extension comes with no additional data-plane resource usage. The particular contributions are:

(i) we introduce a data structure to represent SPDPs which admits destination-based multi-path routing with $O(1)$ routing table entries, which provides *optimal data-plane scalability* [11]; and

(ii) we propose a tagging mechanism and give a theoretical lower bound on the required tag size, and we show empirically on real-world topologies that the *packet overhead is negligible*.

Our scheme is fully backward compatible with traditional link-state routing in that it can be deployed only on a subset of routers to obtain partial gains. We believe that these properties make our scheme an efficient *incrementally deployable source-selectable multi-path routing scheme*.

The rest of the paper is organized as follows. In Section II we discuss related work. Section III contains the formulation of the SPDP problem and Section IV presents our main result, an improved data structure for encoding SPDP routing. Encoding is discussed together with a general lower bound on the tag size in Section V. Finally, Section VI presents our simulations results and the paper is concluded in Section VII.

II. Background and Related Work

In legacy intra-domain routing IP packets are forwarded along the single shortest path between each source and destination node in a hop-by-hop scheme, where each router decides the next-hop (out-link) based on the destination address d in the packet header and the information stored in the forwarding table (called *rules*). In traditional link-state routing protocols (e.g., Open Shortest Path First) the network topology is discovered by exchanging control messages between neighboring routers, and Dijkstra's algorithm [1] is used to calculate the shortest paths in the resultant graph. Simplicity notwithstanding, the deficiencies associated with single-path routing have been well-known for quite some time, including the potentially poor utilization of network resources due to the lack of path diversity, proneness to grave traffic disruptions even in the case of a single failure, instability and oscillations, etc. [2]. Below, we summarize the proposals for extending this scheme to multi-path routing using multiple pre-calculated paths; the most important references are highlighted in Table I.

TABLE I
Multi-path routing methods (Rule # is given per d).

Paths	Routing scheme	Disjoint path #	Rule #	Tag size ($\lceil$bits$\rceil$)	Stretch
Single	SPT [1]	1	1	-	Opt
Double	2-SP [19]	1	$\leq n$	$1+\log n$	Any
	RDP [6]	2	-	$O(n \log n)$	Any
	RB-trees [17]	2	2	1	Any
	SPDP naive (Fig. 1b)	2	$\leq n$	$1+\log n$	Opt
	SPDP w/ rewrite [16]	2	2	$\log n$	Opt
	SPDP+SPT w/ TAG	**2**	**2**	$\approx \mathbf{0.5} \log \mathbf{n}$	**Opt**
Multi	Path Splicing [10]	1	k	$D \log k$	Any
	Deflections [9]	1	$\leq \delta$	16	Any
	SPDP+MP w/ segment	**2**	-	$O(\mathbf{D} \log \mathbf{n})$	**Any**
	SPDP+MP w/ TAG	**2**	**2**	$\approx \mathbf{0.5} \log \mathbf{n}$	**Any**

A. Routing on Disjoint Path-Pairs

As an early attempt to provide alternative low-delay paths beyond the shortest path tree (SPT), k-shortest path algorithms (*2-SP* in Table I) were proposed [19]. Unfortunately, all the k paths might share a single link in certain cases and hence k-shortest path routing is still prone to single failures. The easiest way to eliminate single points of failure is to provide *two disjoint* paths between each source and destination node [20]. Accordingly, *red-blue trees* (RB-trees) [17], [18] are two independent trees rooted at each destination d so that the (unique) path from any node v to d in one of the trees is disjoint from the path in the other tree. Encoding such the RB-trees requires two forwarding table entries per destination (one next-hop along the red and another one along the blue tree) and packets can be tagged to any of the trees using a single header bit. Unfortunately, the stretch (the path-length-dilation compared to the theoretical minimum, i.e., the SPDP) can be arbitrarily large. In response, [17] asked for the shortest pair of RB-trees; unfortunately stretch may still be unlimited and with the introduction of this new objective the computational complexity becomes NP-complete (from a trivial linear time as in [18]). Finally, in [6] the goal is to minimize the length of the longer path of the path-pair, so that the paths are robustly disjoint (RDP); i.e., they remain disjoint even after a link fails. Unfortunately, this problem is NP-complete [21], which would skyrocket control plane overhead by requiring routers to optimize intractable problems for path calculation.

In contrast to these proposals, routing along the shortest pair of disjoint paths (SPDP) provides *theoretically optimal* stretch and admits a *polynomial-time* implementation – if the total length of the paths is minimized – using the *Suurballe-Tarjan algorithm* [15]. Unfortunately, a naive data-plane SPDP encoding would result in quadratic size (per-source-destination-pair) forwarding tables. Although [16] proposes a data-plane

SPDP encoding with only two forwarding table entries and a single node identifier in the packet headers, it does not admit a trivial implementation in current link-state routing because it neither automatically delivers the segments to be used in segment routing nor it is compatible with tagging (because of identifier rewrites in the packer header at certain hops, referred to as *SPDP w/ rewrite* in Table I).

B. Multi-Path Routing Approaches

In *source-selectable routing* [9], [10] the network provides multiple (not necessary disjoint) pre-calculated paths to source nodes, which can select among them by marking packets with different tags. As the source has no explicit knowledge of the paths tags describe, no topology information is revealed to the sources (mitigating security concerns) and the routing can be changed by the operator any time without the sources being aware. In [9] deflection routing is proposed, which defines a set of alternate next-hops at each router so that the resultant (multiple) paths are loop-free. A fixed sized small (16 bit) tag is added to the packet headers, and a pseudo-random operation is used to map the tag to a unique next-hop at each router. Path splicing [10] improves path stretch and lowers the number of routing table entries compared to deflection routing by introducing k routing trees. The first tree is the SPT for backward compatibility, and each subsequent tree is calculated as an SPT in the same topology after a degree-based random perturbation of the link weights. The source explicitly selects the path by arbitrarily combining next-hops along the trees.

C. Our Routing Methods

As no efficient and scalable implementation exists for SPDP [6], [16], we introduce a data structure by improving the recursive construction proposed in [16]. We define SPDP paths as a sequence of *SPDP-segments* in Sec. IV, and propose in Sec. V a tag encoding that does not mandate packet rewrite at intermediate nodes (*SPDP+SPT w/* `TAG`). We demonstrate that our scheme inherently encodes the SPT, and also provides additional low-stretch paths for 20%–40% of source nodes (*SPDP+MP w/* `TAG` in Table I). This is a significant improvement over prior work [6], [16], [17], [19], restricted to only two paths. Furthermore, in contrast to previous source-selectable routing methods [9], [10], our paths provide 100% resilience against single failures with only two forwarding table entries. We also discuss a possible segment routing implementation (*SPDP+MP w/ segment*) and its limitations in Sec. VI-C.

III. Problem Formulation

The input of the destination-based hop-by-hop routing problem is the network topology $G = (V, E)$ with $n = |V|$ nodes (routers) connected with $m = |E|$ bi-directional links. Links in network topology G have an administrative cost $\forall e \in E : c(e)$, corresponding to e.g., physical length or delay. We define a path P as a set of directed links connecting source node $s \in V, s \neq d$ and destination node d. We refer to minimum cost paths and path-pairs as *shortest*. As a disjoint set of entries is maintained in the forwarding tables for each destination node $d \in V$, the routing problem can be decomposed per destination. Hence, in the rest of the paper we fix d and provide our results for these individual sub-problems.

Problem 1: Routing on Shortest Pairs of Disjoint Paths

Input : Network topology $G = (V, E)$
Destination node d
Set of SPDP paths $\mathcal{P}$ towards d

Output: Match rule and high prior. action $p_s, \forall s \in V$
Default action $t_s, \forall s \in V$
Tag values $\texttt{TAG}(P)$ addressing $\forall P \in \mathcal{P}$

A. Destination-Based Routing with Tags

The set of all paths from s towards d in the routing problem is denoted as $\mathcal{P}_s$, while $\mathcal{P} = \bigcup_{\forall s \in V} \mathcal{P}_s$. To distinguish the paths in $\mathcal{P}$, h bits in the packet header, denoted as `TAG` and represented as an integer in $[0, 2^h - 1]$, is used and optionally consulted at intermediate routers. Since destination d is always part of the decision from now on we assume forwarding tables match only on the `TAG`, but we explicitly allow forwarding table entries to define wildcard rules on the tags.

Definition 1: The h bit long binary vector $\texttt{TAG} \in \{0,1\}^h$ **matches** the wildcard rule w if $w[i] \stackrel{?}{=} *$ or $w[i] \stackrel{?}{=} \texttt{TAG}[i]$ for $i = 1 \dots h$, where operator $[i]$ denotes the i^{th} bit of a string.

The routing table contains an ordered list of rules (entries), where each rule is composed of a *match* string w (consisting of $0, 1$ and wildcard characters $*$) and an *action*. For a given `TAG` and node d, intermediate routers find the first (highest priority) matching rule and the corresponding action is executed, i.e., the packet will be forwarded along the corresponding out-link. Let $\texttt{TAG}(P)$ denote the tag assigned to path $P \in \mathcal{P}$.

Definition 2: We say that a path $P \in \mathcal{P}_s$ from a node s to destination d is **addressed** with $\texttt{TAG}(P)$, if packets with header $\texttt{TAG}(P)$ injected at node s into network will be forwarded exactly along the links of path P; formally $\forall e = (u \to v) \in P$ the action of the first matching rule at node u is to forward the packet on out-link $u \to v$.

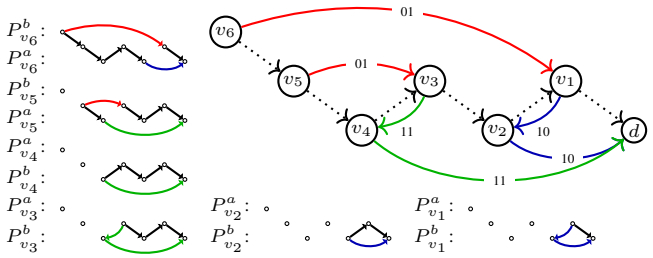
Note that, $P_1, P_2 \in \mathcal{P}$ might be addressed with the same tag $\texttt{TAG}(P_1) = \texttt{TAG}(P_2)$ if they start at different source. The goal is to address every $P \in \mathcal{P}$ path with an appropriate $\texttt{TAG}(P)$, while the tag size h and the number of forwarding table entries $|\mathcal{R}| = \sum_{\forall s \in V \setminus \{d\}} |\mathcal{R}_s|$ are minimal, where $|\mathcal{R}_s|$ denotes the number of entries at node s.

Definition 3: Given network topology G, destination node d and set of paths $\mathcal{P}$, the efficiency of a routing can be measured with the following performance metrics:

- **average routing table entries** $\rho := \frac{\sum_{\forall s \in V \setminus \{d\}} |\mathcal{R}_s|}{n-1}$,
- **tag size** h, where $0 \leq \texttt{TAG}(P) < 2^h$ for $\forall P \in \mathcal{P}$.

B. Routing on the Shortest Pairs of Disjoint Paths

Our goal is to propose an efficient routing for link-disjoint SPDP paths (can be easily extended to node-disjoint paths [15]), given in Problem 1. It was shown in [15], that from



(a) Network topology and SPDP paths in $\mathcal{P}_s$ towards d. The SPT is shown with black dotted links, the secondary next hops are shown with solid colored lines. Values a and b identifies the path within the disjoint path-pair, while the tag values at secondary links correspond to Fig. 1c. $c(v_4, d) = c(v_6, v_1) = 6, c(v_5, v_3) = c(v_2, d) = 3$, all other link costs are 1.

Match at v_4		Act.
src	TAG	
v_6	a	v_3
v_5	a	d
v_4	a	v_3
v_4	b	d
v_3	b	d

Match at v_3		Act.
src	TAG	
v_6	a	v_2
v_5	b	v_2
v_4	a	v_2
v_3	a	v_2
v_3	b	v_4

Match at v_2		Act.
src	TAG	
v_6	a	d
v_5	b	v_1
v_4	a	v_1
v_3	a	v_1
v_2	a	v_1
v_2	b	d
v_1	b	d

(b) Naive encoding of $\mathcal{P}_s$ requires tag and source address matching, and needs one entry per traversing path ($\rho = 29/6$, $h = 1 + \lceil \log n \rceil$).

Match at v_4	
TAG	Act.
11	d
**	v_3

Match at v_3	
TAG	Act.
11	v_4
**	v_2

Match at v_2	
TAG	Act.
10	d
**	v_1

(c) Proposed encoding of $\mathcal{P}_s$ requires tag matching on the secondary next-hop and one default entry for the SPT next-hop per destination ($\rho = 2$, $h = 2$).

Fig. 1. Example where a naive SPDP path encoding gives $O(n)$ entries, while the proposed tag encoding based on the SPDP data structure has only 1 extra entry per destination.

an arbitrary source s one path from the SPDP starts on the out-link t_s of the shortest path (denoted as P^a_s), while the other path (P^b_s) use a different out-link p_s. A directed link is called *t-link* (or *p-link*) if it is t_s (or p_s) for any node s towards d. As a direct consequence, the union of links of the SPDP paths in $\mathcal{P}$, i.e., $\bigcup_{\forall s \in V} P^a_s \cup P^b_s$ contains all t_s links of the SPT; thus, $\mathcal{P}$ implicitly always contains all shortest paths P^*_s towards d besides P^a_s and P^b_s. An efficient routing for Problem 1 is shown in Fig. 1 for destination d with $\rho = 2$, $h = 2$. Note that each forwarding table has two entries, i.e., a high-priority rule w which matches on the TAG value and forwards packets on the secondary next-hop p_s, and a default entry which forwards packets along the shortest path link t_s otherwise. The shortest pairs of disjoint paths $\mathcal{P}_s = \{P^a_s, P^b_s\}, \forall s \in V, s \neq d$ can be addressed with $h = 2$ bit tags at each source. For example, adding TAG $= 11$ to the packet header at v_4 addresses $P^b_{v_4} = v_4 \to d$, while packets with TAG $= 01$ will be forwarded along $P^a_{v_4} = v_4 \to v_3 \to v_2 \to v_1 \to d$ as they only match the default rule $w = **$ at each hop.

IV. Structure of the SPDP Paths

In this section we introduce a data structure for the SPDP paths that enables an efficient implementation of Problem 1; thus, it makes our approach a candidate as an incrementally deployable source-selectable multi-path routing scheme.

Theorem 1: Let $\mathcal{P}$ consist of all the shortest pairs of disjoint paths P^a_s, P^b_s (and the unique shortest paths P^*_s) from every source $\forall s \in V \setminus \{d\}$. There exists an efficient destination-based hop-by-hop routing with $\rho = 2$ forwarding table entries per destination d.

Proof: In a nutshell, we will show that both SPDP paths P^a_s and P^b_s consist of t-links and p-links only $\forall s \in V \setminus \{d\}$, which results in $\rho = 2$.

Assume the SPDP is unique for every source, which can be achieved e.g., by adding a different random epsilon cost to all links. The result of the ST algorithm [15] is a data structure that must be traversed in two phases to construct both paths P^a_s and P^b_s. The data structure stores two out-links for each node $v \neq d$ (t_v and p_v), and a pointer. Because the paths are built up only using these two (t_v and p_v) links at each node v, each router has two out-links which can be encoded in two entries $|\mathcal{R}_s| = 2$ using a bit map (i.e., each bit corresponds to a path in $\mathcal{P}$ and the rule w has $*$ in the position of the paths for out-link p_v), from which $\rho = 2$ follows. ■

Although an important theoretical result, the ST data structure can not be used to efficiently describe the SPDP paths in routing. Therefore, Ogier-Rutenburg-Shacham [16] proposed a distributed data structure and a recursive traversal algorithm to obtain the SPDP paths P^a_s and P^b_s at source s in a single phase. Instead of using the complex pointers and involved conditions in [16], in this paper we propose a simple traversal that only stores two path segments for each node v, called **SPDP-segments**, see Fig. 2. The first segment of the two SPDP paths at v start on the p_v and t_v link denoted as S_v and S'_v, respectively, and traverses (zero or more) t-links until a p-link (start of another segment) or destination d is reached. Hence, S_v consists of the link p_v and consecutive t-links (zero or more) along the P^b_v path, while the first segment S'_v of P^a_v contains only t-links (at least t_v) until the first p-link or destination d (start of another segment). Once we have S_v and S'_v defined for every node $v \neq d$, based on the previous constructions [15], [16] we can obtain the two paths for an arbitrary source s as follows:

- initialize $P^a_s := S'_s; i := v'_s$, where v'_s is the last node of segment $S'_s = s \to \ldots \to v'_s$. P^a_s is constructed by appending $S_i = i \to \ldots \to v_i$ SPDP-segments together:

$$P^a_s := \langle P^a_s \to S_i \rangle; i := v_i \ , \text{ until } i = d,$$

- initialize $P^b_s := S_s; i := v_s$, where v_s is the last node of segment $S_s = s \to \ldots \to v_s$. P^b_s is constructed by appending $S_i = i \to \ldots \to v_i$ SPDP-segments together:

$$P^b_s := \langle P^b_s \to S_i \rangle; i := v_i \ , \text{ until } i = d,$$

where $\langle P_s \to S_i \rangle$ means SPDP-segment S_i is added to the path, and node i is replaced by the last node of S_i. Note that, P^a_s will traverse S'_s first and optionally some other S_v segments, while path P^b_s first traverses S_s and optionally some other S_v segments. For example, in Fig. 2 $P^a_{a_9} = S'_{a_9} \to$

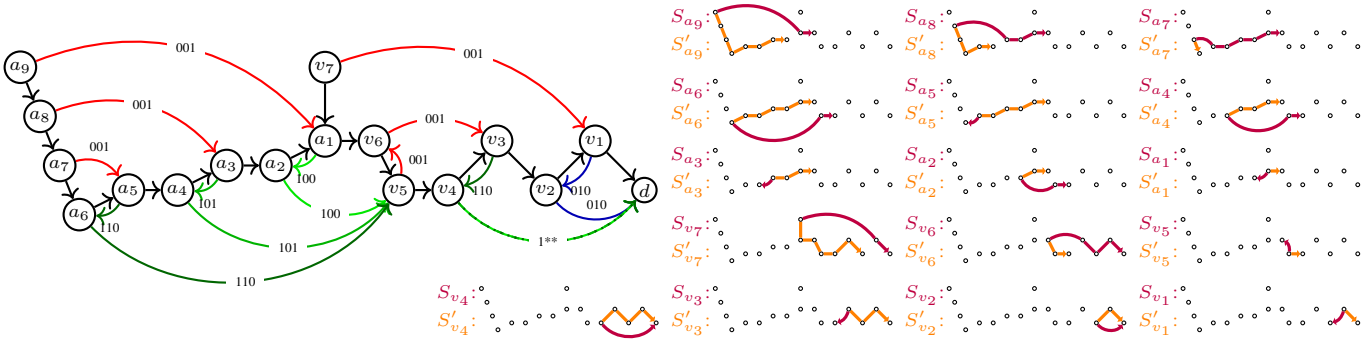


Fig. 2. Topology where multiple TAG values should be matched by wildcard rule w for out-link p_{v_4}. Black links are t_s links on the SPT and have unit cost. p_s links are colored according to their TAG values. The ones reverse to the SPT have unit cost, the other p_s links have cost one plus their SPT distance, e.g., $c(v_7, v_1) = 8$, except $c(v_4, d) = 6$, $c(a_6, v_5) = 10$, and $c(a_4, v_5) = 7$. In the small figures the SPDP-segments S_v and S'_v from each node are drawn with purple and orange, respectively.

$S_{a_2} \rightarrow S_{v_4}$, while $P^b_{a_9} = S_{a_9} \rightarrow S_{v_6}$. This data structure gives us an important structural property of the SPDP paths:

Observation 1: If either P^a_s or P^b_s from source s towards destination d traverses link p_v, it traverses the whole path P^b_v.

In other words, *once a packet is forwarded along SPDP-segment S_v, it will follow the same path (same series of SPDP-segments) towards d,* ***regardless of the source of the packet***. Note that, in Fig. 2 SPDP paths $P^b_{a_9}, P^b_{a_8}, P^b_{a_7}$ and $P^b_{v_6}$ all traverse SPDP-segment S_{v_6}, and after using the corresponding p-link (v_6, v_3) they will follow the same $P^b_{v_6}$ path until destination d is reached. This crucial property provides us the opportunity to propose efficient implementation for routing on the shortest pairs of disjoint paths. We propose a tag based encoding in Section V and discuss a possible segment routing implementation in Section VI-C. However, our result is general and can fit other routing architectures as well.

V. Tag Size Bound and Forwarding Rule Encoding

In this section we propose a tagging mechanism for the SPDP paths. In Section V-A we provide a general lower bound on h for an arbitrary set of paths $\mathcal{P}$. Section V-B shows the difficulties of tag assignment and wildcard rule construction to meet the $\rho = 2$ requirement of Problem 1. Finally, we discuss the additional path diversity for *SPDP+MP w/* TAG in Section V-C, which comes as a side effect of our rule encoding and the SPDP data structure in Section IV.

A. Lower Bound on the Tag Size for General Path Sets

First, we define when two paths in an arbitrary path set $\mathcal{P}$ have to be addressed with different $\texttt{TAG}(P)$ values.

Definition 4: Two paths $P_1, P_2 \in \mathcal{P}$ are in **conflict** at node v if in the union of links $P_1 \cup P_2$ node v has out-degree 2.

When P_1 and P_2 use different out-links at v (i.e., they are in conflict), they need to be distinguished from each other with different $\texttt{TAG}(P_1) \neq \texttt{TAG}(P_2)$ tags. The conflicts in $\mathcal{P}$ are modeled in an undirected auxiliary graph G', called the *path conflict graph*, constructed as follows. The path conflict graph G' has $|\mathcal{P}|$ nodes, where each node corresponds to a unique path in $\mathcal{P}$. We connect two nodes in G' with an undirected edge if the corresponding paths P_1 and P_2 are in conflict at any node v in G. Recall that the *chromatic number* $\chi(G')$ of a graph G' is the minimal number of colors required to color its nodes, such that any two adjacent nodes have distinct colors [22]. By the above construction, there is a full correspondence between a valid node coloring in G' and a valid $\texttt{TAG}(P)$ value assignment which appropriately addresses all paths in $\mathcal{P}$, that immediately allows us to express the minimal tag size h in terms of the chromatic number:

Theorem 2: The tag size to address all paths in $\mathcal{P}$ is at least $h \geq \lceil \log \chi(G') \rceil$, where $\chi(G')$ is the chromatic number of the path conflict graph G'.

Proof: Let us color G' with $\chi(G')$ different colors. As each node in G' corresponds to a path in $\mathcal{P}$, we may treat it as a coloring of paths, too. Recall that each pair of paths in conflict corresponds to two nodes in G' which are connected by an edge. Hence, any two paths in conflict are assigned different colors. Finally, for each color we assign a unique tag, which ensures that two paths towards d have different $\texttt{TAG}(P)$ if they use a different out-link at any v. Match rules w in the forwarding tables can be assigned accordingly on $\lceil \log \chi(G') \rceil$ bits. We also need to show that h cannot be smaller. We prove it indirectly. Assume that there is a solution with a smaller tag size, then map these tags into colors and assign them to the nodes of G'. In this case, we have the valid coloring of G', because if two nodes are connected in G' the corresponding paths in $\mathcal{P}$ have two distinct out-links at a given node, which requires different tags. However, the number of colors is less than $\chi(G')$, which is a contradiction. ■

Note that, the above bound is tight for $|\mathcal{R}| = \sum_{P \in \mathcal{P}} |P|$ forwarding table entries. Unfortunately, we have no hope to find a polynomial-time algorithm for minimizing h in general, because the graph node coloring problem is NP-hard [23].

B. Encoding Tags into Wildcard Forwarding Rules with $\rho = 2$

In this section, we focus on the problem of assigning $h \geq \lceil \log \chi_g(G') \rceil$ bit long tags to the $\chi_g(G')$ colors obtained from the conflict graph coloring with e.g., a greedy algorithm in order to address all SPDP paths in $\mathcal{P}$ with one single

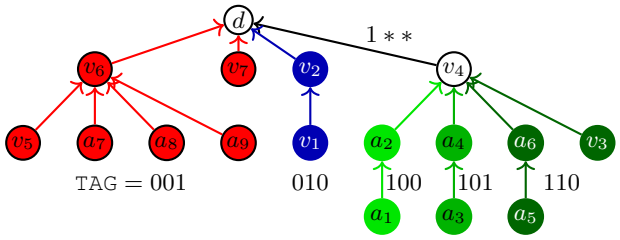


Fig. 3. SPDP-segment hierarchy (T^S) for Fig. 2. The color of the nodes is computed according to the conflict graph.

(wildcard) rule w for p_s at each router besides the default shortest path entry t_s. Fig. 2 shows an example, where encoding of the colors to appropriate TAG values is not trivial. Note that, paths $P^a_{a_7} = a_7 \to a_6 \to v_5 \to v_4 \to d$, $P^a_{a_8} = a_8 \to a_7 \to a_6 \to a_5 \to a_4 \to v_5 \to v_4 \to d$ and $P^a_{a_9} = a_9 \to a_8 \to a_7 \to a_6 \to a_5 \to a_4 \to a_3 \to a_2 \to v_5 \to v_4 \to d$ are in conflict with each other because of node a_6 ($P^a_{a_7}$ with $P^a_{a_8}, P^a_{a_9}$) and a_4 ($P^a_{a_8}$ with $P^a_{a_9}$). In other words $P^a_{a_7}$, $P^a_{a_8}$ and $P^a_{a_9}$ must be assigned a different TAG, but their binary TAG value should be matched with a single wildcard rule w at node v_4. Therefore, it is not enough to ensure that different colors are assigned to different TAG values, but also that certain TAG values can be matched by wildcard rule w while others are not. For example, tags TAG $= 000$ and TAG $= 111$ cannot be matched by $w = ***$ without matching all other tag values. These additional constraints will increase tag size h if we fix $\rho = 2$.

Next we give a heuristic approach to assign TAG values for each color. According to Observation 1, if P^b_v and P^b_u both traverse the same p_x link, then the two paths will traverse the same SPDP-segments from node x till destination d. It means we can define a *segment hierarchy tree* $T^S = (V_T, E_T)$ among the SPDP-segments S_v, where $V_T = V$, and there is a directed link $(v \to x) \in E_T$, when x is the end node of segment S_v, see Fig. 3. Hence, T^S is a rooted tree at d, where the links of the unique path from v to d in tree T^S correspond to the S_x segments of P^b_v. Wildcard rules required at each node (router), which has child nodes of multiple colors (e.g., v_4 in Fig. 3). We assign the colors to TAG values (binary vectors) incrementally $1, 2, \ldots, \chi_g(G')$ in the order of a pre-order traversal of T^S. The traversal in Fig. 3 is v_6, v_5, a_7, a_8, a_9, v_7, v_2, v_1, v_4, a_2, a_1, a_4, a_3, a_6, a_5 and v_3.

Let $\chi(v)$ denote the number of different colors the child nodes of v has in T^S. In an ideal case, $\chi(v)$ is a power of 2 (say 2^i), and the first unused tag j is divisible by 2^i. Then we can assign binary TAG values $j, \ldots, j + 2^i - 1$ which share the same prefix. However, in practice we might not be able to assign all TAG values. For example, in Fig. 3 we start with TAG $= 1$, and $\chi(v_6) = 1 = 2^0$, thus we can assign the TAG for all of its children, and also to v_7. Next at v_2 we have $\chi(v_2) = 1$ and we see a new color, thus, its children are assigned to TAG $= 2$. The situation is different for v_4, because $\chi(v_4) = 3$ (i.e., branch $a_1 \to a_2$; branch $a_3 \to a_4$; and branches $a_5 \to a_6$ and v_3). To aggregate 3 tags, we need 2 bits. Note that, the first unused value TAG $= 3$ is not divisible by $4 = 2^2$. In this case, there will be some TAG values that cannot be assigned to any color (they will be 011 and 111). The TAG values we can use are 4, 5 and 6 that is 100, 101 and 110 in binary representation and the (prefix) rule is $w = 1**$ for action p_{v_4}. The above process is repeated until all colors are assigned with a binary tag. Surprisingly, in real-world networks the segment hierarchy tree is small, and we measured $h \leq 4$ bits in all investigated topologies in Section VI.

C. Inherent Path Diversity of the SPDP Data Structure

Our main objective is to propose an efficient source-selectable routing scheme along the SPDP paths in $\mathcal{P}$. However, in order to make it a fully fledged multi-path routing method, further path-diversity is required, which is luckily enough inherently offered by the SPDP-segments.

We already discussed that P^*_s is always included as a path on the links in $\mathcal{P}$ (i.e., $\bigcup_{\forall s \in V} P^a_s \cup P^b_s$) for every source s (either as P^a_s or as a third path). For incremental deployment and backward compatibility, we always require that the SPT paths can be addressed with TAG $= 0$ value for all $s - d$ pairs. Thus, in the example in Fig. 1 TAG $= 00$ addresses $P^*_{v_4}$, which is the same as $P^a_{v_4}$ in this case. Moreover, v_4 might find multiple paths on the links of $\bigcup_{\forall s \in V} P^a_s \cup P^b_s$ by further exploring the available tag space, e.g., by sending packets with the fourth tag value TAG $= 10$ it can address $P_{v_4} = v_4 \to v_3 \to v_2 \to d$, which is neither the SPT nor one of the SPDP paths. Note that, using a TAG value other than the SPT TAG $= 0$ or the ones addressing the SPDP paths, source s replaces the S'_s SPDP-segment with another $S''_s \subset P^*_s$ shortest path segment on t-links until a node v, and use out-link p_v and the corresponding P^b_v path afterwards. Therefore, the total number of paths available in source-selectable routing for a source s towards d through exploring the whole tag space (referred to as TAG paths in Section VI) depends both on its depth in the SPT, and on the tag assignment and wildcard rule encoding.

VI. Simulation Results

We conducted thorough simulations on 2-connected small and medium size real-world and large synthetic topologies to demonstrate the efficiency of our approach (properties are summarized in Table II). In real-world Internet Topology Zoo [24] and SNDLib [25] networks we used the physical distances of the nodes as the administrative cost $c(e)$ on the links measured in km, while for the large synthetic planar topologies [26] we applied unit link costs ($\forall e \in E : c(e) = 1$). In Section VI-A we analyse the efficiency of the proposed data structure and tag encoding of the SPDP paths. In Section VI-B we compare our method to path splicing [10], while Section VI-C reveals the difficulties of a possible segment routing implementation of the SPDP paths. Simulations were performed on a machine running Debian Linux version 4.9, with four 2.59 GHz Intel processors and with 8 GB RAM.

A. Tag Assignment and Rule Encoding with Greedy Coloring

We present our simulation results for all investigated topologies in Table II, averaged for all destination d. First, we

TABLE II

QUALITY OF SOURCE-SELECTABLE TAG PATHS, NUMBER OF COLORS AND h TAG SIZE, AND WILDCARD RULE ENCODING ON REAL-WORLD (UPPER PART: INTERNET TOPOLOGY ZOO [24], MIDDLE PART: SNDLIB [25]), AND SYNTHETIC PLANAR TOPOLOGIES [26] (LOWER PART).

Topology Name	$\|V\|$	$\|E\|$	Avg SPT Depth (D hops)	Avg Path Number Per $s-d$	Max Path Number Per $s-d$	Avg Path Stretch [km] Per $s-d$	Avg Colors Per d	Max Colors & Tag Size (h bits)	Max Colors in Wildcard	Avg Rules w/ Wildcard Per d	Running Time [s] (All $s-d$)
Abilene	11	14	2.51	2.07	3	1.72	3.00	3 (2 bits)	1	0.00	0.03
BtEurope	17	30	1.99	2.10	3	3.93	3.06	4 (2 bits)	1	0.00	0.09
BtNorthAmerica	33	70	3.09	2.49	5	3.00	4.12	5 (3 bits)	2	0.79	0.97
Dfn	51	80	3.70	2.77	5	1.68	4.80	6 (3 bits)	3	2.02	10.30
Polska	12	18	2.17	2.17	4	1.49	3.75	5 (3 bits)	2	0.08	0.36
India	35	80	3.29	2.60	5	1.21	4.57	6 (3 bits)	3	0.94	1.30
Janos	39	61	4.39	2.63	6	1.34	4.82	6 (3 bits)	3	1.51	2.25
Germany	50	88	4.46	2.80	7	1.30	5.34	7 (3 bits)	3	2.68	5.52
n100e287	100	287	4.14	3.04	7	1.15	5.80	8 (3 bits)	4	3.81	46.70
n100e145	100	145	5.72	3.18	7	1.34	6.13	7 (3 bits)	4	6.38	68.20
n200e575	200	575	5.95	3.64	9	1.12	7.25	9 (4 bits)	5	11.16	869.00
n200e299	200	299	8.42	3.77	9	1.25	7.51	11 (4 bits)	5	14.63	1650.00

measured the average depth D of source nodes s in the SPTs in terms of hop count, which bounds the maximum available TAG paths (addressable S''_s segments). As expected, in topologies where source nodes are further away from d the maximum path number available through exploring the TAG space is up to 9 for certain $s-d$ pairs in *SPDP+MP w/* TAG. The average stretch of all TAG paths is below 1.5 for most topologies (compared to the shortest path), except for Internet Topology Zoo [24], where there is a three orders of magnitude difference (in km) between some link costs $c(e)$, resulting extremely long detour paths for some $s-d$ pairs.

The average and maximum number of different tag values (i.e., colors in the path conflict graph G') and the corresponding tag size h are also shown in Table II. For coloring the nodes of the path conflict graph G' we implemented a greedy algorithm, where in each step the node with the most uncolored neighbour is colored with the lowest available color value. Note that, we tried multiple heuristics, but the node coloring order provided by this greedy algorithm significantly outperformed the others. As a result, we observed that 11 colors, thus, $h = 4$ bit tags are sufficient even in larger networks, which is $\approx 0.5 \log n$ for all investigated topologies.

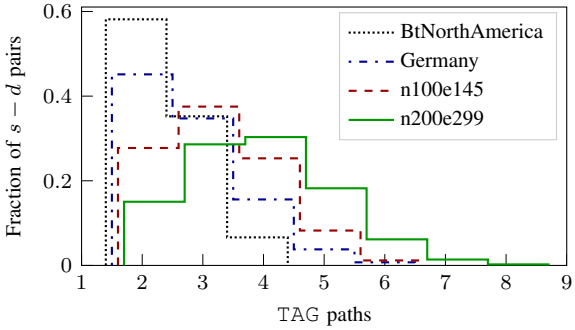
We analysed the wildcard rule encoding in Table II as well and measured the number of p_s links where exact TAG matching is not sufficient, i.e., multiple TAG values should be matched by w (discussed in Fig. 2). One can observe, that in the Germany network (and other real world topologies) at most three TAG values should be matched by a single wildcard rule. On average (per d) 2.68 forwarding table entries require wildcard match in this network; in other words, 95% of the rules in the routers are an exact match for action p_s.

In order to shed more light on the number of source-selectable paths we performed further simulations in Fig. 4a. One can observe that 40% of $s-d$ pairs has at least three paths in the BTNorthAmerica topology (goes up to 85% for n200e299). Recall that P_s^a, P_s^b are always among these paths, moreover $\texttt{TAG}(P_s^*) = 0$ addresses the shortest path. Furthermore, there are even more TAG paths that can be addressed by s, e.g., for the Germany topology about 20% of nodes have access to $4-7$ paths. Fig. 4b demonstrates tag space density, i.e., how many unique TAG paths exists compared to the tag space 2^h. High density means that even when the source is not aware of the TAG values corresponding to paths, it will frequently find a new path by sending packets with a different random TAG. Our results show that every fifth try will result in a new TAG path in worst case, but for more than 50% of s nodes less than two tries is sufficient on average, which is important for a prompt reaction.

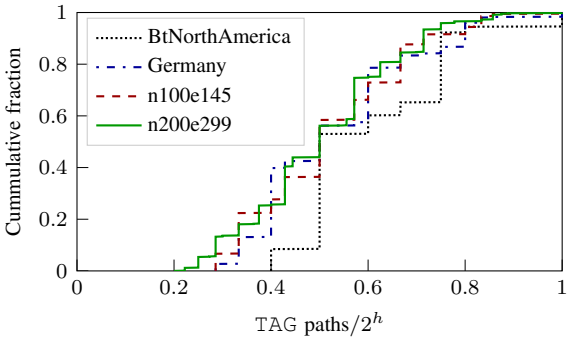
B. Path Splicing Trees versus SPDP Data Structure

We compare our tag assignment and rule encoding scheme proposed for Problem 1 to another source-selectable routing approach, *path splicing* [10]. We selected it because it provides higher path diversity, lower path stretch and less routing table entries than deflection routing [9]. In our path splicing implementation we used the degree-based perturbation of link costs to get $c'(e)$ according to Eq. (1) in [10] in the range of $[0, 20]$ in order to construct the k shortest path trees on the perturbed $c'(e)$ link costs. In Fig. 5 we allowed $D = 20$ hops similarly to [10], which is larger than the maximum depth of the SPT in the investigated networks. This results in tag size of 20 header bits for $k = 2$ and 60 bits for $k = 5$, compared to the 4 bits of our tag encoding. For each $s-d$ pair we made 16 random tries to find diverse paths (without any guarantees) in path splicing, while we explored the whole 4 bit tag space (guaranteed SPT and SPDP paths) in our tag assignment. We present the results only for the n200e299 topology where path splicing had the best performance.

One can observe in Fig. 5a that our method guarantees at least two paths for 100% of $s-d$ pairs (i.e., the disjoint path-pair), while with $k = 2$ path splicing using the same amount of forwarding table entries about 30% of s nodes only have access to the shortest path, i.e., an arbitrary single link failure along the path will disrupt the connection. Although path splicing with $k = 5$ outperforms *SPDP+MP w/* TAG in terms of the maximum number of TAG paths (still 10% of s without second path), it requires five forwarding table entries per d compared to two of the other two approaches; thus, if

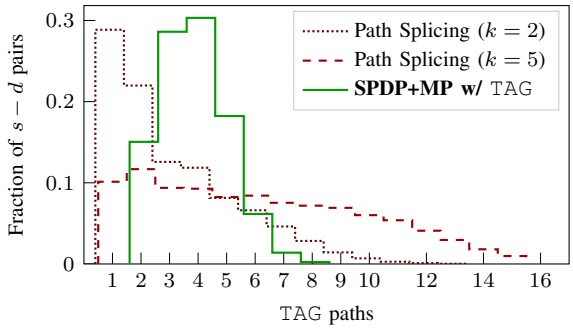


(a) Fraction of $s-d$ pairs versus the unique TAG paths per source s.

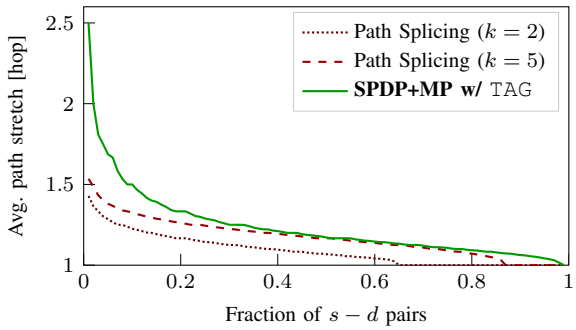


(b) Cumulative distribution function of tag space density. $|P|/2^h = 0.5$ means that s finds a new path for every second try on average.

Fig. 4. The number of TAG paths and the tag space density in *SPDP+MP w/* TAG (h values per topology are in Table II).



(a) Fraction of $s-d$ pairs versus the unique TAG paths per source s after testing 16 different TAG values.



(b) Fraction of $s-d$ pairs with at least y path stretch in hop count compared to the shortest path, averaged for all TAG paths at s.

Fig. 5. Comparison of our approach with path splicing with two next hops per destination ($k=2$), and with $k=5$ trees proposed in [10] on the n200e299 network with unit link costs.

the data plane resources are scarce, it is not a viable approach. In Fig. 5b we also demonstrated that for 80% of $s-d$ pairs our TAG paths produced similar average path stretch per s as path splicing with $k=5$ (between $1-1.3$), while splicing $k=2$ has the lowest stretch owing to its poor path diversity. The worst-case stretch of SPDP grows to 2.5, while the splicing methods have 1.5, i.e., this is the price we pay to have the disjoint path-pairs in $\mathcal{P}$ which provide 100% single link failure resilience in our *SPDP+MP w/* TAG implementation.

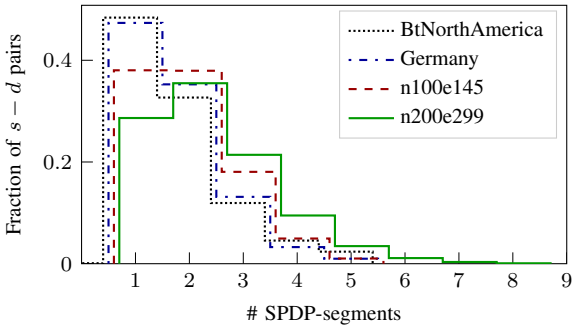
C. SPDP Data Structure with Segment Routing

Finally, we investigate an alternative implementation of the introduced SPDP data structure using *segment routing (SR)* instead of tags. In order to do so, we need to translate SPDP-segments into SR labels to address the corresponding TAG paths. Note that, in [6] the authors measured that addressing SPDP paths requires up to 26 SR labels in worst case and around 10 SR labels for several networks. As real routers support only a maximum stack of three SR labels [27], SPDP paths did not have an efficient implementation. Note that, with our proposed data structure each S_v segment can be encoded with at most two SR labels, i.e., the p-link with an adjacency segment and t-links (if any) with a shortest path segment.

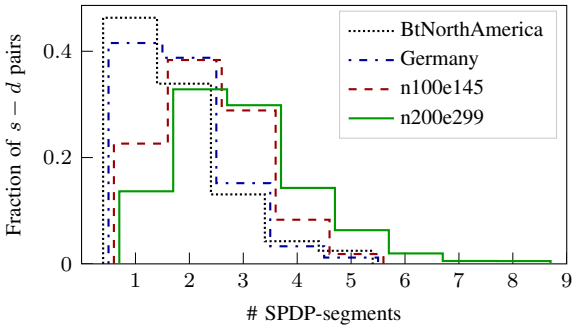
In Fig. 6a we analyze the number of S_v SPDP-segments in the P_s^b paths. Our results demonstrate that for 70-80% of $s-d$ pairs the maximum number of SPDP-segments is 2. Hence, these paths can be encoded in segment routing with at most 4 SR labels (two adjacency segments and $0-2$ shortest path segments), yielding an efficient implementation for the SPDP paths. However, in larger topologies about 10% of $s-d$ pairs requires more than 10 SR labels, which is beyond the capabilities of current routers [6], [27]. If we investigate all TAG paths in addition to P_s^b (which are the combination of S_s', S_s'' and P_v^b paths), the number of $s-d$ pairs with short SR label stack follows a similar tendency for small networks, while drops below 50% for n200e299, shown in Fig. 6b. Furthermore, in segment routing our S_v and S_v' segments can be arbitrary combined by the source s, resulting in a huge path diversity for the price of increased path stretch and an even deeper SR label stack. Although our data structure lowered the number of SR labels and offers an efficient routing for several $s-d$ pairs, our simulations suggest that a general SR implementation is not viable yet for SPDP. However, if routers will support larger SR label stacks in the future, segment routing will be a promising alternative for our tag encoding.

VII. Conclusions

With the recent increase in delay-critical traffic on the Internet, source-initiated instantaneous failure recovery and load balancing have become a must, instead of a nice-to-have feature. Currently, multi-path routing approaches fulfill this imminent requirement by pre-calculating multiple paths



(a) Maximum number of S_v SPDP-segments in P_s^b per $s-d$.



(b) Maximum number of S_v SPDP-segments in TAG paths per $s-d$.

Fig. 6. Number of S_v SPDP-segments in the paths, which equals to the p-links in the SPDP paths and gives a lower bound on the SR label stack size in *SPDP+MP w/ segment*.

from which sources can select. Easily, the simplest approach to provide failure resilience and path diversity simultaneously is routing on the shortest pairs of disjoint paths (SPDP), but, unfortunately, this far no efficient SPDP routing implementation – subject to the limitations of the current link-state routing architecture – has been proposed. In this paper, building on previous results [15], [16], we proposed a new data structure for encoding SPDP paths and we demonstrated that with tagging packets and using wildcard rules in routers, SPDP paths can be addressed with two forwarding table entries per destination and a 4-bit tag. We showed that our data structure possesses an inherent property, which makes additional source-selectable paths accessible to the sources through tag probing, and thus it yields as an efficient incrementally deployable source-selectable multi-path routing scheme.

Acknowledgements

This work was supported in part by Project no. 134604 and Project no. 128062 that have been implemented with the support provided by the National Research, Development and Innovation Fund of Hungary, financed under the FK_20 and K_18 funding schemes, respectively. G. Rétvári was also funded by the NKFIH/OTKA Project #135606, the MTA-BME Information Systems Research Group and the MTA-BME Network Softwarization Research Group. The research of L. Rónyai was supported in part by the Hungarian Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program.

References

[1] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, p. 269–271, Dec. 1959.
[2] J. Qadir, A. Ali, K.-L. A. Yau, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: A holistic survey of network-layer multipath," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2176–2213, 2015.
[3] M. Chiesa, A. Kamisinski, J. Rak, G. Retvari, and S. Schmid, "A survey of fast-recovery mechanisms in packet-switched networks," *IEEE Communications Surveys and Tutorials*, pp. 1–50, 2021.
[4] K.-W. Kwong, L. Gao, R. Guérin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in ip networks," *IEEE/ACM Transactions on Networking (ToN)*, vol. 19, no. 5, pp. 1543–1556, 2011.
[5] K. P. Gummadi, H. V. Madhyastha, S. D. Gribble, H. M. Levy, and D. Wetherall, "Improving the reliability of internet paths with one-hop source routing," in *USENIX OSDI*, 2004, pp. 13–13.
[6] F. Aubry, S. Vissicchio, O. Bonaventure, and Y. Deville, "Robustly disjoint paths with segment routing," in *ACM CoNEXT*, 2018, p. 204–216.
[7] A. Cianfrani, M. Listanti, and M. Polverini, "Incremental deployment of segment routing into an isp network: a traffic engineering perspective," *IEEE/ACM Trans. on Networking*, vol. 25, no. 5, pp. 3146–3160, 2017.
[8] M. Caesar, M. Casado, T. Koponen, J. Rexford, and S. Shenker, "Dynamic route recomputation considered harmful," *SIGCOMM CCR*, vol. 40, no. 2, pp. 66–71, Apr. 2010.
[9] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," in *ACM SIGCOMM*, 2006, pp. 159–170.
[10] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path splicing," in *ACM SIGCOMM*, 2008, pp. 27–38.
[11] X. Zhao, D. J. Pacella, and J. Schiller, "Routing scalability: an operator's view," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1262–1270, 2010.
[12] D. Meyer, L. Zhang, and K. Fall, "Report from the IAB workshop on routing and addressing," RFC 4984, 2007.
[13] C. Filsfils, S. Previdi, A. Bashandy, B. Decraene, S. Litkowski, M. Horneffer, R. Shakir, J. Tantsura, and E. Crabbe, *Segment Routing with MPLS data plane*, Active Internet-Draft Std., 2015.
[14] Y. Ohara, S. Imahori, and R. Van Meter, "Mara: Maximum alternative routing algorithm," in *INFOCOM*, 2009, pp. 298–306.
[15] J. W. Suurballe and R. E. Tarjan, "A quick method for finding shortest pairs of disjoint paths," *Wiley Networks*, vol. 14, no. 2, pp. 325–336, 1984.
[16] R. G. 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, 1993.
[17] J. Tapolcai, G. Rétvári, P. Babarczi, and E. R. Bérczi-Kovács, "Scalable and efficient multipath routing via redundant trees," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 982–996, 2019.
[18] M. Médard, S. G. Finn, and R. A. Barry, "Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs," *IEEE/ACM Transactions on Networking (ToN)*, vol. 7, no. 5, pp. 641–652, Oct. 1999.
[19] J. Y. Yen, "Finding the k shortest loopless paths in a network," *management Science*, vol. 17, no. 11, pp. 712–716, 1971.
[20] P. Babarczi, A. Pašić, J. Tapolcai, F. Németh, and B. Ladóczki, "Instantaneous recovery of unicast connections in transport networks: Routing versus coding," *Computer Networks*, vol. 82, pp. 68 – 80, 2015.
[21] C. Li, S. McCormick, and D. Simchi-Levi, "The complexity of finding two disjoint paths with min-max objective function," *Discrete Applied Mathematics*, vol. 26, no. 1, pp. 105–115, 1990.
[22] B. Bollobás, *Modern Graph Theory*, ser. Graduate Texts in Mathematics. Springer-Verlag GmbH, 1998.
[23] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer, "Some simplified NP-complete problems," in *ACM Symp. on Theory of Computing*, 1974.
[24] 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, October 2011.
[25] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, "SNDlib 1.0–Survivable Network Design Library," in *Proc. INOC*, 2007.
[26] "LEMON: A C++ library for efficient modeling and optimization in networks." [Online]. Available: http://lemon.cs.elte.hu
[27] J. Tantsura, "The critical role of maximum SID depth (MSD) hardware limitatons in segment routing ecosystem and how to work around those," in *NANOG71*, 2017, pp. 1–21.