Node Virtualization for IP Level Resilience

Máté Nagy, János Tapolcai, Gábor Rétvári
MTA-BME Lendület Future Internet Research Group and MTA-BME Information Systems Research Group
Dept. of Telecommunications and Media Informatics, Budapest University of Technology and Economics
Email: {nagym, tapolcai, retvari}@tmit.bme.hu

***Abstract*—For IP to evolve into a true carrier-grade transport facility, it needs to support fast resilience out-of-the-box. IP-level failure protection based on the IP Fast ReRoute/Loop-Free Alternates (LFA) specification has become industrial requirement recently. The success of LFA lies in its inherent simplicity, but this comes at the expense of letting certain failure scenarios go unprotected. Realizing full failure coverage with LFA so far has only been possible through completely re-engineering the network around LFA-compliant design patterns. In this paper, we show that attaining high LFA coverage is possible without any alteration to the installed IP infrastructure, by introducing a carefully designed virtual overlay on top of the physical network that provides LFAs to otherwise unprotected routers. Our main contribution is formulating the corresponding Resilient IP Overlay Design problem and providing constructions that can achieve full failure coverage against single link failures by adding at most 4 virtual nodes to each physical one. We also show that the problem of finding the minimal number of virtual nodes achieving full failure coverage is NP-hard, and thus propose heuristic algorithms that are guaranteed to terminate with a fully protected topology in polynomial time. According to the numerical evaluations the performance of our algorithm is on par with, or even better than, that of previous ones, lending itself as the first practically viable option to build highly resilient IP networks.**

***Index Terms*—IP Fast ReRoute, Loop Free Alternates, resilience, network optimization**

I. Introduction

The Internet is quickly becoming the main bearing platform for converged telecom services. For the Internet Protocol (IP) suite to become a real carrier-grade transport infrastructure, however, it needs to deliver five-nines availability, the key to which is fast convergence from link and device failures. Historically, the IP control plane adopts a *sluggish restoration mechanism* to handle outages, according to which the Interior Gateway Protocol (IGP), upon detecting a topology change, advertises the altered network state throughout the routing domain, re-computes shortest paths at each router, and then downloads the new forwarding state into the data plane. This process, while robust and easy to configure, is lengthy.

Focusing on this problem IETF founded the IP Fast Re-Route Framework (IPFRR) [1], which is based on two principles, *local* re-routing and the use of *pre-computed* secondary next-hops. Unfortunately, IPFRR today does not come equipped with a practical and deployable implementation that would provide an all-out solution. What the basic IPFRR specification recommends instead, and what most router vendors implement [2]–[6], is Loop-Free Alternates (LFA, [7], [8]), whereby the IGP attempts to find a secondary next-hop that protects against the failure of the default shortest path. The main reason behind LFA's popularity is it's simplicity. First, it does not require any protocol modification that makes it easier to deploy. From the operator's perspective the implementation is rather easy too, and after upgrading all the routers the network itself can remain intact. The price of this simplicity is, however, moderate protection level; in general, LFA protects only 50-80% of single link failures, and node protection is poorer [9]–[11]. Alternatives to LFA that would provide 100% failure protection [9]–[17], unfortunately, could not yet gain sufficient adoption from standardization bodies, router vendors, and network operators, due to the implied management burden and the breaking of the incremental deployment path.

In order to realize high protection coverage with LFA, currently operators need to change the very physical network topology or straight out rebuild it from scratch [8], [18], [19] or, alternatively, re-engineer the default forwarding paths by re-computing the IGP link costs [20], [21]. Both of these approaches, unfortunatley, require significant network operations and managenment intervention and conflict with long-term traffic engineering goals. Meanwhile the IETF has published a tunnel-based extension of the basic LFA method, called Remote LFA (rLFA) [22] that increases LFA efficiency but still does not guarantee full protection. Consequently, there is a compelling industrial motivation to find *LFA-based network optimization techniques, which promise with boosting LFA failure case coverage with minimal or no alterations to the installed IP infrastructure*, until more efficient IPFRR mechanisms eventually become commonly available.

In this paper, we show that improving LFA failure case coverage is feasible without touching the physical topology and the forwarding paths in any ways, or requiring any new features from the IP data and the control planes that are essentially fixed by what is available in commercial network gear today. *The idea is to intervene at the management plane* by taking advantage of router virtualization, a technique for sharing a single IP routing device between multiple virtual routing instances. Instead of altering the original network itself the idea is to create virtual nodes over the physical substrate, a so called *resilient IP overlay*, to provide "virtual LFAs" for the physical nodes that would go unprotected otherwise. Thanks to today's router design, logical instances can easily share the same physical hardware and can behave like completely different devices. Adding virtual (also called "fake") nodes and links to an underlying link-state routing protocol is a well recognized technique to implement new functionalities, such as to enable better load balancing, traffic engineering, and

backup routes [23]. Our resilient IP overlays also borrow from the tunnel-based IPFRR mechanisms and LFA extensions [13], but instead of defining a new control plane protocol we rather only "emulate" tunnels, by a suitably provisioned overlay, intervening solely at the management plane.

The significance of this delegation of the responsibility for IP fast resilience optimization from the control plane to the management plane is not to be dismissed; current IP devices come with the control plane protocols deeply embedded into the hardware and software and despite ongoing efforts to separate the two, like in Software-defined Networking [24] or the ForCES framework [25], this will continueue to be the case for the coming years in many transport and service provider networks. Therefore, most efforts to modify basic IP control protocols have gone unsuccessful for years due to network operators being reluctant to ditch expensive IP network gear (the phenomenon often referred to as "the ossification of the Internet" [26]).

The main question is now how to provision the virtual overlay in a way as to maximize LFA coverage. Special attention must be paid to leave the default forwarding paths, often carefully engineered beforehand to reflect crucial operational concerns [27]–[29], intact. In addition, as a failure of a physical link or node results the failure of its virtual links or nodes we also need to account for local *Shared Risk Link Groups (SRLGs)*, collections of out-links at each node which are likely to fail jointly, for which LFA currently has scarce support for.

As the main contributions of the paper, we formulate the resultant Resilient IP Overlay Design problem as a network optimization problem in a concise mathematical framework and we provide upper bounds on the amount of virtual devices needed and close most of the related algorithmic questions.

- We show the problem is always feasible for 2-connected topologies and full LFA coverage can always be achieved by adding virtual nodes.
- We provide constructions to achieve full LFA coverage in 4- (and 2)-connected graphs by adding 2 (or 4) virtual nodes to each physical node.
- We show that even this sub-problem of adding a single virtual router while maximazing LFA coverage is already NP-complete.
- We propose a greedy optimization strategy that in each step inserts a single, or a small set of, virtual routers into the network that improves LFA protection the most.
- We propose several heuristic solutions as well, which, depending on a configuration parameter, are *either optimal in each greedy step or guaranteed to terminate in polynomial time*, and we show that it is possible to efficiently balance between the two according to the preferences of the operator.
- Furthermore, we present experimental evidence that the proposed techniques are efficient in improving LFA coverage in many common ISP topologies by adding a few virtual routers only.

The rest of this paper is organized as follows. In Section II we walk through existing IPFRR proposals and identify the main barriers for deployment. After a brief introduction to LFA in Sec. III we present the problem formulation in Sec. IV. In Sec. V we show the corresponding network design problem is NP-hard, and show feasibility conditions and constructive bounds on the number of virtual nodes needed for full coverage. In Sec. VI we provide heuristics solving the problem by introducing a greedy optimization strategy. Afterwards in Sec. VII we provide numerical results and finally Sec. VIII concludes the paper.

II. Related work

One of the earliest proposals for immediate recovery techniques use MPLS with the Resource Reservation Protocol-Traffic Engineering (RSVP-TE) [30] extension to reroute the traffic along a precomputed alternate path. Although this provides a standardized and broadly implemented fast protection scheme, there are many operators that have not deployed MPLS at all, or not using RSVP-TE for distributing label information. The only viable option in such cases is IPFRR. A similar approach uses stateful packet forwarding, meaning that whenever a packet is received on an unusual incoming port the node's internal state and the outgoing link are changed [31]. This method offers high resiliency, at the price of modifying packet headers and expecting operators to carry out a complete software upgrade at the data plane level.

Instead, to be able to recover within hundreds of milliseconds in native IP networks, the networking community turned to the IPFRR framework. There is a colorful spectrum of approaches that intend to give more and more efficient propositions for achieving perfect protection, but the majority of them require non-standard IP level functionality or additional management burden, which prevents them to become a real offer for network operators. One example is interface-based forwarding [12] that breaks the traditional IP forwarding principle where the next-hop is solely defined by the destination address. Instead, based on the interface on which the packet arrives, the router can decide if there is a fault in the network and it can find an alternate next-hop to the destination. However, the proposed implementation of FIFR requires an additional forwarding table, together with a so-called backwarding table, for each router interface, which poses a substantial resource burden on routers with many interfaces per line card. This concept was further improved in [32], which simplifies FIFR so that no additional forwarding table is needed; in turn the router must tediously check if the packet is arrived from a link opposite with the primary next hop during forwarding. Unfortunately the current devices do not support changing IP's destination-based forwarding [33]–[35] or introducing some forms of signaling to indicate that a packet is on a detour. What is worse, they do not support out-of-band failure signaling either [36], or use invaluable extra bits in the IP header [37], or adding special information to it for in-band signaling [15].

Other proposals use tunneling to route around the failed component [11], [16], [17], [38]. However, these require the ncapsulatation of the re-routed packets using an additional IP header, which may lead to hard-to-debug MTU issues. The *Failure-carrying Packets* (FCP [39]) method also uses the

packet header to carry information about the failing links that the packet has hit along it's way to the router; correspondingly, FCP can protect against multiple simultaneous failures. Nevertheless, the computational overhead that is required to process the incoming link set and the necessity of an update protocol to maintain routers' consistent view of the network may become significant deployment narriers in the long term.

The *Multiple Routing Configurations* (MRC, [14]) approach calculates a small set of backup network configurations offline and, if a link or node fails in the network, the identifier of the proper recovery configuration is marked in the packet header. This marking enables the routers to switch the packet to the overlay that is free from the failing component. *O2 routing* [40] keeps track of two alternate next-hops towards each destination so that, in case of a link failure, traffic can be immediately switched to the other one. Unfortunately this concept breaks shortest-path-based routing. Fibbing [23] is a concept very similar to ours, whereby routers are "tricked" by using virtual routers into sending traffic into the desired direction. However, Fibbing requires the presence of a centralized SDN controller, whereas our proposal reamins completely within the conventional distributed IP routing model.

In conclusion, the aforementioned solutions are struggling with complexity and management issues; unsurprisingly, LFA has been the only IPFRR method that has been deployed and gained widespread adoption amongst network operators.

III. Router Virtualization and Loop-free Alternates

Loop-Free Alternates is the barebones IP Fast ReRoute specification. To understand LFA in operation, consider the sample network in Fig. 1a and suppose that node a is willing to send a packet to node d. Normally, this occurs via the shortest $a \to d$ path $a - f - d$ of cost 3. However, when a's link to its next-hop f fails, a looses connectivity to d intermittently. In such cases, a is safe to send the packet to e as e still has an intact path to d. In fact, any neighbor suits as long as it does not loop the packets back to a, i.e., is not upstream of a. Such neighbors are called Loop-Free Alternates (LFAs).

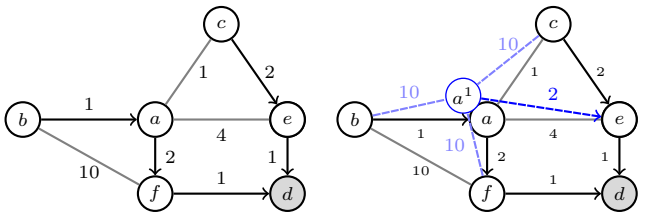
In general, for some source s, destination d, and next-hop t, a neighbor $n \neq t$ of s is a *link-protecting LFA* if [7]:

$$\mathrm{dist}(n, d) < \mathrm{dist}(n, s) + \mathrm{dist}(s, d) \ , \qquad (1)$$

where $\mathrm{dist}(x, y)$ denotes the shortest path distance from node x to node y.

When no ambiguity arises, we omit the default next-hop and simply say "e is an $a \to d$ LFA". However, a is not a $f \to d$ LFA, because if f passed packets to a when (f, d) had failed then those packets would eventually loop back to it along the $a \to d$ shortest path (recall that a is not aware of the failure). In fact, f does not have an LFA to d in this configuration at all, leaving the network vulnerable to the failure of link (f, d). Overall, there are 30 source destination pairs in this six-node network, from which the followings are not protected by LFA: $d \to f$, $d \to e$, $c \to e$, $f \to d$, $c \to b$, $a \to b$ and $c \to a$, resulting a 77% coverage.

In summary, a node does not have an LFA if all its neighbors except the next-hop are upstream. However, if we somehow



(a) The phyisical topology (b) The virtual topology after tadding the virtual router a^1 to a.

Figure 1. Sample network and edge costs.

The network is undirected; however, on the figures we often direct edges to highlight the shortest paths to one node drawn with grey background.

provision a new neighbor that is not upstream to it, then this neighbor will provide a suitable LFA. In this paper, we propose to achieve this by adding a "virtual router" to the physical router, duplicating some of its physical links as virtual links, and assigning costs to these links in a way as to ensure that the new neighbor is no longer upstream. Since a virtual router has a separate routing table and it runs its own instance of the IGP, it will show up as an individual entity in the routing state of its neighbors and hence is eligible as an LFA. This makes it possible to provide LFA to otherwise unprotected routers.

The virtual network obtained by adding a virtual router a^1 to a is depicted in Fig. 1b. For instance setting the IGP costs on the virtual links of b^1 so that the shortest path to d traverses e as next-hop, then a^1 will provide an LFA to $f \to d$. Such an IGP cost setting is $c(b, a^1) = c(f, a^1) = c(c, a^1) = 10$ and $c(a^1, e) = 2$, where c denotes edge cost. Not just that a^1 is now an LFA from f to d, but it also protects several more node pairs too that were unprotected in the default topology. In particular, a^1 provides LFA for $c \to e$ as well, increasing LFA failure coverage from 77% to 83%.

We emphasize that *the same effect could not have been achieved by layer-3 tunnels* (as of [22]), because IP and MPLS/LDP tunnels must follow shortest paths. In contrast, router virtualization allows to establish essentially any tunnel we want, by provisioning consecutive layer-2 virtual links through a series of physically adjacent virtual routers.

There are many appealing aspects of leveraging router virtualization to improve LFA coverage. The isolation of routing contexts provided by virtual routers gives a flexible way to fine-tune the virtual topology to arbitrary protection requirements. Major vendors all support virtualization in hardware in off-the-shelf routers, capable to handle hundreds of virtual contexts [41], [42]. Therefore, *our proposal is deployable right away with minimal management effort*. Improved resilience, however comes at a price, in the form of moderately larger IGP signaling load, IP address management burden, and growing IP forwarding tables at routers. Nevertheless, today's IP routers are powerful enough to let ISPs run hundreds of IGP instances in a single area, and so this price seems negligible for better network robustness and service availability.

The decision of how to provision the virtual overlay is by far a non-trivial one. There are the natural requirements that are already difficult enough to fulfill, like the need to minimize

Suppose that a is about to send packets to f. If, for some reason, link (a, f) goes down, a may choose to redirect its traffic to the LFA e^1. However, said traffic will never arrive to f as the $e^1 \to f$ detour degrades into the LFA loop $a - e^1 - c^1 - a^1 - c - a$. Here, a^1 also switches to its LFA c realizing that its link to f has disappeared due the physical failure which a^1 is unaware of.

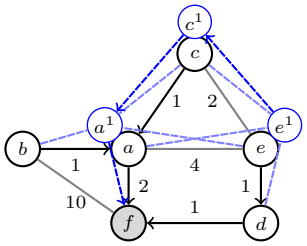


Figure 2. A sample network with a possible LFA loop.

the number of virtual instances executing on a router side by side. But there are much more subtle issues to consider as well, like the curious fact that a careless intervention might very well *decrease* LFA coverage instead of *increasing* it. We find that inadvertent virtualization decision might easily end up *corrupting existing LFAs and decreasing LFA-coverage*, instead of increasing it. To demonstrate how this can happen, we give a virtual topology in Fig. 2 where the spurious LFA provisioned at the virtual router e^1 may replace the legitimate LFA e. Naturally, such cases must be avoided at all costs.

IV. Model and Problem Formulation

Router virtualization opens up a broad range of new LFA-optimization strategies. In this section, we narrow this wealth of options to a well-defined, practically motivated subset. The main goal is to minimize interference with the normal operation of the network, and only involve virtual routers in packet forwarding when absolutely necessary. This guarantees that packets do not take excess detours, helps break down management complexity, and eases debugging data plane misconfigurations. The requirements are as follows.

A. Assumptions on Physical and Virtual Topology

Problem inputs. We are given the physical network, or *substrate*, as an undirected graph $G_S = (V_S, E_S)$ and IGP link costs $c_S : E_S \mapsto \mathbb{Z}^+$. We assume that the substrate consists of point-to-point links only (i.e., no LANs, NBMA media, etc.). Our task is to construct a virtual topology $G_V = (V_V, E_V)$ with link costs $c_V : E_V \mapsto \mathbb{Z}^+$ in a way as to maximize LFA coverage in G_V. In fact, G_S is a subgraph of G_V in our model.

There is a default routing and forwarding layer G_S. We assume that associated with each physical router there is a default context, holding the interface and loopback IP addresses of the physical router, running the common control and management protocols a router usually runs, and originating and terminating all traffic entering or leaving the network at that router. Nodes in $G_V \setminus G_S$ are called *virtual nodes* or contexts and links are called *virtual links*. Mark the default context for a physical router $v \in V_S$ and denote its virtual nodes by $v^i \in V_V \setminus V_S$ for $i = 1, \ldots, k_v$ where k_v denotes the number of virtual instances running on the particular physical router. Let k_{max} denote the maximal number of virtual instances a router may have. In addition, denote the set of neighbors of some node $v \in V_S$ in G_S by $N_S(v)$. Similarly, $N_V(v)$ denotes the neighbors of some $v \in V_V$ in G_V.

Traffic flows in the default layer along the default shortest paths. Traffic only enters a virtual router when a failure shows up, and so virtual routers serve exclusively as LFAs for nodes not protected in the physical topology. This minimizes the disruptions under error-free conditions and ensures that in normal operations the virtual topology distributes load as efficiently as the underlying physical network. To achieve this, the cost of virtual links is set so that they never appear in any $u \to v$ shortest path in G_V for any $(u, v) \in V_S \times V_S$, $u \neq v$.

Virtual links connect physically connected nodes. Virtual links are provisioned between nodes that are adjacent in the substrate, or inside the same the physical router:

$$\forall (v^k, u^l) \in E_V \setminus E_S : \{(u, v) \in E_S \text{ or } u = v)\} \ . \quad (2)$$

The reasons for this assumption are manifold. First, as virtual links never span multi-hop paths, they are easy to provision as layer-2 virtual links (say, Ethernet VLANs). Such connections often do not even require distinct IP addresses. This minimizes impact on the IP layer and eliminates much of the configuration overhead and MTU issues that plague tunnel-based IPFRR mechanisms [11], [17]. Additionally, layer-2 connections are free from the limitations of layer-3 tunnels, which are bound to shortest paths. Finally, two virtual links now belong to the same SRLG if and only if they share the same physical link, which would not hold over multi-hop tunnels.

Single link failures in the physical network. As single link failures have been shown to constitute the major portion of unplanned outages in operational networks [43], we concentrate on this case and we also ignore node failures; these cases can be incorporated into the model with little extra effort. We observe, however, that even a single physical link failure usually manifests itself as multiple simultaneous failures in the virtual topology, because not just the default link but all the virtual links provisioned on it also go down.

The LFA specification introduces *local SRLGs* as the minimum requirement for conforming implementations [7]. A local SRLG at node v is defined as a set of output ports, i.e. links adjacent with the node, with the semantics that a conforming IGP will never install an LFA through a link that shares a local SRLG with the primary next-hop. Associated with each link $(u, v) \in E_S$ we define local SRLGs at both end nodes composed of all the links provisioned on the same physical link as e, formally

- SRLG at v is $\mathcal{S}_{(v,u)} = \{(v, u), (v, u^1), \ldots, (v, u^{k_u})\}$,
- SRLG at u is $\mathcal{S}_{(u,v)} = \{(v, u), (v^1, u), \ldots, (v^{k_v}, u)\}$.

Local SRLG support is easy to deploy as it does not need network-wide configuration and dissemination mechanisms [44], [45], but it is also quite limited in that routers will only spot non-SRLG-disjoint paths at the first hop.

LFA is disabled for virtual nodes. Fig.2 shows an example where, after a failure, a virtual node forwards the packet to its LFA. Our experience suggests that such "cascade LFAs" are the primary origin of LFA loops and can be omitted without any perceptible degradation in the resultant LFA coverage. Correspondoingly, we assume a packet can be deflected to

an LFA only at most once during its journey from the source to the destination.

B. Problem formulation

First, we present an augmented LFA definition that accounts for local SRLGs and reules out LFA loops.

Definition 1: For source s, destination d, and $s \to d$ next-hop t, node n is an $s \to d$ *LFA* if

LFA-1 $n \in N_V(s)$ and $n \neq t$,
LFA-2 $\text{dist}(n,d) < \text{dist}(n,s) + \text{dist}(s,d)$,
LFA-3 $(s,n) \notin \mathcal{S}_{s,t}$ (local SRLG condition), and
LFA-4 $\text{dist}(n,d) < \text{dist}(n,s^i) + \text{dist}(s^i,d)$ for $i = 1,\ldots,k_s$.

Here, LFA-4 requires that n is an LFA with respect to *all* virtual neighbors[1]. Since current LFA implementations are restricted to LFA-1, LFA-2, and LFA-3, our virtual overlay construction algorithms need to be designed so that LFA-4 automatically fulfill.

Next, we define the following metric for LFA failure case coverage in the virtual topology. Let $I_E(s,d)$ be an indicator function, taking the value 1 if (s,d) is protected and zero otherwise, where $s,d \in G_S$. Using this definition, the *LFA coverage* is defined as

$$\eta(G_V, c_V) = \frac{\sum_{(s,d)\in V_S \times V_S} I_E(s,d)}{|V_S|(|V_S|-1)} \quad . \tag{3}$$

With these notations in place, we can now pose the *Resilient IP Overlay Design* (RIOD) problem. Here, the task is to compute the overlay that maximizes LFA-coverage, using only a given number of virtual routers. In addition, we also allow to limit the set of routers that can host virtual instances.

Definition 2: $RIOD(G_S, c, U, k, l)$: given a graph $G_S = (V_S, E_S)$, link costs c, node set $U \subseteq V$, and positive integer k, design a graph $G_V = (V_V, E_V)$ and link costs c_V so that:

- $V_S \subseteq V_V$ and virtual nodes provisioned only inside U,
- $E_S \subseteq E_V$ and virtual links are only between physically connected routers, see Eq. (2),
- shortest paths between node pairs in V_S do not change (the substrate is unaltered),
- $|V_V \setminus V_S| \leq k$ (no more than k virtual instances), and
- $\eta(G_V, c_V) \geq l$ (*the LFA coverage is a least* l).

One ultimate goal is to build an entire virtual topology in one step so that each node-pair in the substrate becomes LFA-protected. We also focus on a somewhat less ambitious task $RIOD(G, c, \{v\}, 1, l)$ to add a *single virtual router to a selected node* v, and our objective is merely *to maximize LFA coverage* along the way instead of aiming for full coverage. We shall refer to this useful special case of RIOD as the *LFA Virtual Router Augmentation Problem* $\text{LFAVirt}(G, c, v)$.

[1]Note that in [18] instead of LFA-4 the following weaker condition was given: LFA-4* each $n \to d$ shortest path is SRLG-disjoint from e.
We chose this stricter version of the LFA condition, because in very rare situations the original weaker condition caused LFA loops when many virtual nodes are added (see Sec. III.A in [46] for an example and further discussion).

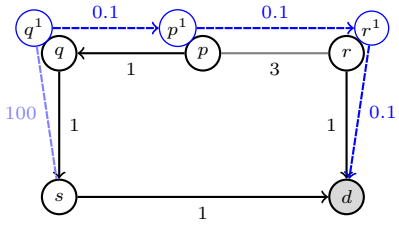


Figure 3. A sample network with a "virtual tunnel" between nodes s and d.

V. Complexity and Upper Bounds

Next, we address the problem of building an LFA-optimized overlay under the model assumptions introduced above. The model basically asks for augmenting a physical topology with virtual routers and set the cost on the resultant virtual links in a way as to maximize LFA coverage.

First, we will show that full LFA coverage can always be achieved but we show that finding the fewest virtual nodes to achieve this is NP-hard; however, we will provide constructions to achieve full LFA coverage in 4- and 2-connected graphs by adding limited number of virtual nodes.

A. Conditions to Reach Full Failure Coverage

Theorem 1: For a given 2-connected graph G with positive costs c there always exists an overlay G_V and cost setting c_V that solves $\text{RIOD}(G, c, V, \infty)$ with $\eta(G_V, c_V) = 1$.

Proof: We show that, given any unprotected node-pair $s \to d$, there is a proper set of virtual nodes whose addition will create a link-protecting SRLG-disjoint $s \to d$ LFA. It follows that if we apply this step to each unprotected node-pair, then full LFA-coverage eventually reaches. We protect the $s \to d$ pair by provisioning a *"virtual tunnel"* between s and d that provides a detour for s bypassing its failed next-hop (Fig. 3). Let $s - q - \ldots - r - d$ be an $s \to d$ path disjoint from the $s \to d$ next-hop (such a path is guaranteed to exist as G is 2-connected). Create a virtual node for each node between q and r and denote the new virtual router on q by q^1 and the one on r by r^1. Connect s to q^1 and r^1 to d and set the link cost on (s, q^1) "high" (say, larger than the length of the longest shortest path) and at the rest of the virtual links to the lowest possible. As one easily checks, q^1 is now an $s - d$ LFA. We still need to show that LFA-4 holds, but this is guaranteed as there are only two entry points to the virtual tunnel, q^1 and r^1, and q^1 is protected by the local SRLG at s (as of LFA-3) and r^1 is never an LFA due to its low cost. ■

B. Complexity analysis

The first question we ask is whether the RIOD problem is tractable. Consider the below characterization.

Theorem 2: $\text{LFAVirt}(G_S, c, v)$ is NP-complete.

Note that this implies that $RIOD(G_S, c, U, k, l)$ is NP-complete. The main idea of the proof is constructing a special substrate and designating a node in a way that virtualizing the node opens up a plethora of LFA-options. Deciding on which LFA to choose means fixing c_V, which is then shown to solve arbitrary instances of the *minimum feedback arc set* problem,

a well-known NP-complete problem [47, GT8, pg.192.]. For the complete proof, refer to the Appendix A.

C. Constructions to Achieve Full LFA Coverage by Adding Limited Number of Virtual Nodes

Let us introduce some constructions.

Definition 3: Let G be a connected graph and T a spanning tree, and K be a constant larger than the longest shortest path between any two nodes in G. Adding a k^{th} *virtual layer with distance* K means we add a virtual node v^k to each physical node v, where virtual nodes u^k and v^k are connected if and only if $(u, v) \in T$. The cost of $c(u^k, v^k) = \epsilon$, where ϵ is a small positive number (smaller then the smallest cost edge of E_S divided by $|V_S|$). For each node v we also add virtual edge (v, v^k) with cost $c(v, v^k) = K$.

Claim 1: Let G be a connected graph and T a spanning tree which is added as a virtual layer with distance K. This cannot decrease the LFA coverage, and the shortest path between virtual node v^k and physical node d is $T(v^k - d)$, where $T(x - y)$ denotes the path between x and y in tree T.

Proof: First we show that adding a new virtual layer with distance K cannot decrease the LFA coverage. One can verify that Claim 2 can be applied recursively: start with any node and add its neighbors in the tree one by one.

Next we show that the shortest path between v^k and d is $T(v^k - d)$. Note that any path from v^k to d is at least K because it should traverse between the layers and thus through a link (u, u^k) at some node u. The path in tree T between v^k and d^k is unique and, traversing along the virtual nodes, its cost is $h \cdot \epsilon$ where h is the number of hops. The total cost if $u = d$ is $K + h \cdot \epsilon$, while any other path would have larger cost as it should traverse an edge with cost K and a physical edge with cost larger than $h \cdot \epsilon$. ■

Theorem 3: For a given 4-connected graph G with positive costs c there always exists an overlay G_V and cost setting c_V that solves $\text{RIOD}(G, c, V, \infty)$ with $\eta(G_V, c_V) = 1$ and $k_{max} = 2$.

Proof: A corollary of Tutte-Nash-Williams Theorem [48, Thm. 4.4.4] is that an undirected graph $G = (V, E)$ contains 2 pairwise edge-disjoint spanning trees if it is 4-edge-connected. Let T_1 and T_2 denote such pairwise edge-disjoint spanning trees of the input topology G. Let us add T_1 and T_2 to G as two virtual layers with distance K.

In this case, for any node pair n and d the short path from n^1 to d and from n^2 to d are two disjoint paths, by using Claim 1 and that T_1 and T_2 are pairwise edge-disjoint spanning trees. Therefore every node s has an LFA s^1 or s^2 having a shortest path to d that is disjoint from the failed next hop. ■

Theorem 4: For a given 2-connected graph G with positive costs c there always exists an overlay G_V and cost setting c_V that solves $\text{RIOD}(G, c, V, \infty)$ with $\eta(G_V, c_V) = 1$ and $k_{max} = 4$.

See Appendix B for the proof, which is based on st-numbering technique of the undirected graph topology introduced by Itai and Rodeh [49].

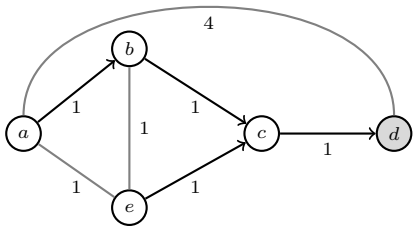


Figure 4. A graph where LFA coverage cannot be increase by adding a single virtual node.

VI. Heuristic Algorithms to the Resilient IP Overlay Design Problem

The main idea in our heuristics is to iteratively add new virtual nodes to the network until full LFA coverage is achieved.

First, we focus on the special case $\text{LFAVirt}(G, c, v)$ where a single virtual node v^1 is added to v and the task is to set the link costs so that LFA coverage increases the most. We will show a sufficient condition to ensure the LFA coverage is never decreasing. Recall that in Section V-B we have shown that even this simple variant is already NP-complete (which immediately sets the complexity of the one-step optimization problem as NP-hard), but it is still complex enough to build efficient optimization strategies on top of it.

As the next step, we can use $\text{LFAVirt}(G, c, v)$ as a building block for a greedy heuristic to solve the fully fledged $RIOD(G, c, U, 1, \max)$, whereby iteratively a single virtual node is added to the network to increase the LFA-coverage. This approach would curtails the ensuing computational complexity significantly; on the negative side, as we show below this simple greedy approach may stuck in a local maximum in certain cases. Therefore, we shall use a slightly extended version of LFAVirt as the basic building block of our algorithmsm whereby we may instantiate multiple virtual LFAs in each step if the need arises.

Theorem 5: There are networks with $\eta(G_V, c_V) < 1$ where adding a single node cannot increase the LFA coverge.

Proof: We show a counter-example in Fig. 4. Here, $\eta = \frac{19}{20}$, as the only unprotected node-pair is $c \to d$. Additionally, all the 2-hop neighbors of c are upstream for d, and because Alg. 1 provisions only a single virtual router in each step the furthest it can reach with a virtual tunnel is a, which is also upstream. Hence, any greedy algorithm that adds only a single node in each step terminates with $\eta < 1$. ■

It may be tempting to believe that our counter-example is a pathologic case due to the large cost of the (a, d) link. This, however, is not the case, as one can easily show unit-cost counter-examples similar to the one in Fig. 4 (e.g., by substituting the (a, d) link with a long chain of unit-cost links).

A. Basic Heuristic

As it turns out, augmenting the graph with only a single virtual node in each iteration might be too restrictive. Instead, one might try to instantiate two virtual routers when adding only a single one did not help, then try three virtual routers at

once, etc. This observation is reflected in the below definition of *connected l-sets*.

Definition 4: For a graph G, call a set of nodes in $U_l \in V$ a *connected l-set* if the induced subgraph of G spanned by U_l is connected and $|U_l| = l$.

Claim 2: Adding a set of nodes $U_l \in V$ the LFA coverage cannot decrease, if node $v' \in U_l$ does not appear in any $u \to w$ shortest path in G_V for any $(u, w) \in V_V \times V_V$, $u \neq w$.

Proof: To decrease the LFA coverage there must be a source s, destination d, and $s-d$ next-hop t, which had LFA through node n before v' was added, however adding v' this LFA is not valid any more. This can only happen if the shortest path from node v' to d has changed, which is not possible according to the condition in the claim. ∎

Our heuristic is then based on simply trying increasingly larger connected sets of virtual nodes until LFA-coverage eventually improves. Note that, by Theorem V-A, there is a sufficiently large connected l-set for which at least one node-pair will gain a new LFA, and therefore this modification to the algorithm implies optimal termination. The greedy algorithm implements these ideas.

Algorithm 1: Greedy algorithm for $\mathrm{RIOD}(G, c, V, \infty)$

```
while 1 > η(G, c) do
    foreach l = 1, ..., k do
        foreach connected l-set U_l ⊆ U do
            (c_{U_l}, η_{U_l}) ← solve LFAVirt(G, c, U_l)
        (U', η') ← choose U_L ∈ U that maximizes η_{U_l}
        if η' > η then
            add U' to G and set costs to c_{U'}
            break
```

Note that the algorithm is parametrized on an integer k, which allows to set an upper bound on the maximum size of the connected l-sets examined, and hence on the running time.

There still remains the problem of how to solve the general form of the LFA Virtual Router Augmentation problem $\mathrm{LFAVirt}(G, c, U_l)$. Here, we need to assign an entire "island of virtual nodes" on a connected set U_l. Let $\mathrm{neigh}(U_l)$ denote the neighbors of nodes in U_l that are outside U_l. Suppose that we are about to solve $\mathrm{LFAVirt}(G, c, U_l)$ by provisioning a set of virtual routers U_l' on U_l, with each $u \in U_l$ hosting a single virtual instance u'.

The main idea of our heuristics is to set a single exit point for the virtual nodes in U_l', that is, to let all traffic that enters U_l' through LFAs leave via a single exit link (v', g). Thus, we create a virtual router at each node of U_l and we connect these to each other with small cost, plus we connect these nodes to all nodes in $\mathrm{neigh}(U_l)$ with a large cost except for the virtual link to the exit link (v', g) which is again set to low cost, just like in Fig. 1b. It is now trivial to check that this setting indeed yields that U_l' has a single exit node: g'. In this way the LFA coverage never decreases by Claim 2.

This is a simple yet efficient method, with the main drawback that for each $g \in \mathrm{neigh}(U_l)$ we need to evaluate the LFA coverage. The LFA coverage is computed according to Eq. (3) which needs evaluating every edge-node pair: $(s, n) \in E_V$ and $d \in V_D$. Recall, $s \in V_S$, thus the number of possible (s, n) LFA candidate links are at most $k_{max}|V_S|$. This is performed for each node $g \in \mathrm{neigh}(U_l)$; thus, the process takes $O(k_{max}|V_S|^2 \cdot |\mathrm{neigh}(U_l)|)$ steps in total each time virtual nodes U_l are added.

B. Reducing the running time of the basic heuristic

Clearly evaluating Eq. (3) is the bottleneck in the running time as it must be launched each time a virtual node is added. To speed up the basic heuristic we will incrementally evaluate Eq. (3). To do so, we keep track the set of *eligible node-pairs* $\mathcal{L}$ that can gain an LFA. Clearly, a virtual router u' can provide LFA only if it is a neighbour of the source node. Let $\mathcal{L}_{U_l} \subseteq \mathcal{L}$ denote the set of eligible node-pairs with source node adjacent with U_l, formally $\mathcal{L}_{U_l} \subseteq \mathcal{L} | (s, d) \in \mathcal{L}, s \in \mathrm{neigh}(U_l)$. In other words, the new virtual nodes U_l can provide LFA to node pairs $\mathcal{L}_{U_l}$; thus $\frac{|\mathcal{L}_{U_l}|}{|V_S|(|V_S|-1)}$ is the upper bound in the increase of $\eta(G)$ after adding U_l. This measure helps in selecting a proper virtual nodes U_l to add.

Suppose that s is about to send packets to d, and $v = U_1$. By setting the cost of virtual link (v^1, g) small, and (s, v^1) large, node s will have an LFA through node v'.

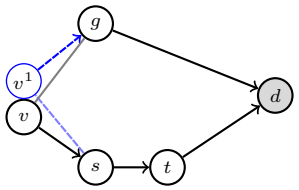


Figure 5. Illustration for escape nodes.

Let $(s, d) \in \mathcal{L}_{U_l}$ be the set of source-destination pairs that can gain an LFA when new virtual nodes U_l are added to the physical node v. Note that node s has no LFA, thus it has a single next hop $t = \mathtt{nh}(s, d)$ and all of its neighbours are upstream to it or a virtual node corresponding to s. We seek nodes $g \in \mathrm{neigh}(U_l)$ that can provide LFA to $s-d$ by U_l.

The key idea is to pre-calculate the following set: for each $(s, d) \in \mathcal{L}_{U_l}$ a set of *escape nodes* $\mathcal{E}_{s \to d}$ consisting of the nodes which, if chosen as the only exit link of U_l to d, would render a virtual node $v' \in U_l$ an $s \to d$ LFA. Computing $\mathcal{E}_{s \to d}$ for all $(s, d) \in \mathcal{L}_{U_l}$ takes $O(|\mathrm{neigh}(U_l)| \cdot |\mathcal{L}_{U_l}| \cdot k_{max})$ steps in total. Based on this, we can select optimal g as follows

$$g = \underset{g' \in \mathrm{neigh}(U_l)}{\mathrm{argmax}} \; |(s, d) \in \mathcal{L}_{U_l} : g' \in \mathcal{E}_{sd}| \quad . \tag{4}$$

The computational complexity of the improved Alg. 1 is $O(n^3 + n^{k+3})$. Here, $O(n^3)$ comes from the all-pairs shortest path problem needed be solved to obtain $\mathrm{dist}(.)$ and $O(n^k)$ is the number of connected l-sets U_l of size at most k. The above heuristics solves $\mathrm{LFAVirt}(G, c, U_l)$ for each U_l also in $O(n^3)$, as $\mathcal{L}$ and $\mathcal{L}_{U_l}$ contain $O(n^2)$ elements and $\mathcal{E}_{sd}$ contain $O(n)$ elements, and each can be calculated in $O(n^2)$ steps.

Correspondingly, Alg. 1 allows to trade-off optimality for computational complexity through the parameter k. When running time is of no concern then k can be set to n, in which case we are guaranteed to obtain a fully-protected overlay in a finite number of steps. On the other hand, fixing k at a small constant results strictly polynomial running time. For the rest of this paper we set $k = 3$. In this case the theoretical worst-case

complexity of Alg. 1 is $O(n^6)$ steps, however, in practice we found the all-pairs-shortest path problem to dominate running time and hence $O(n^3)$ to be a more reasonable complexity characterization. Besides, using a reduced set of connected l-sets for selecting U_l can also improve the performance of Alg. 1. Such a modification is when we use *shortest path slices* of rank l, meaning that only those set of nodes are examined in each step that are part of an existing shortest path in G with a length of l. Thus, $O(n^k)$ is reduced to $O(n^2)$. As we shall see in the next section, this setting yields an overlay with close to perfect LFA-coverage in most practical cases with very fast running time.

C. Integer Linear Program for $\mathrm{LFAVirt}(G, c, v)$

Since the $\mathrm{LFAVirt}(G, c, v)$ problem is NP-hard, we also provide an Integer Linear Programming (ILP) that will serve as the baseline when evaluating the heuristics. The formulation is based on the definition of escape nodes discussed in the previous subsection; however, instead of selecting one escape node, the ILP computes the optimal virtual link costs so that the most escape nodes become next-hops for the new virtual node v' and hence LFA-coverage is maximized when adding v'.

The ILP is built around the following two constraints (see Appendix C for details):

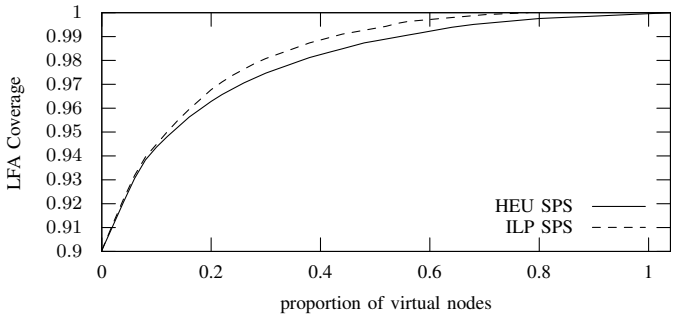
1) We want to assign at least one escape node as the next-hop of v' towards d, as then v' will provide a new LFA to $s \to d$. For this, we need to set link costs c_V such that
$$\mathrm{dist}(v', d) = c_V(v', g) + \mathrm{dist}(g, d) \text{ for some } g \in \mathcal{E}_{sd} \ . \quad (5)$$
2) Besides we need to ensure the LFA-4 condition holds
$$\mathrm{dist}(v', d) < c_V(v', s^i) + \mathrm{dist}(s^i, d) \text{ for all } i = 1, \ldots, k_s. \quad (6)$$

We shall also evaluate a simple extrionsion of the ILP that allows to add more than a single node in a each step.

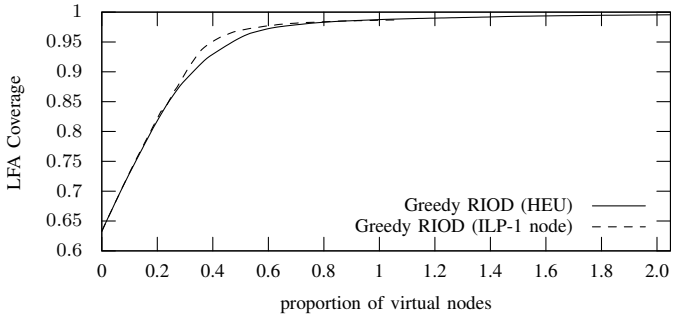
VII. Simulation results

We evaluated the performance of different RIOD implementations in extensive numerical studies. In particular we developed the ILP and heuristic of Alg. 1 and the 4-layered construction of Theorem 4. The implementation was done in C++ with the help of the LEMON graph library [50]. The simulations were run on a Linux PC with an Intel 3.3GHz CPU and 4G RAM. As an input for the measurements we used numerous real-life ISP topologies; namely we used ISP topologies [51] and the Rocketfuel data-set [52], where we set costs randomly wher link costs were not available.

In the first run, we compared the performance of Alg. 1 when the embedded LFAVirt instances were solved with the heuristic and ILP implementations, respectively. The parameter k was set to 3 (i.e. $|U_l| = 3$) meaning that the algorithm tries to provision node sets with size at most 3 in each step. As we observed in our measurements that there is a minimal difference in the attained LFA coverage in case of using shortest path slices instead of all possible connected l-sets, we settle for using the former one in order to improve



(a) Germany_50



(b) Deltacom

Figure 7. Progression of LFA coverage in backbone topologies.

the execution time of our algorithm. The detailed results are summarized in Table I. Here, the values refer to the LFA coverage (η) attained by the heuristic as an increasing number of virtual nodes are provisioned in the network. The deviation compared to the ILP is shown in parenthesis.

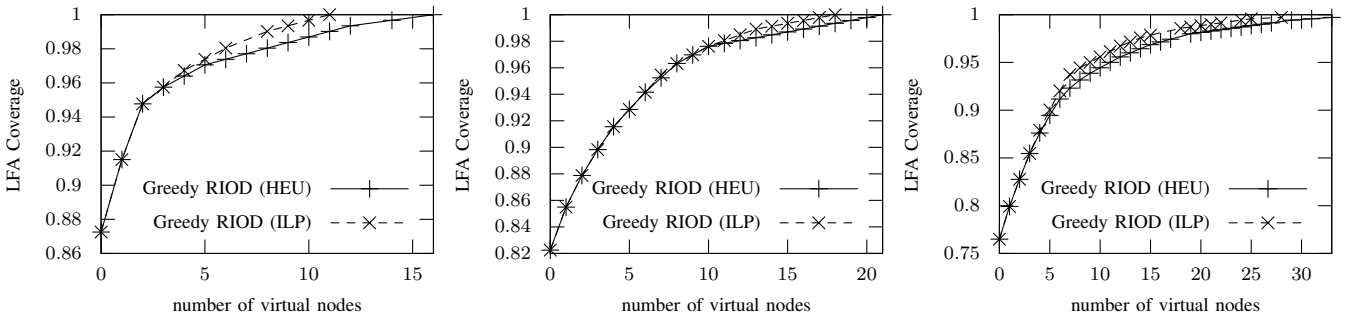
The most important observations are as follows. First, the initial LFA coverage is 70-90% and it can be easily boosted up to 95%, just by introducing a single virtual instance for half of the physical nodes. Second, the heuristic barely overshoots the ILP in the cases when the number of virtual nodes is less than, or equal to n. The last two columns show that if we do not maximize the amount of virtual nodes the algorithm terminates with $\eta = 1$ in most of the cases, and there are only 3(!) networks whereby it gets stuck with the setting $k = 3$. The price that we pay for reaching this final state is given below as v_∞. As an average the heuristic shows some 19% overhead compared to the the ILP when adding n virtual nodes, however it executes 10 times faster. The reason is that the LFA coverage improvement has a logarithmic trend, so if the goal is to solely improve LFA protection to a certain level then a couple of new nodes are usually enough. In contrast, to achieve full protection, we need to provide alternate tunnels from all sources to all destinations that can significantly increase the size of the virtual layer.

This is clearly visible on Figs. 6-7 where we show the progression of the algorithms. In the first phase there is a steep increase in the LFA protection and the performance gap between the algorithms is minimal. We also observe that in most steps the algorithms prefer to add a single virtual node, however there are cases (see e.g., Fig. 6c) when both methods need a tunnel to overcome a certain complex scenario. We also show a topology (Deltacom) where the ILP with $k = 3$

Table I

RESULTS OF RIOD$(G, c, v, 3)$ USING SHORTEST PATH SLICES: TOPOLOGY NAME, NUMBER OF NODES (n) AND EDGES (m); INITIAL LFA COVERAGE (η_0; [%]), ATTAINED LFA COVERAGE ($\eta_{\frac{n}{4}}$, $\eta_{\frac{n}{2}}$, η_n) AND RUNNING TIME ($t_{\frac{n}{4}}$, $t_{\frac{n}{2}}$, t_n; $[sec]$) WHEN PROVISIONING $25-50$ AND 100% OF THE PHYSICAL NODES RESPECTIVELY, THE FINAL LFA COVERAGE (η_∞) WITH THE REQUIRED EXECUTION TIME (t_∞) AND THE RATIO OF VIRTUAL NODES IN THE FINAL STATE (v_∞; [%]). VALUES IN PARENTHESES REPRESENT THE DIFFERENCE COMPARED TO THE CORRESPONDING RESULT OF THE ILP. IN CASE OF LFA COVERAGE AND RATIO THE VALUES ARE GIVEN IN PERCENTAGE (E.G. IF n VIRTUAL NODES ARE ALLOWED, THEN ILP OUTPERFORMS THE HEURISTIC BY GAINING 4% BETTER LFA COVERAGE FOR THE ABILENE NETWORK, AND REACHES THE PERFECT COVERAGE BY USING 36% LESS VIRTUAL NODES). THE ILP EXECUTION TIME IS GIVEN IN THE MULTIPLES OF RUNNING TIME NEEDED BY THE HEURISTIC (E.G. IN CASE OF ABILENE, THE ILP RUNS 9 TIMES SLOWER THAN THE HEURISTIC, REQUIRING 135 SECS.)

Topology	n	m	η_0	$\eta_{\frac{n}{4}}$	$t_{\frac{n}{4}}$	$\eta_{\frac{n}{2}}$	$t_{\frac{n}{2}}$	η_n	t_n	η_∞	t_∞	v_∞
Abilene	11	14	0.61	0.70 (.00)	0.1 (>23)	0.80 (.01)	0.2 (>31)	0.92 (.04)	0.5 (>20)	1 (.00)	1.5 (>9)	1.72 (.36)
Germany	17	25	0.69	0.83 (.01)	0.7 (>42)	0.90 (.01)	1.6 (>25)	0.96 (.02)	4.8 (>16)	1 (.00)	12.1 (>6)	1.58 (.29)
BtEurope	17	30	0.96	0.98 (.00)	3.4 (<1)	0.99 (.00)	8.5 (<1)	1 (.00)	13.2 (<1)	1 (.00)	13.2 (<1)	0.70 (.12)
AS6461	17	37	0.93	0.99 (.00)	2.6 (>1)	1 (.00)	5.4 (>1)	1 (.00)	5.4 (>1)	1 (.00)	5.4 (>1)	0.35 (-.12)
Internet_MCI	18	32	0.95	0.98 (.00)	1.8 (>5)	0.99 (.00)	6.3 (>2)	1 (.00)	7.7 (>2)	1 (.00)	7.7 (>2)	0.55 (.00)
AS1755	18	33	0.87	0.96 (.00)	2.1 (>10)	0.98 (.01)	6.8 (>5)	1 (.00)	15.6 (>2)	1 (.00)	15.6 (>2)	0.88 (.27)
ChinaTelecom	20	44	0.95	0.98 (.01)	19.5 (>2)	0.99 (.01)	49.9 (<1)	1 (.00)	60.7 (<1)	1 (.00)	60.7 (<1)	0.65 (.20)
AS3967	21	36	0.78	0.95 (.00)	3.3 (>140)	0.99 (.00)	9.2 (>50)	1 (.00)	14.0 (>30)	1 (.00)	14.0 (>30)	0.61 (.00)
BellSouth	21	36	0.79	0.96 (.01)	12.4 (>90)	0.98 (.01)	44.9 (>25)	1 (.00)	149.9 (>7)	1 (.00)	149.9 (>7)	0.85 (.33)
ATnT	22	38	0.82	0.92 (.00)	6.2 (>230)	0.97 (.01)	19.1 (>95)	1 (.00)	63.3 (>30)	1 (.00)	63.3 (>30)	0.95 (.14)
NSF	26	43	0.86	0.93 (.00)	6.1 (>50)	0.97 (.01)	18.1 (>25)	1 (.00)	46.9 (>10)	1 (.00)	46.9 (>10)	0.88 (.08)
BICS	27	42	0.76	0.91 (.01)	3.8 (>25)	0.96 (.01)	11.6 (>10)	0.99 (.00)	42.1 (>3)	0.99 (.00)	52.1 (>3)	1.25 (.22)
AS3257	27	64	0.92	0.99 (.00)	50.7 (>9)	1 (.00)	160.8 (>3)	1 (.00)	160.8 (>3)	1 (.00)	160.8 (>3)	0.40 (-.04)
AS1239	30	69	0.87	0.98 (.00)	54.0 (>25)	0.99 (.01)	109.0 (>13)	1 (.00)	130.7 (>11)	1 (.00)	130.7 (>11)	0.60 (.14)
Arnes	31	47	0.83	0.97 (.00)	10.1 (>34)	0.98 (.01)	33.5 (>13)	0.99 (0.01)	134.2 (>4)	1 (.00)	143.4 (>3)	1.09 (.13)
Geant	31	49	0.83	0.96 (.00)	8.7 (>55)	0.98 (.01)	28.2 (>29)	0.99 (0.01)	79.6 (>7)	0.99 (.01)	86.1 (>7)	1.09 (.09)
Italy	33	56	0.78	0.89 (.00)	24.3 (>470)	0.94 (.02)	58.0 (>210)	0.99 (.00)	178.5 (>70)	1 (.00)	367.5 (>36)	1.57 (.24)
BtNorthAmerica	36	76	0.83	0.96 (.00)	65.6 (>95)	0.99 (.00)	211.0 (>31)	1 (.00)	407.1 (>16)	1 (.00)	407.1 (>16)	0.80 (.17)
BellCanada	39	55	0.61	0.80 (.00)	10.2 (>150)	0.90 (.01)	25.9 (>75)	0.98 (0.01)	73.9 (>29)	1 (.00)	169.6 (>13)	1.82 (.34)
Germany_50	50	88	0.90	0.96 (.01)	60.1 (>14)	0.99 (.00)	168.2 (>45)	0.99 (0.01)	412.8 (>19)	1 (.00)	430.0 (>18)	1.04 (.26)
Deltacom	103	151	0.63	0.85 (n/a)	476.4 (n/a)	0.96 (n/a)	950.1 (n/a)	0.99 (n/a)	2494 (n/a)	0.99 (n/a)	8323 (n/a)	2.04 (n/a)
Average	29.2	50.7	0.81	0.92 (.002)	39.1 (>73)	0.96 (.007)	92.2 (>34)	0.99 (.005)	214.0 (>14)	0.99 (.0005)	507.6 (>10)	1.02 (.19)



(a) AS1755 (b) AT&T (c) BICS

Figure 6. Progression of LFA coverage in small and middle-sized networks.

does not perform in acceptable running time; this validates the need for efficient heuristics.

Finally, we evaluate the change in the length of the detours in order to demonstrate that traffic entering the virtual layer does not spend too much time there and hence it does not utilize the physical resources extensively. The values provided in Fig. 8 are the overhead compared to the default paths in percentage. We observe that Alg. 1 results in a ~30% of overhead, whereas using the 4-layer construction of Thm 4 the increase becomes roughly twofold.

In summary, we see very little performance lag with our heuristics as compared to the ILP. Both algorithms can bring small and middle-sized networks close to perfect LFA-coverage by provisioning just a couple of virtual routers, and in larger backbones we see similar improved protection coverage just by provisioning roughly one virtual router per node on average.

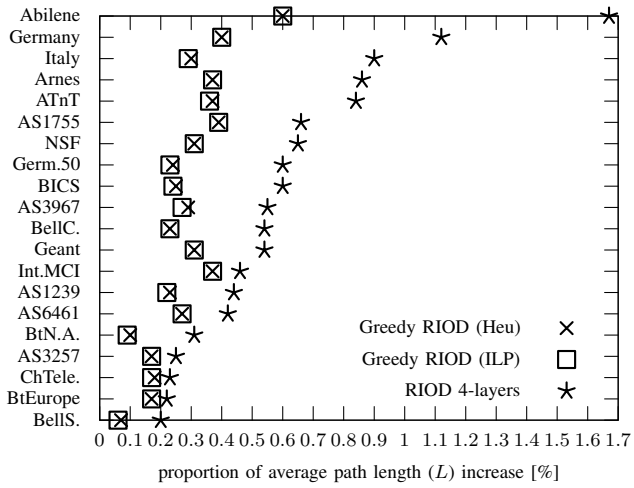


Figure 8. Average path stretch for single link failres.

VIII. Conclusions

With the advent of Seamless MPLS, Loop-Free Alternates for fast IP-level failure protection has become an indispensable tool in telecom networks. This is despite that LFA was not designed with carrier-grade requirements in mind, and therefore it does not provide out-of-the-box protection levels acceptable to most profit-oriented businesses.

In this paper, we invoked router virtualization, a commonplace feature in contemporary IP devices, to improve the level of protection provided by LFA. The motivation is to facilitate integrating existing operator infrastructure into modern multiservice MPLS/LDP networks without interfering with the normal operation of the network, or the network topology itself, in any ways. Our solution is deployable immediately with minimum management effort by establishing a Resilient IP Overlay on top of the physical network, which supplies "virtual LFAs" to unprotected node-pairs.

Even though the underlying optimization problem is NP-complete, practice shows that LFA virtual router augmentation can be efficiently solved by heuristics in polynomial time up to even ll LFA coverage. In extensive numerical evaluations we found that, depending on the extent to which SRLG support is available, practically complete protection can be attained against single link failures just by provisioning one or two virtual contexts at each IP router.

We again emphasize that this can be realized with existing IP hardware and software available in off-the-shelf routers today, with a one time management intervention. We believe that this finding opens the door for a wider adoption of Resilient IP Overlays in operational IP networks.

References

[1] M. Shand and S. Bryant, "IP Fast Reroute framework," RFC 5714, Jan 2010.

[2] Cisco Systems, "Cisco IOS XR Routing Configuration Guide, Release 4.2," 2016.

[3] Juniper Networks, "JUNOS 12.3 Routing protocols configuration guide," 2016.

[4] Hewlett-Packard, "HP 6600 Router Series: QuickSpecs," 2008.

[5] Huawei Technologies, "Huawei cx600 Metro Services Platform," 2011.

[6] Alcatel-Lucent, "Alcatel-Lucent 7750 Service Router Series," 2016.

[7] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-Free Alternates," RFC 5286, 2008.

[8] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "Loop-free alternate (lfa) applicability in service provider (sp) networks," IETF, RFC 6571, June 2012.

[9] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *ACM CoNEXT*, 2005, pp. 244–245.

[10] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *IEEE Comsware*, 2007.

[11] M. Menth, M. Hartmann, R. Martin, T. Čičić, and A. Kvalbein, "Loop-free alternates and not-via addresses: A proper combination for IP fast reroute?" *Comput. Netw.*, vol. 54, no. 8, pp. 1300–1315, 2010.

[12] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *INFOCOM*, 2004.

[13] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote loop-free alternate (lfa) fast reroute (frr)," IETF, RFC 7490, April 2015.

[14] A. Kvalbein, A. F. Hansen, T. Čičic, S. Gjessing, and O. Lysne, "Multiple routing configurations for fast IP network recovery," *IEEE/ACM Trans. Netw.*, vol. 17, no. 2, pp. 473–486, 2009.

[15] A. Li, X. Yang, and D. Wetherall, "SafeGuard: safe forwarding during route changes," in *ACM CoNEXT*, 2009, pp. 301–312.

[16] S. Bryant, S. Previdi, and M. Shand, "A framework for ip and mpls fast reroute using not-via addresses," IETF, RFC 6981, August 2013.

[17] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *ACM CoNEXT*, 2007.

[18] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "IP Fast ReRoute: Loop Free Alternates revisited," in *INFOCOM*, 2011, pp. 2948–2956.

[19] M. Nagy and G. Rétvári, "An evaluation of approximate network optimization methods for improving ip-level fast protection with loop-free alternates," in *Proc. of RNDM*, 2011.

[20] M. Menth, M. Hartmann, and D. Hock, "Routing optimization with IP Fast Reroute," Internet Draft, July 2010.

[21] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," in *Proc. DRCN*, 2011.

[22] S. Bryant, C. Filfils, , S. Previdi, M. Shand, and N. So, "Remote LFA FRR," Internet draft, 2013.

[23] S. Vissicchio, O. Tilmans, L. Vanbever, and J. Rexford, "Central control over distributed routing," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 43–56, 2015.

[24] C.-S. Li and W. Liao, "Software defined networks," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 113–113, 2013.

[25] A. Császár, G. Enyedi, G. Retvari, M. Hidell, and P. Sjödin, "Converging the evolution of router architectures and ip networks," *IEEE Network*, vol. 21, no. 4, pp. 8–14, 2007.

[26] M. Handley, "Why the Internet only just works," *BT Technology Journal*, vol. 24, no. 3, pp. 119–129, Jul. 2006.

[27] B. Fortz, J. Rexford, and M. Thorup, "Traffic engineering with traditional IP routing protocols," *IEEE Comm. Mag.*, vol. 40, no. 10, pp. 118–124, Oct 2002.

[28] G. Swallow, S. Bryant, and L. Andersson, "Avoiding equal cost multipath treatment in MPLS networks," RFC 4928, June 2007.

[29] M. Thorup and M. Roughan, "Avoiding ties in shortest path first routing," 2001, aT&T, Shannon Laboratory, Florham Park, NJ, Technical Report, http://www.research.att.com/~mthorup/PAPERS/ties_ospf.ps.

[30] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to rsvp-te for lsp tunnels," Internet Requests for Comments, IETF, RFC 4090, May 2005.

[31] J. Liu, A. Panda, A. Singla, B. Godfrey, M. Schapira, and S. Shenker, "Ensuring connectivity via dataplane mechanisms," in *Proc. NSDI*, 2013.

[32] S. Antonakopoulos, Y. Bejerano, and P. Koppol, "Full protection made easy: the dispath ip fast reroute scheme," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1229–1242, 2015.

[33] Z. Zhong, S. Nelakuditi, Y. Yu, S. Lee, J. Wang, and C.-N. Chuah, "Failure inferencing based fast rerouting for handling transient link and node failures," in *INFOCOM*, 2005.

[34] G. Enyedi, G. Rétvári, and T. Cinkler, "A novel loop-free IP fast reroute algorithm," in *EUNICE*, 2007.

[35] M. Chiesa, I. Nikolaevskiy, S. Mitrović, A. Gurtov, A. Mądry, M. Schapira, and S. Shenker, "On the resiliency of static forwarding tables," *IEEE/ACM Transactions on Networking*, 2016.

[36] I. Hokelek, M. Fecko, P. Gurung, S. Samtani, S. Cevher, and J. Sucec, "Loop-free IP Fast Reroute using local and remote LFAPs," Internet Draft, Feb 2008.

[37] T. Čičic, A. F. Hansen, and O. K. Apeland, "Redundant trees for fast IP recovery," in *Broadnets*, 2007, pp. 152–159.

[38] G. Enyedi, P. Szilágyi, G. Rétvári, and A. Császár, "IP Fast ReRoute: lightweight Not-Via without additional addresses," in *INFOCOM Miniconf*, 2009.

[39] K. Lakshminarayanan, M. Caesar, M. Rangan, T. Anderson, S. Shenker, and I. Stoica, "Achieving convergence-free routing using failure-carrying packets," in *Proc. SIGCOMM*, 2007.

[40] G. Schollmeier, J. Charzinski, A. Kirstadter, C. Reichert, K. Schrodi, Y. Glickman, and C. Winkler, "Improving the resilience in IP networks," in *Proc. of HPSR*, 2003, pp. 91–96.

[41] Cisco Systems, "Router virtualization in service providers," White Paper, 2008.

[42] Juniper Networks, "Control plane scaling and router virtualization," White Paper, 2010.

[43] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, no. 4, pp. 749–762, 2008.

[44] K. Kompella and Y. Rekhter, "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)," RFC 4203, Oct. 2005.

[45] ——, "IS-IS Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)," RFC 5307, Oct. 2008.

[46] M. Nagy, J. Tapolcai, and G. Rétvári, "On the design of resilient ip overlays," in *Proc. DRCN*, 2014, pp. 1–8.

[47] M. Garey, , and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[48] R. Balakrishnan and K. Ranganathan, *A textbook of graph theory*. Springer Science & Business Media, 2012.

[49] A. Itai and M. Rodeh, "The multi-tree approach to reliability in distributed networks," *Information and Computation*, vol. 79, no. 1, pp. 43–59, 1988.
[50] "LEMON – Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu/, 2009.
[51] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," http://www.topology-zoo.org.
[52] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *ACM IMC*, 2002, pp. 231–236.

APPENDIX

A. Complexity Proofs

*Definition 5: Minimum feedback arc set problem (*minFAS*, A1.1: GT8, p.192., [47]):* Given a digraph $G = (V, A)$ and a positive integer $k \leq |A|$. Find a subset $B \subseteq A$ with $|B| \leq k$ such that B contains at least one arc from every directed cycle in G.

Note that if B is removed from the graph, then all cycles are broken. Thus, minFAS asks for a minimal set of arcs which, when removed from the graph, leaves a DAG. The following problem is therefore equivalent to minFAS:

*Definition 6: Maximum spanning DAG (*maxDAG*):* Given a digraph $G = (V, A)$ and a positive integer $k \leq |A|$. Find a subset $B \subseteq A$ with $|B| \geq k$ such that the graph (V, B) is a DAG.

As minFAS is NP-complete, maxDAG is also NP-complete.

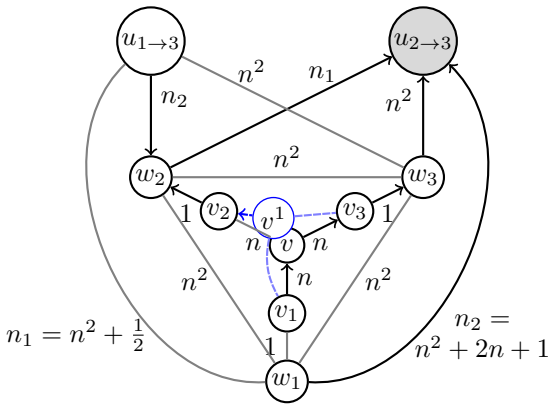
Proof of Theorem 2: LFAVirt(G_S, c_S, v, k) is in NP, since LFA coverage can be verified in polynomial time. To prove it is NP-hard, we (Karp)-reduce it to the maxDAG problem. Given a maxDAG instance with digraph $G_D = (V_D, A)$ and an integer k, we construct a LFAVirt(G_S, c_S, v, k') problem and we show that if, after adding a virtual node v^1 to v, there are k' LFA-protected (s, d) pairs, then the solution can be transformed to a solution to the maxDAG instance with k cardinality.

Next, we show how to construct G_S from G_D. G_S consists of $2n + |A| + 1$ nodes, where $n = |V_D|$. We add two nodes to G_S denoted by v_i and w_i for $i = 1, \dots n$, and for each arc $(i, j) \in A$, we assign a node in G_S, denoted by u_{ij}. For simpler arguments v_0 or w_0 refers to node v_n and w_n, respectively. Plus we have an additional node v. The edges of G_S are the followings.

- For each $i = 1, \dots n$ add an edge (v_i, w_i) with cost 1 and (v, v_i) with cost n. Node v is in fact the center of a star and all its neighbors have degree two.
- For each $i = 1, \dots n$ add an edge (w_i, w_{i-1}) of cost n^2.
- For each $l \in V_D$ and $(i, j) \in A$, add an edge (w_l, u_{ij}) with cost $c_V(w_j, u_{ij}) = n^2$, $c_V(w_i, u_{ij}) = n_1 = n^2 + \frac{1}{2}$, and otherwise $c_V(w_l, u_{ij}) = n_2 = n^2 + 2n + 1$ where $l \neq j$ and $l \neq i$.

Fig. 9 shows an example transformation with shortest paths. To destination $u_{i,j}$ every node has LFA except node v_j.

A node u becomes LFA protected after adding node v^1 towards destination d, if the following conditions hold: (i) u did not have LFA to d; (ii) u is adjacent to v, i.e. $u \in N_S(v)$; (iii) the next-hop of v^1 to d is a node v_i which is not the next-hop of v; and (iv) the next-hop of v_i is node w_i, and not v.



(a) The transformed graph G_S, The edges are directed towards the shortest path to node $u_{2\to3}$.

source node	distance	next-hop	LFA
w_i	$n^2 + 0.5$ t	$u_{i,j}$	w_{i-1}
w_j	n^2	$u_{i,j}$	w_{i-1}
$w_l : l \in V, l \neq i, l \neq j$	$n^2 + 2n + 1$	$u_{i,j}$	w_{i-1}
v_i	$n^2 + 1.5$	w_i	v
v_j	$n^2 + 1$	w_j	–
$v_l : l \in V, l \neq i, l \neq j$	$n^2 + 2n + 1$	v	w_l
v	$n^2 + n + 1$	v_j	v_i

(b) The shortest distance with next-hops to destination $u_{i\to j}$.

Figure 9. The transformed graph G_S, if G_D has 3 nodes a, s, and b, and two arcs $1 \to 3$ and $2 \to 3$.

The new LFAs created through v^1 are as follows. Node v_j becomes LFA protected to destination $u_{i,j}$, if the next-hop of v^1 is exactly v_i. This occurs if $n^2 + 1.5 + c_V(v_i, v^1) < n^2 + 1 + c_V(v_j, v^1)$, from which:

$$c_V(v_i, v^1) + \tfrac{1}{2} < c_V(v_j, v^1) \ , \tag{7}$$

and for $v_l : l \neq i, j$ condition $n^2 + 1.5 + c_V(v_i, v^1) < n^2 + 2n + 1 + c_V(v_l, v^1)$ also holds, so:

$$c_V(v_i, v^1) + \tfrac{1}{2} < 2n + c_V(v_l, v^1) \ . \tag{8}$$

For destinations w_j no new LFA is created, because the next-hop for every $v_i : i \neq j$ is v. Similar is the case for $v_j : j \in V$. As a summary, new LFAs can only appear between node pairs $v_j - u_{i,j} : (i, j) \in A$, and only if both (7) and (8) hold.

To conclude the proof we show that (i) if there is an LFAVirt$(G_S, c_S, v, k_S + k)$ solution, where k_S is the number of protected node pairs in G_S and k new LFAs are created by adding v^1, then there is a DAG of k links in G, and (ii) if there is a DAG of k links in G then there is an LFAVirt$(G_S, c_S, v, k_S + k)$ solution with k new LFAs.

For (i), suppose there is a cost assignment $c_V(v_i, v^1) : i \in V$ so that k new LFAs are created. Add an arc $(i, j) \in A$ to B if $c_V(v_i, v^1) + \frac{1}{2} < c_V(v_j, v^1)$. By (7) and (8), there are exactly k such arcs. Note that c_V is a topological order of the nodes in B. Thus, (V, B) is a DAG composed of k arcs.

For (ii), suppose there is a DAG of k arcs in G. Find a topological order of its nodes with ids $[1, n]$ and assign the order id of node i as $c_V(v_i, v^1)$. Clearly, v_j becomes LFA-protected to destination $u_{i,j}$ if link (i, j) is part of the DAG due to (7) and (8), so we have exactly k new LFAs. ■

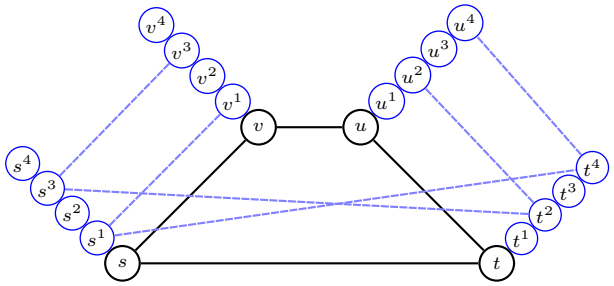


Figure 10. Construction for full coverage by adding $4|V_S|$ virtual nodes.

- The binary variable $x_n : n \in N_V(v)$ tells whether to provision the virtual link (v^k, n): $x_n = 1$ if (v^k, n) is a new virtual link, and zero otherwise.
- The binary variable $y_{s,d} : s, d \in \mathcal{L}$ marks whether $s \to d$ has obtained an LFA: $y_{s,d} = 1$ if $s \to d$ has LFA after adding v^k, and zero otherwise.
- The binary variable $z_{g,s,d} : s, d \in \mathcal{L}, g \in \mathcal{E}_{sd}$ is set so that $z_{g,s,d} = 1$ if g is the next-hop of v^k, zero otherwise.
- The non-negative real variable $c_n : n \in N_S(v)$ represents the cost $c_V(v^k, n)$ of the virtual link (v^k, n). We require that $c_n \geq c_S(v, n) + C$ where C is a problem parameter, to ensure that paths via v^k are longer than the default shortest paths. In the rest of this paper, we set $C = 1$.
- Finally, the non-negative real variable $\delta_u : u \in V_S \setminus \{v, v^k\}$ denotes the shortest path distance from v^k to u.

Consider the ILP below (the role of parameters K and ϵ will be made clear soon).

$$\max \sum_{s,d \in \mathcal{L}} y_{s,d} - \epsilon \sum_{n \in N_S(v)} (c_n + x_n) \tag{9}$$

$$y_{s,d} \leq x_s, \;\; z_{g,s,d} \leq \; x_g \qquad s, d \in \mathcal{L}_v, g \in \mathcal{E}_{sd} \tag{10}$$

$$y_{s,d} \leq \sum_{g \in \mathcal{E}_{sd}} z_{g,s,d} \qquad s, d \in \mathcal{L} \tag{11}$$

$$\delta_d \leq \mathrm{dist}(g, d) + c_g + K(1 - z_{g,s,d}) \;\; s, d \in \mathcal{L}_v, g \in \mathcal{E}_{sd} \tag{12}$$

$$\delta_d + K(1 - x_s) + K(1 - x_{s^i}) \geq \mathrm{dist}(g, d) + c_{s^i} + C \qquad s, d \in \mathcal{L}_v, i = 1, \ldots, s_k \tag{13}$$

$$c_n \geq c_S(v, n) + C \qquad n \in N_S(v) \tag{14}$$

$$x_n, y_{s,d}, z_{g,s,d} \in \{0, 1\}, \;\; c_n \geq 0 \tag{15}$$

The objective function (9) maximizes the number of LFAs the new virtual node v' gives rise to. Parameter ϵ is a small constant, which ensures that the optimization favors the solution with the smallest link costs and the fewest virtual links. The first constraint in (10) states that v' can only become an LFA for s if the virtual link (s, v') is present. Similarly, $z_{g,s,d} \leq x_g$ expresses that we can only set g as next-hop for v' if the virtual link (v', g) is provisioned.

Constraints (11) and (12) correspond to the escape node condition (5) for each $s \to d$ pair in $\mathcal{L}_v$. In particular, (12) will set the shortest path distance from v' to d according to whether the escape node $g \in \mathcal{E}_{sd}$ is chosen as the next-hop for v' to d. If $z_{g,s,d} = 0$, i.e., if g is not the next-hop then the constraint is inactive, while if $z_{g,s,d} = 1$ then the constraint is active and sets δ_d and c_g according to (5). To switch between the active and inactive states, we use the large constant $K \gg C + \max_{(s,d) \in V_S \times V_S} \mathrm{dist}(s, d)$. Furthermore, (11) sets an $s \to d$ pair protected, if at least one escape node has been selected as the next-hop for v' towards d.

Constraint (13) stands for the LFA-4 condition (6) for eligible $s \to d$ pairs. The constraint is only active when both (s, v') and (v', g) virtual links are present, i.e., $x_s = 1$ and $x_g = 1$. In this case, it sets c_g to prevent s^i to become a next-hop for v' to d according to (6) for $i = 1, \ldots, s_k$.

Finally, the domain of the variables is set in (14)–(15).

After solving the ILP, the virtual topology is constructed by augmenting the substrate with the virtual node v' and the virtual links $(v', n) : x_n = 1$ with cost c_n for all $n \in N_S(v)$.

B. Constructions

Proof of Theorem 4: According to Itai and Rodeh Theorem [49] an undirected graph $G = (V, E)$ with any edge $(s, t) \in E$ has an st-numbering. In st-numbering the nodes of the graph are assigned by distinct integers from 1 to n, denoted by $\mathcal{N}_v$ for node v, such that $\mathcal{N}_s = 1$, $\mathcal{N}_t = n$, and every other node v has a neighbor $u \in N_S(v)$ with smaller number ($\mathcal{N}_u < \mathcal{N}_v$), and a neighbor $w \in N_S(v)$ with larger number ($\mathcal{N}_w > \mathcal{N}_v$). With st-numbering we can define two trees T_1 and T_2 with the following properties:

- the nodes along path $T_1(s{-}v)$ has decreasing st-numbers,
- the nodes along path $T_2(v{-}t)$ has increasing st-numbers,
- $(s, t) \notin T_1$, and $(s, t) \notin T_2$.

This ensures that for any node v the paths $T_1(v - s)$ and $T_2(v - t)$ are edge disjoint.

In the construction, we add T_1 and T_2 to G as first and second virtual layers with distance K, and also for third and fourth virtual layers with distance $2K$. We also add virtual link (t^4, s^1) and (s^3, t^2) both with cost ε (see Fig. 10).

At node v let $v \to u$ be a next hop edge ($u \in \mathtt{nh}(v, d)$), we will use case-checking to show there is always an LFA.

If $(v, u) = (s, t)$ we have an LFA through the virtual node v^1 (and also v^2) according to Claim 1.

If $\mathcal{N}_v < \mathcal{N}_u \leq \mathcal{N}_d$ we have an LFA through v^3, which first traverses to node s^3 along the path $T_1(v^3 - s^3)$ and then through link (s^3, t^2) will reach node d along $T_1(s^2 - d^2)$. Note that this path is disjoint with $v \to u$, as the nodes in $T_1(v^3 - s^3)$ have numbers at most $\mathcal{N}_v$, and $T_1(s^2 - d^2)$ have numbers at least $\mathcal{N}_u$ because $\mathcal{N}_u \leq \mathcal{N}_d$.

If $\mathcal{N}_v > \mathcal{N}_u \geq \mathcal{N}_d$ for the above reason v^4 is an LFA.

If $\mathcal{N}_v < \mathcal{N}_u$ and $\mathcal{N}_u > \mathcal{N}_d$ we have an LFA through v^1, as the path $T_2(s^1 - d^1)$ is disjoint with $v \to u$ because the nodes along the path have numbers smaller than $\mathcal{N}_u$.

If $\mathcal{N}_v > \mathcal{N}_u$ and $\mathcal{N}_u < \mathcal{N}_d$ for the above reason v^2 is an LFA.

■

C. ILP for $\mathrm{LFAVirt}(G, c, w)$

The ILP is based on the idea that eligible node pairs and the respective escape nodes can be pre-computed statically, so $\mathcal{L}$, $\mathcal{L}_v$, and $\mathcal{E}_{sd}$ can be generated offline. Hence, in the course of the optimization we only need to take care of satisfying (5) and (6).

The variables of the ILP are as follows: