Router Virtualization for Improving IP-level Resilience

János Tapolcai, Gábor Rétvári
MTA-BME Future Internet Research Group, High-Speed Networks Laboratory (HSN*Lab*),
Dept. of Telecommunications and Media Informatics, Budapest University of Technology,
Email: {tapolcai, retvari}@tmit.bme.hu

***Abstract*—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. We study the problem of how to provision the overlay to maximize LFA coverage, we find that this problem is NP-complete, and we give Integer Linear Programs to solve it. We also propose novel methods to work-around the limitations of current LFA implementations concerning Shared Risk Link Groups (SRLGs), which might be of independent interest. Our numerical evaluations suggest that router virtualization is an efficient tool for improving LFA-based resilience in real topologies.**

***Index Terms*—IP Fast ReRoute, Loop-Free Alternates, router virtualization**

I. Introduction

Demand for native, fast protection in IP networks has increased lately. Failure recovery was traditionally a responsibility of Interior Gateway Protocols (IGPs), which adopt a restoration mechanism based on global failure notification, a robust but lengthy process. To bring routing convergence down to the orders of tens of milliseconds, as required by many applications and services today, the IETF has standardized a fast protection framework for IP, called IP Fast ReRoute (IPFRR, [1]). IPFRR, on the one hand, advocates *local rerouting*, meaning that only routers directly adjacent to a failure are notified about it in order to eliminate global flooding of failure information. On the other hand, IPFRR is *proactive* in that detours are installed in the forwarding engine long before any failure occurs. Thus, when a failure eventually shows up routers are able to switch to an alternate path instantly. Once alternate next-hops are active, traffic flows undisrupted bypassing the failed component, letting the IGP to converge in the background.

An accompanying basic specification for IPFRR is Loop-Free Alternates (LFA, [2]). In order to foster deployment, LFA is very simple. Routers try to assign to each next-hop an adjacent router as a secondary next-hop, called a Loop-Free Alternate, which can forward traffic when connectivity to the default next-hop is lost even without being explicitly notified about the failure. LFA can be implemented with minimal modifications to IGPs and deployment is easy as support from other routers is not required. Unfortunately, very often not all routers can find proper LFAs to all next-hops and so in general it is not possible to achieve 100% failure case coverage in all topologies with LFA. A relatively recent addition to the LFA suite is Remote LFA [3], which allows for non-adjacent routers as well to become LFAs via IP-IP tunnels or MPLS/LDP LSPs. Unfortunately, Remote LFA shares many of the limitations of LFA (although, to a much smaller extent), and hence in general 100% protection coverage is still not guaranteed [3].

Lately, several alternatives to LFA have been proposed to realize complete IP-level failure protection [4]–[13]. At the end of the day, however, price for complete protection is considerable added complexity and management burden, modifications to the essential IP infrastructure, and breaking the incremental deployment path. For instance, [4] proposes to change IP's destination-based forwarding, [5] calls for out-of-band failure signaling to indicate that a packet is on a detour, while others use invaluable extra bits in the IP header for this purpose [6] or add special information to it [7] for in-band signaling. Still others propose to mark detours by tunneling [8], raising considerable address management concerns [9], [10], and [11] necessitates a centralized control plane. For further pointers, consult the surveys [12], [13]. Currently, we do not see any of these IPFRR methods passing standardization bodies and becoming widely available in commercial IP routers (let alone being deployed by operators), and there does not seem to be much hope that this situation will change in the next couple of years. Operators, however, need fast IP-level protection *now*.

LFA, in contrast, is readily available in basically all commercial IP routers out of the box [14]–[16], and support for Remote LFA is also beginning to appear sporadically. What is more, *LFA has become a de-facto industrial requirement with the advent of Seamless MPLS*, an architecture to provide scalable end-to-end MPLS/LDP transport service based on existing IP protocols [17].

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 [18], [19] or, alternatively, re-engineer the default forwarding paths by re-computing the IGP link costs [20]. 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, and *building a virtual overlay topology especially tailored for LFA-based failure protection*. Conceptually, virtual routers are indistinguishable from physical routers, each instance having its own forwarding and control planes, which allows us to assign virtual routers as LFAs to routers that originally did not have one. Network virtualization was primarily introduced to overcome Internet ossification and increase diversity, to provide isolated environments for experimental protocols, and to improve utilization and security in provider networks, all in all, to let operators to extract larger profit from their valuable infrastructure [21]–[23]. Our work demonstrates that router virtualization can also be beneficial as a way to improve resilience in operational networks.

The main question is then 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 [24]–[26], intact. In addition, we also need to account for Shared Risk Link Groups (SRLGs), collections of network links likely to fail jointly due to virtualization, for which LFA currently has scarce support for. As the main contributions of the paper, we formulate the resultant network optimization problem in a concise mathematical framework, we find that the problem is NP-complete even in a very minimalistic setting, and we give Integer Linear Programs (ILPs) to solve it under different SRLG models. Furthermore, we present experimental evidence that the proposed techniques are efficient in improving LFA coverage in many common ISP topologies. In what follows, we concentrate mainly on LFA, with noting that the proposed techniques are easily adaptable to Remote LFA as well.

The paper is structured as follows. The formal framework is given in Section II and Section III, NP-hardness and the ILPs are discussed in Section IV, numerical results are given in Section V, and finally Section VI concludes the paper.

II. 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 $a - b - c - d$ of cost 7. However, when a's link to its next-hop b fails, a looses connectivity to d intermittently. In such cases, a is safe to send the packet to c as c still has

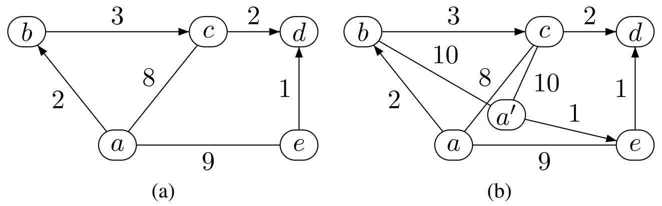


Fig. 1. Sample network and edge costs (a) and virtual topology (b) when adding the virtual router a' to a. The edges are directed to highlight the shortest path tree to node d.

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 [2]:

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

where $\operatorname{dist}(x, y)$ denotes the shortest path distance from node x to node y. An LFA is also *node-protecting* [2], if n's shortest paths to d are disjoint from t, i.e., the following holds in addition to (1):

$$\operatorname{dist}(n, d) < \operatorname{dist}(n, t) + \operatorname{dist}(t, d) \ . \qquad (2)$$

For instance, c is a node-protecting LFA from a to d w.r.t. next-hop b. When no ambiguity arises, we omit the default next-hop and simply say "c is an $a - d$ LFA". Similarly, e is a node-protecting $a - d$ LFA. However, a is not a $c - d$ LFA, because if c passed packets to a when (c, 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, c does not have an LFA to d in this configuration at all, leaving the network vulnerable to the failure of link (c, d).

In summary, a node does not have an LFA if all its neighbors except the next-hop are upstream. However, if we somehow 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' to a is depicted in Fig. 1b. Not just that a' is now a node-protecting LFA from c to d, but it also protects several more node pairs too that were unprotected in the default topology. In particular, a' provides LFA for $b-c$, $b-d$, $b-e$, $c-d$, and $c-e$, boosting LFA failure coverage from 50% to 75%. We emphasize that *the same effect could not have been achieved by layer-3 tunnels* (as of [3]), 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 [27], [28]. 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 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.

To understand how this can happen, consider the virtual topology depicted in Fig. 2. The physical topology consists of 6 nodes, a to f. Two virtual nodes are also provisioned: b' is added to b to create an $e-c$ LFA and c' is added to c to provide a $d-b$ LFA. These nodes, however, introduce an unexpected LFA loop: c' is now a $b-a$ LFA, but should b try to use this LFA in case of the failure of (b,a) the packet would get into the loop $b-c'-b'-e-b$. (Note that b', seeing its next-hop link (b',a) went down, also switches to the LFA e.) The reason is that the $c' \to a$ shortest path contains the virtual link (b',a) provisioned on the same physical link as (b,a), so the LFA and the link it protects fail jointly. Such an LFA will be called a *spurious LFA* henceforth.

Components that are expected to fail concertedly are called a *Shared Risk Link/Node Group* (SRLG). At the moment, LFA implementations in IGPs contain varying support for SRLGs, and hence it is implementation-dependent whether an IP router is able to identify this LFA loop. What is worse, in the physical topology b used to have a perfectly legal LFA to a, namely node f. Unfortunately, as there is no way to declare a precedence on LFAs it is possible that the spurious LFA is chosen, sending a packet into a loop or a blackhole.

Our example demonstrates that an inadvertent virtualization decision easily introduces spurious LFAs, and these can override legal LFAs. Correspondingly, when building an LFA-optimized overlay one must take extreme care of the SRLGs introduced through virtualization.

III. Notations and model

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.

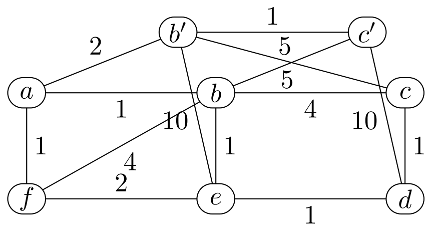


Fig. 2. Virtual network containing an LFA loop.

A. 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 $c_S : E_S \mapsto \mathbb{N}$. 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 $c_V : E_V \mapsto \mathbb{N}$ in a way as to maximize LFA coverage in G_V. Nodes in G_V are called *virtual nodes* or contexts and links are called *virtual links*. 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.

There is a default routing and forwarding layer. 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. Mark the default context for a physical router $v \in V_S$ by the same $v \in V_V$. Similarly, for each edge $e \in E_S$ there is a default virtual link $e \in E_V$ and $\forall e \in E_S : c_V(e) = c_S(e)$. In fact, G_S is a subgraph of G_V in our model. We call this subgraph the default layer. Let $f_N : V_V \mapsto V_S$ be a mapping which to each $v \in V_V \setminus V_S$ orders the corresponding default context and to the default context orders itself.

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 $(u,v) \in V_S \times V_S$.

There is a default next-hop for each source-destination pair. For the sake of notational brevity, we shall deliberately ignore Equal-Cost MultiPath and assume that for each $s-d$ pair there is a well-defined default next-hop $\mathrm{nh}(s,d)$. In addition, we presume that a proper LFA only needs to be available to protect this default next-hop. We note, however, that multiple next-hops are easy to incorporate into the proposed algorithms.

Virtual links connect physically connected nodes. Virtual links are provisioned between nodes that are adjacent in the substrate: $\forall(i,j) \in E_V : (f_N(i), f_N(j)) \in E_S$. 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 [9], [10]. 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.

B. Failure and SRLG model

Single component failures in the physical network. We assume single link/node failures, which constitute the major portion of unplanned outages in operational networks [29]. However, a single link failure in the physical network 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 same applies to nodes. To represent this fact in our model, we define the following SRLGs. Associated with each link $e \in E_V$ there is an SRLG $S_E(e)$ composed of all the links provisioned on the same physical link as e. If e fails, all links in $S_E(e)$ fail as well. Similarly, for each node $v \in V_V$ there is an SRLG $S_N(v)$ that consists of all the nodes co-located with v. Again, if v fails all nodes in $S_N(v)$ fail.

SRLG support varies across implementations. The LFA specification introduces *local SRLGs* as the minimum requirement for conforming implementations [2]. A local SRLG has all its member links with one end connected to the same router. Thus, associated with each $v \in V_V$ and each $e \in E_V$ incident to v, there is a local SRLG containing all the edges incident to v sharing an SRLG with e: $\{(i,j) \in S_E(e) : i = v \vee j = v\}$. A conforming IGP then will never install an LFA through a link that shares a local SRLG with the primary next-hop. Local SRLG support is easy to deploy as it does not need network-wide configuration and dissemination mechanisms [30], [31], but it is also quite limited in that routers will only spot non-SRLG-disjoint paths at the first hop. There is no such restriction when the IGP supports *general SRLGs*, but this needs global management and it may also add unacceptably to the computational complexity of finding LFAs [3]. Finally, there are IGP implementations that do not support SRLGs at all. We call this the *no-SRLG* model. According to our best available data IGPs today either do not support SRLGs or only support local SRLGs, but no implementation we know of features general SRLG support. Consequently, we shall present specific algorithms for the former two cases only, and we shall not address general SRLGs at all in the sequel.

C. SRLG-disjoint LFAs

LFAs obey the IGP's SRLG model. Ignoring SRLGs in the course of building the virtual topology might introduce microloops or blackholes. To eliminate such cases in our model, we shall use the following SRLG-compliant LFA definition in the followings.

Definition 1: Let $(s,d) \in V_S \times V_S$ be node pair, let $t = \mathrm{nh}(s,d)$, and let n be some neighbor of s. Then, n is an *SRLG-disjoint $s-d$ LFA protecting link* $e = (s,t)$, if:
LFA-1: $n \neq t$,
LFA-2: the loop-free condition (1) holds for s, d, and n,
LFA-3: e and (s,n) do not belong to a common SRLG, and
LFA-4: each $n \to d$ shortest path is SRLG-disjoint from e.

Full packet tracing for detecting forwarding anomalies. As demonstrated by the example in Fig. 2, one must trace the procession of a packet hop-by-hop from the source node all the way to the destination in order to reliably identify LFA loops. The tracing procedure must also keep in sight the cases when the packet reaches its destination, or enters a blackhole, through a cascade of LFAs. We shall assume that a proper packet tracing procedure is available to the network designer. Using this procedure, 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 a packet sent from s would reach the destination d in G_V over the link costs c_V (this is checked by the tracing procedure), and zero otherwise. Note that $I_E(v,v) = 1$. Then, the *link-protecting LFA coverage* is defined as

$$\eta_E = \frac{\sum_{(s,d)\in V_S\times V_S} I_E(s,d)}{|V_S \times V_S|} \quad .$$

Due to space constraints, we focus on the link-protecting case in the rest of the paper. The development for the node-protecting case goes along similar lines, with the difference that we require (2) in addition to (1) for LFA-2 in Definition 1.

IV. The LFA virtual router augmentation problem

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. Since SRLG support varies widely across commercially available routers, we give a specific algorithm for the case when only local SRLGs can be set, and one when no SRLGs can be used at all.

The ultimate goal would be to build an entire virtual topology in one step so that each node-pair in the substrate becomes LFA-protected. Instead, in this paper we solve the somewhat less ambitious task to add a *single virtual router to a known node*, and our objective is merely *to maximize LFA coverage* along the way instead of aiming for full coverage. This problem will be referred to as the *LFA virtual router augmentation problem* (LFAVirt). As shall be shown, 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.

Informally, in LFAVirt we are given the substrate G_S and some node v in G_S, and our task is to build a virtual topology G_V by adding a virtual node v' to v and setting the link costs so that default shortest paths remain the same while LFA coverage increases the most. The formal description of LFAVirt for the link-protecting case is as follows.

Definition 2: $\mathrm{LFAVirt}(G_S, c_S, v, k)$: Given a substrate $G_S(V_S, E_S)$ with link costs c_S, a positive integer k, and a node $v \in V_S$, is there a graph $G_V(V_V, E_V) : V_V = V_S \cup \{v'\}$, $E_V \subseteq E_S \cup \{(v', u) : u \in N_S(v)\}$ and a cost setting c_V on G_V, so that the link costs and shortest paths in G_S do not change and $\sum_{(s,d)\in V_S\times V_S} I_E(s, d) \geq k$?

Below, we shall solve the optimization version where we ask for the highest k for which LFAVirt answers affirmative. The version where we ask for adding a virtual node to an arbitrary router (instead of a known one) is solvable by doing this optimization for each $v \in V_S$ and taking the maximum.

A. Complexity analysis

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

Theorem 1: $\mathrm{LFAVirt}(G_S, c_S, v, k)$ is NP-complete under any SRLG model.

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 determining c_V, which is then shown to solve arbitrary instances of the *minimum feedback arc set* problem, a well-known NP-complete problem [32, GT8, pg.192.]. A crucial property of the proof is that it does not involve SRLGs at all, and so it remains valid under basically any SRLG model. For the complete proof, refer to the Appendix.

B. Solving LFAVirt *under local SRLG support*

The LFA specification requires local SRLG support from implementations [2]. Below, we ask to what extent this model allows to improve LFA coverage in virtual topologies.

Under the local SRLG model, the IGP drops LFAs that share the first-hop link with the default next-hop, but it can not look "further" to determine whether the complete path from the LFA to the destination is SRLG disjoint. In terms of Definition 1, the IGP will be able to decide on LFA-1, LFA-2, and LFA-3, but it will not be able to take LFA-4 into account. Therefore, in the course of building the virtual topology *we must guarantee ourselves that spurious LFAs are never provisioned*, not even accidentally. The way we achieve this is that to use the packet tracing procedure to check whether packets would be sent into an LFA loop or blackhole and explicitly disallow such LFAs to be provisioned.

Below, we show an Integer Linear Program (ILP) to solve the optimization version of LFAVirt under the local SRLG model. In fact, we solve a somewhat more complex version as we allow the substrate to already contain virtual nodes and links with the corresponding SRLGs. This allows to incorporate the ILP into a greedy optimization framework for building complete LFA-optimized overlays. In every iteration we add the node that maximizes LFA coverage, taking the virtual topology from the previous iteration as the substrate.

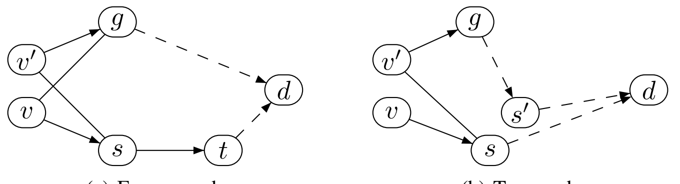


Fig. 3. Illustrations for escape nodes and trap nodes. Dashed arrows mark shortest paths.

First, we characterize the source-destination pairs that can gain an LFA when a new virtual node v' is added to the physical node v. Clearly, an $s-d$ pair can obtain an LFA only if the new virtual node v' is adjacent to s and the (s, v') link is local-SRLG-disjoint from the $(s, \mathrm{nh}(s, d))$ link (otherwise, the IGP will drop the LFA). Correspondingly, v' is eligible as a new LFA for some $s - d$ pair if:

- s is adjacent to v',
- $s - d$ is unprotected, and
- the links $(s, \mathrm{nh}(s, d))$ and (s, v') are local-SRLG-disjoint, i.e., $\mathrm{nh}(s, d) \neq v$.

Denote the set of such $s - d$ pairs with $\mathcal{L}$. Then, for each $s-d$ pair in $\mathcal{L}$ we seek the nodes that can provide an SRLG-disjoint LFA to $s - d$ through v'. Let $t = \mathrm{nh}(s, d)$ and call a g node an *escape node* if

- $g \in N_V(v)$ and
- a packet sent from g would reach d when link (s, t) fails.

Here, the second condition is verified by the packet tracing procedure (see Fig. 3a). For each $s, d \in \mathcal{L}$, let $\mathcal{E}_{sd}$ denote the set of escape nodes for $s - d$.

Easily, 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 $s - d$. For this, we need to set link costs c_V such that

$$\mathrm{dist}(v', d) = c_V(v', g) + \mathrm{dist}(g, d) \text{ for some } g \in \mathcal{E}_{sd} \ . \quad (3)$$

Next, we characterize the cases when the new virtual node v' may, under an inauspicious selection of the link costs, give rise to a spurious LFA. An $s - d$ pair is said to be *critical*, if

- s is adjacent to v', and
- $\mathrm{nh}(s, d) \neq v$.

Let $\mathcal{Q}$ denote the set of such critical $s - d$ pairs. For any critical $s - d$ pair, call a node $g \in N_V(v)$ a *trap node* if a packet sent from g would *not* reach d when $(s, \mathrm{nh}(s, d))$ fails (see Fig. 3b). Trap nodes are the direct opposite of escape nodes, as a trap node, if chosen as a next-hop for v', would create a spurious LFA. Mark the set of trap nodes for $s - d$ by $\mathcal{T}_{sd}$.

The below condition, if holds for each critical $s - d$ pair, ensures that no trap node will be chosen as a next-hop for v':

$$\mathrm{dist}(v', d) < c_V(v', g) + \mathrm{dist}(g, d) \text{ for all } g \in \mathcal{T}_{sd} \ . \quad (4)$$

The ILP is based on the idea that eligible and critical node pairs and the respective escape nodes and trap nodes can be pre-computed statically, so $\mathcal{L}$, $\mathcal{Q}$, $\mathcal{E}_{sd}$, and $\mathcal{T}_{sd}$ can be generated offline. Hence, in the course of the optimization we only need to take care of satisfying (3) and (4).

The variables of the ILP are as follows:

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

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

$$\max \sum_{s,d \in \mathcal{L}} y_{s,d} - \epsilon \sum_{n \in N_S(v)} (c_n + x_n) \quad (5)$$

$$y_{s,d} \leq x_s, \; z_{g,s,d} \leq x_g \qquad s, d \in \mathcal{L}, g \in \mathcal{E}_{sd} \quad (6)$$

$$y_{s,d} \leq \sum_{g \in \mathcal{E}_{sd}} z_{g,s,d} \qquad s, d \in \mathcal{L} \quad (7)$$

$$\delta_d \leq \mathrm{dist}(g, d) + c_g + K(1 - z_{g,s,d}) \quad s, d \in \mathcal{L}, g \in \mathcal{E}_{sd} \quad (8)$$

$$\delta_d + K(1 - x_s) + K(1 - x_g) \geq \mathrm{dist}(g, d) + c_g + C \qquad s, d \in \mathcal{Q}, g \in \mathcal{T}_{sd} \quad (9)$$

$$c_n \geq c_S(v, n) + C \qquad n \in N_S(v) \quad (10)$$

$$x_n, y_{s,d}, z_{g,s,d} \in \{0, 1\}, \; c_n \geq 0 \quad (11)$$

The objective function (5) 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 (6) 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 (7) and (8) correspond to the escape node condition (3) for each $s - d$ pair in $\mathcal{L}$. In particular, (8) 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 (3). 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, (7) sets an $s - d$ pair protected, if at least one escape node has been selected as the next-hop for v' towards d.

Constraint (9) stands for the trap node condition (4) for critical $s - d$ pairs. The constraint is only active when both (s, v') and (v', g) virtual links are present for some trap node $g \in \mathcal{T}_{sd}$, i.e., $x_s = 1$ and $x_g = 1$. In this case, it sets c_g to prevent g to become a next-hop for v' to d according to (4).

Finally, the domain of the variables is set in (10)–(11).

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 $n \in N_S(v)$.

C. *Solving* LFAVirt *when no SRLG support is available*

Some LFA implementations do not support SRLGs at all, not even local SRLGs. In this case, the IGP will only be able to check LFA-1 and LFA-2, but will not take LFA-3 and LFA-4 into account. Under this model, *all LFAs provisioned through virtual nodes must be local-SRLG-disjoint as well*, because the IGP will not be able to tell these cases apart.

To incorporate this restricted SRLG model into the ILP, we need to broaden the definition of critical node pairs and trap nodes somewhat. Specifically, for some node pair $s - d$ in $N_s(v) \times V_S$ and some node $g \in N_V(v)$, if either

- $\mathrm{nh}(s, d)$ and v' are *not* local-SRLG-disjoint or
- packets sent from g would not reach d when the link from s to $\mathrm{nh}(s, d)$ fails,

then $s - d$ is added to the set of critical node pairs $\mathcal{Q}$ and g is set as a trap node for $s - d$. With this definition in place, the ILP is solved and the overlay is provisioned as before.

V. Numerical evaluations

In the course of our numerical studies, we asked to what extent LFA virtual router augmentation allows to improve LFA coverage in real-world networks and how differing SRLG support in IGPs affects the results? To answer these questions, we implemented the ILPs for the local SRLG and the no-SRLG model, and we integrated these into a greedy virtual network optimization strategy where in each iteration we add the virtual router that maximizes LFA coverage.

We selected common-place ISP topologies from the Rocketfuel dataset [35], SNDlib [33], and the Topology-Zoo project's dataset [34]. We applied the standard preprocessing techniques to obtain approximate POP-level network maps, in that we collapsed the topologies so that nodes correspond to cities and we eliminated leaf-nodes. The topologies in the Rocketfuel data set (AS1239, AS1755, AS3257, AS3967, and AS6461) come with inferred link costs. For the topologies in the Topology-Zoo dataset (Arnes, BellSouth, BellCanada, BICS, BtEurope, BtNAmerica, ChinaTelecom, Deltacom, Geant, and InternetMCI) we set the link costs inversely proportional to the link capacity whenever capacities were available and we used random costs otherwise. Unfortunately, no link costs or capacities were available in the SNDlib library (Abilene, Italy, Germany, NSF, AT&T, and the extended German backbone Germ_50) except for the last one, so here we used unit costs. We used CPLEX [36] for solving the ILPs on a server with 3GHz Intel Xeon CPU 5160.

In the first round of simulations, we observed the LFA coverage after adding an increasing number of virtual routers to the network under the local-SRLG model. The results are given in Table I. Here, $\eta_E(t)$ marks the link-protecting LFA

TABLE I
TOPOLOGIES, LINK-PROTECTING, AND NODE-PROTECTING LFA COVERAGE WHEN ADDING AN INCREASING NUMBER OF VIRTUAL NODES UNDER THE LOCAL-SRLG MODEL: NAME, NUMBER OF NODES n, NUMBER OF LINKS m FOR EACH TOPOLOGY, $\eta_E(t)$ UNDER SINGLE LINK FAILURES, AND $\eta_N(t)$ UNDER SINGLE LINK AND NODE FAILURES, AND AVERAGE EXECUTION TIME OF ONE ITERATION OF THE ALGORITHM IN SECONDS.

			single link failures						single link and node failures					
Name	n	m	$\eta_E(0)$	$\eta_E(\frac{1}{3})$	$\eta_E(\frac{2}{3})$	$\eta_E(1)$	$\eta_E(2)$	time [s]	$\eta_N(0)$	$\eta_N(\frac{1}{3})$	$\eta_N(\frac{2}{3})$	$\eta_N(1)$	$\eta_N(2)$	time [s]
Abilene [33]	11	14	0.618	0.772	0.900	0.963	1.000	0.014	0.572	0.700	0.781	0.827	0.909	0.040
Germany [33]	17	25	0.694	0.886	0.944	0.981	1.000	0.025	0.562	0.694	0.790	0.838	0.897	0.109
BtEurope [34]	17	30	0.966	0.988	0.988	0.988	0.988	0.120	0.577	0.823	0.922	0.922	0.922	0.433
AS6461 [35]	17	37	0.933	0.996	0.996	0.996	0.996	0.072	0.757	0.941	0.977	0.977	0.977	0.350
InternetMCI [34]	18	32	0.954	0.986	1.000	1.000	1.000	0.022	0.643	0.803	0.898	0.941	0.977	0.216
AS1755 [35]	18	33	0.872	0.983	1.000	1.000	1.000	0.027	0.709	0.882	0.957	0.980	0.980	0.221
ChinaTelecom [34]	20	44	0.950	0.994	0.994	0.994	0.994	0.170	0.768	1.000	1.000	1.000	1.000	0.217
AS3967 [35]	21	36	0.785	0.983	1.000	1.000	1.000	0.052	0.688	0.840	0.914	0.950	0.976	0.278
BellSouth [34]	21	36	0.797	0.997	1.000	1.000	1.000	0.043	0.614	0.957	1.000	1.000	1.000	0.188
AT&T [33]	22	38	0.822	0.963	0.993	1.000	1.000	0.050	0.683	0.798	0.796	0.796	0.796	0.586
NSF [33]	26	43	0.860	0.958	0.993	1.000	1.000	0.065	0.633	0.801	0.890	0.943	0.986	0.344
BICS [34]	27	42	0.764	0.948	0.982	0.988	0.988	0.068	0.578	0.722	0.810	0.856	0.904	0.361
AS3257 [35]	27	64	0.923	1.000	1.000	1.000	1.000	0.081	0.682	0.948	0.988	0.988	0.988	1.924
AS1239 [35]	30	69	0.873	0.995	1.000	1.000	1.000	0.091	0.675	0.890	0.916	0.916	0.916	3.882
Arnes [34]	31	47	0.830	0.982	0.994	0.997	0.997	0.097	0.740	0.916	0.939	0.939	0.939	0.699
Geant [34]	31	49	0.829	0.983	0.994	0.995	0.995	0.105	0.569	0.858	0.919	0.943	0.951	0.601
Italy [33]	33	56	0.784	0.923	0.969	0.982	0.985	0.170	0.626	0.810	0.813	0.813	0.813	3.689
BtNAmerica [34]	36	76	0.831	0.987	0.998	0.998	0.998	0.222	0.741	0.944	0.972	0.972	0.972	0.757
BellCanada [34]	39	55	0.614	0.852	0.965	0.983	0.985	0.118	0.477	0.693	0.788	0.809	0.849	0.329
Germ_50 [33]	50	88	0.900	0.982	0.997	0.998	0.998	0.269	0.827	0.917	0.946	0.959	0.964	0.833
Deltacom [34]	103	151	0.632	0.906	0.951	0.954	0.954	1.159	0.536	0.785	0.851	0.854	0.854	2.286

coverage η_E after provisioning t times n virtual nodes in the network, where n denotes the number of nodes in the substrate. So $\eta_E(0)$ gives the initial LFA coverage in the physical network, $\eta_E(1)$ gives the LFA coverage when on average 1 virtual router is provisioned at each physical router, etc. We also ran the evaluations for the node-protecting case. The interpretation of the node-protecting LFA coverage $\eta_N(t)$ is similar. Table I also highlights some details for the topologies (number of nodes n and number of links m in the substrate) and it also gives the average execution time of a single iteration of the greedy algorithm in seconds (i.e., the time needed to solve the ILP for each node in the network and choosing the virtual node that maximizes LFA coverage).

The most important observations are as follows.

First, it seems that router virtualization is indeed useful in increasing the level of fast protection feasible with LFA. For single link failures, from an initial 65-85% LFA coverage adding a virtual node to just 33% of the physical routers improves LFA coverage beyond 90% in most of the cases, an additional 33% nodes improve this to more than 95%, while two virtual nodes per router gives almost perfect (beyond 99%) link protection. The improvement is in the range of 10-30%. The results are similar for node-protection case as well, with the difference that the final LFA coverage is smaller but the improvement is more significant (40-50% in many cases).

Surprisingly, the ILPs turned out much easier to solve than we expected. A major reason for this is that the ILP instances themselves are not particularly large: even though the formulation (5)–(11) seems daunting, the number of binary variables remains in the range of $\Delta^2 n$, where Δ is the average node degree and n is the number of nodes, and so for most of the cases the ILPs contain only a couple of dozen variables and columns. It seems, therefore, that the intractability result proven in Theorem 1 rarely manifests itself on real instances.

In the second round of evaluations, we asked to what extent the SRLG model (local SRLG or no-SRLG) affects the results. We were also curious about the robustness of the algorithms against the initial link costs in the substrate, so we reran the evaluations for both SRLG models 20 times, each time over freshly generated random initial costs. The mean values and the confidence intervals near 95% significance for the link-protecting and node-protecting LFA coverage are given in Fig. 4 for some select topologies.

It looks that the no-SRLG model is overly restrictive, in that it admits only a 2-10% initial improvement in the LFA coverage and then it quickly goes into saturation. We also find that the results are pretty robust against IGP costs[1].

VI. 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 multi-service MPLS/LDP networks without interfering with the normal operation of the network, or the network topology itself, in any ways. As far as we know, this is the first time that such a solution is proposed. What is more, our solution is deployable immediately with minimum management effort.

[1]See the web demo at http://opti.tmit.bme.hu/~tapolcai/demo/?lfa-vn

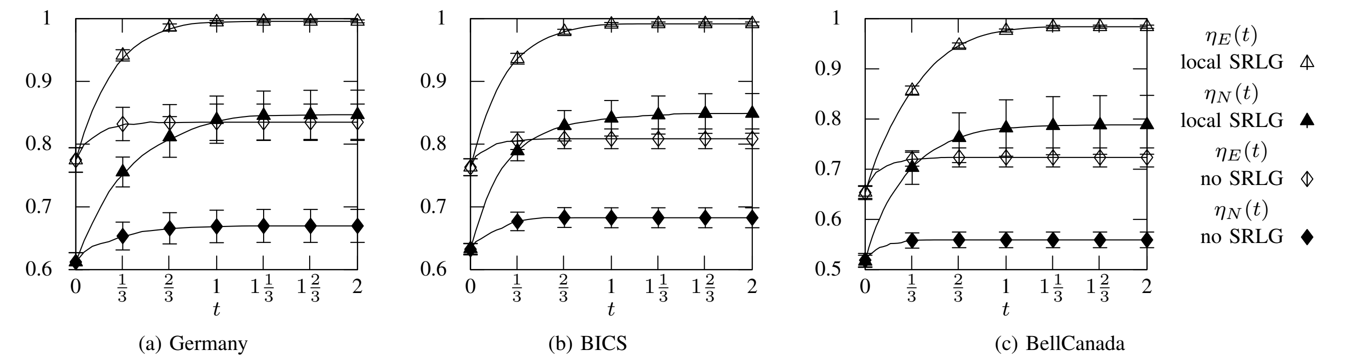


Fig. 4. Link-protecting $\eta_E(t)$ and node-protecting $\eta_N(t)$ LFA coverage in different SRLG models in select topologies when adding an increasing number of virtual nodes.

Even though the underlying optimization problem is NP-complete, practice shows that LFA virtual router augmentation lends itself readily to be solved as an Integer Linear Program. 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. For node failures, LFA coverage is in the range 90-98%. 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 found that failure coverage greatly depends on the SRLG support available in the IGP, and our techniques are most powerful when the IGP supports local SRLGs. We believe that this observation, coupled with the fact that it is very easy to implement and deploy, can be a strong motivation for vendors to add local SRLG support to their IP products if they haven't done that yet.

VII. Acknowledgments

The authors wish to thank András Császár, Gábor Enyedi, Máté Nagy and Levente Csikor for their valuable comments and help. The research was partially supported by the project TÁMOP - 4.2.2.B-10/1–2010-0009, and the OTKA/PD-104939 grant.

References

[1] M. Shand and S. Bryant, "IP Fast Reroute framework," RFC 5714, Jan 2010.
[2] A. Atlas and A. Zinin, "Basic specification for IP Fast Reroute: Loop-Free Alternates," RFC 5286, 2008.
[3] S. Bryant, C. Filfils, M. Shand, and N. So, "Remote LFA FRR," Internet Draft, jun 2012.
[4] S. Lee, Y. Yu, S. Nelakuditi, Z.-L. Zhang, and C.-N. Chuah, "Proactive vs reactive approaches to failure resilient routing," in *INFOCOM*, 2004.
[5] 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.
[6] 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.
[7] A. Li, X. Yang, and D. Wetherall, "SafeGuard: safe forwarding during route changes," in *ACM CoNEXT*, 2009, pp. 301–312.
[8] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using Not-via addresses," Internet Draft, March 2010.
[9] A. Li, P. Francois, and X. Yang, "On improving the efficiency and manageability of NotVia," in *ACM CoNEXT*, 2007.
[10] 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.
[11] K.-W. Kwong, L. Gao, R. Guerin, and Z.-L. Zhang, "On the feasibility and efficacy of protection routing in IP networks," in *INFOCOM*, 2010.
[12] P. Francois and O. Bonaventure, "An evaluation of IP-based fast reroute techniques," in *ACM CoNEXT*, 2005, pp. 244–245.
[13] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *IEEE Comsware*, 2007.
[14] Cisco Systems, "Cisco IOS XR Routing Configuration Guide, Release 3.7," 2008.
[15] Juniper Networks, "JUNOS 9.6 Routing protocols configuration guide," 2009.
[16] Hewlett-Packard, "HP 6600 Router Series: QuickSpecs," 2008, available online: http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF.
[17] N. Leymann, B. Decraene, C. Filsfils, and M. Konstantynowicz, "Seamless MPLS architecture," Internet Draft, March 2011.
[18] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "LFA applicability in SP networks," Internet Draft, March 2010.
[19] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, "IP Fast ReRoute: Loop Free Alternates revisited," in *INFOCOM*, 2011, pp. 2948–2956.
[20] M. Menth, M. Hartmann, and D. Hock, "Routing optimization with IP Fast Reroute," Internet Draft, July 2010.
[21] J. Turner and D. Taylor, "Diversifying the Internet," in *GLOBECOM*, vol. 2, 2005.
[22] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, pp. 34–41, April 2005.
[23] N. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Comput. Netw.*, vol. 54, pp. 862–876, April 2010.
[24] 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.
[25] G. Swallow, S. Bryant, and L. Andersson, "Avoiding equal cost multipath treatment in MPLS networks," RFC 4928, June 2007.
[26] 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.
[27] Cisco Systems, "Router virtualization in service providers," available online: http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns562/ns573/white_paper_c11-512753.pdf, 2008.
[28] Juniper Networks, "Control plane scaling and router virtualization," available online: http://www.juniper.net/us/en/local/pdf/whitepapers/2000261-en.pdf, 2010.
[29] 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.
[30] K. Kompella and Y. Rekhter, "OSPF Extensions in Support of Generalized Multi-Protocol Label Switching (GMPLS)," RFC 4203, Oct. 2005.

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

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

[33] SNDlib, "Survivable fixed telecommunication network design library," http://sndlib.zib.de.

[34] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet Topology Zoo," http://www.topology-zoo.org.

[35] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, "Inferring link weights using end-to-end measurements," in *ACM IMC*, 2002, pp. 231–236.

[36] ILOG, Inc, "ILOG CPLEX: High-performance software for mathematical programming and optimization," 2006, see http://www.ilog.com/products/cplex/.

Appendix

*Definition 3: Minimum feedback arc set problem (*minFAS*, A1.1: GT8, p.192., [32]):* 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 4: 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.

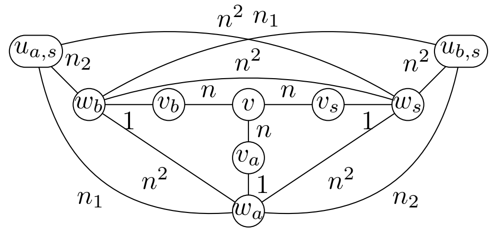
As minFAS is NP-complete, maxDAG is also NP-complete.

Proof of Theorem 1: $\mathrm{LFAVirt}(G_S, c_S, v, k)$ 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 $G = (V, A)$ and an integer k, we construct a $\mathrm{LFAVirt}(G_S, c_S, v, k')$ problem and we show that if, after adding virtual node v' 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.

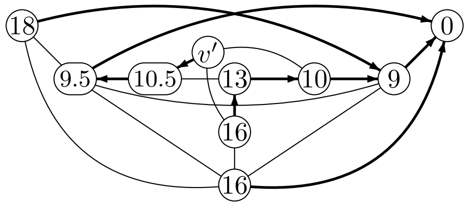
Next, we show how to construct G_S from G. G_S consists of $2|V| + |A| + 1$ nodes. For each arc $(i, j) \in A$, we assign a node in G_S, denoted by u_{ij}, and for each node i in V we add two nodes to G_S denoted by v_i and w_i. Plus we have an additional node v. The edges of G_S are the follows.

- For each $i \in V$ add an edge (v_i, w_i) with cost 1 and (v, v_i) with cost n, where $n = |V|$. Node v is in fact the center of a star and all its neighbors have degree two.
- For each $i \in V$ add an edge (w_i, w_{i+1}) of cost n^2, where i is the id of each node from $0, \ldots, n-1$ and addition is modulo n.
- For each $l \in V$ and $(i, j) \in A$, add an edge (w_l, u_{ij}). If $l = j$ then set the cost to n^2, if $l = i$ then set cost to $n_1 = n^2 + \frac{1}{2}$, otherwise set cost to $n_2 = n^2 + 2n + 1$.

The shortest paths are as follows. To destination $u_{i,j}$ nodes $w_k : k \in V$ have $u_{i,j}$ as the next-hop, node v_i and v_j has next-hop w_i and w_j, respectively, while all the other $v_l : l \in A$ have v as next-hop (see Fig. 5). Finally, the next-hop of node v is v_j. A node u becomes LFA protected after adding node v' 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' 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, if G has 3 nodes a, s, and b, and two arcs (a, s) and (b, s).



(b) The shortest path tree to node u_{bs}. The distances are written as node labels, where $n = 3$, $n^2 = 9$, $n_1 = 9.5$, and $n_2 = 16$.

Fig. 5. The transformed graph.

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

$$c_V(v_i, v') + \tfrac{1}{2} < c_V(v_j, v') \ , \tag{12}$$

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

$$c_V(v_i, v') + \tfrac{1}{2} < 2n + c_V(v_l, v') \ . \tag{13}$$

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 (12) and (13) hold.

To conclude the proof we show that (1) if there is an $\mathrm{LFAVirt}(G_S, c_S, v, k_S + k)$ solution, where k_S is the number of protected node pairs in G_S and k new LFAs are created by adding v', then there is a DAG of k links in G, and (2) if there is a DAG of k links in G then there is an $\mathrm{LFAVirt}(G_S, c_S, v, k_S + k)$ solution with k new LFAs.

For (1), suppose there is a cost assignment $c_V(v_i, v') : i \in V$ so that k new LFAs are created. Without loss of generality, $c_V(v_i, v')$ are integer in the range $[1, n]$. Add an arc $(i, j) \in A$ to B if $c_V(v_i, v') < c_V(v_j, v')$. By (12) and (13), there are exactly k such arcs and c_V is a topological order of the arcs in B. Thus, (V, B) is a DAG of k arcs.

For (2), 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 $c_V(v_i, v')$. Clearly, v_j becomes LFA-protected to destination $u_{i,j}$ if link (i, j) is part of the DAG due to (12) and (13), so we have exactly k new LFAs. ■