On the Design of Resilient IP Overlays

Máté Nagy, János Tapolcai, Gábor Rétvári
MTA-BME Lendület Future Internet 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. Unfortunately the de facto IP protection mechanism, Loop-Free Alternates (LFA), does no cover all possible failure scenarios that can show up in an operational network. The main concern in this paper is, correspondingly, to construct an overlay on top of the physical network, whereas virtual routers are provisioned that provide LFA protection to otherwise unprotected failure cases. Our main contribution is a new Resilient IP Overlay Design algorithm, which, in contrast to previous work, is guaranteed to terminate with a fully protected topology, runs in polynomial time, and eliminates all adverse LFA loops. 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.**

Keywords*—*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.

Unfortunately, the framework for *fast IP-level protection*, the IP Fast ReRoute scheme (IPFRR, [1]), today does not come equipped with a practical and deployable implementation that would provide an all-out solution. What the basic IPFRR specification recommends, and what most router vendors implement [2]–[4], is Loop-Free Alternates (LFA, [5]), whereas the IGP attempts to find a secondary next-hop that protects against the failure of the default shortest path. It turns out that LFA can protect only 50-80% of possible link failures in general, and node protection is even poorer [6]–[8]. Alternatives to LFA that would provide 100% failure protection [6]–[14], 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.

To build truly resilient IP networks on top of LFA, today operators resort to re-designing the physical topology [15] or straight out re-building it from scratch [16], or re-adjusting the IGP link costs [17]–[19]. These interventions are, however, intrusive and costly, due to adversely interfering with the normal operation of the network, the traffic engineering goals of the operator, and the user experience of the customers.

Recently, there has appeared a new *LFA network optimization technique* that eliminates these hurdles. The idea is to construct a *resilient IP overlay* on top of the physical IP substrate, thereby provisioning "virtual LFAs" to physical routers that have gone unprotected otherwise. In essence, this technique is similar to the tunnel-based IPFRR mechanisms and LFA extensions [10], [20], but instead of defining a new control plane protocol it rather "emulates" 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 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" [21]); compared with this, a one time management plane intervention to deploy a virtual overlay and turning on LFA that is supported by most IP routers out-of-the box [2]–[4] seems a bargain.

The authors in [22] propose an Integer Linear Programming (ILP) based greedy optimization strategy to design the overlay and show that deploying only about 2 virtual routers per physical node already boosts LFA-failure protection close to 100%. Again, this comes without having to define, standardize, and deploy new protocols or interfering with the normal operation of the network in any observable ways, using commercially available networking gear, and with a one time management effort.

The authors in [22], however, leave some important questions open. First, it is left unclear whether their optimization strategy is *optimal*, i.e., whether it always terminates with a 100% LFA-protected network. At the moment, even the very question whether such fully protected overlay is guaranteed to exist goes undecided. Second, at the heart of their technique is an ILP with unpredictable running time. This is especially troublesome, taking into consideration the large size of IP

Research partially supported by the Hungarian Scientific Research Fund (grant No. OTKA 108947 and PD-104939). This document has been produced with the financial assistance of the European Union under the FP7 GÉANT project grant agreement number 605243 as part of the MINERVA Open Call project.

backbones in use today, which can easily trigger unsolvable ILP instances. At the moment it is, thus, of question whether a technique with a firm *polynomial running time bound* exists. Finally, as shall be shown below their technique is vulnerable to a detrimental phenomenon, henceforth referred to as *trap node leakage*, in that it can happen that the virtual overlay supplies LFA to routers that already had LFA-protection in the physical network and, under certain inauspicious circumstances, such "spurious LFAs" can force traffic into an infinite LFA loop, causing packet loss.

Our main goal in this present paper is to close these gaps. In particular, we show that the algorithm in [22] is not optimal, in the sense that it terminates with a partially protected overlay for certain inputs. Correspondingly, we go on to design an improved heuristics. As a main contribution we prove that, depending on a setting of a simple configuration parameter, *our algorithm is either optimal or it is guaranteed to terminate in polynomial time*, and it is possible to efficiently balance between the two according to the preferences of the operator. In addition, we identify cases when the original algorithm of [22] suffers from trap node leakage and *we give an improved set of conditions to eliminate trap node leakage* all together. Finally, we evaluate the performance of our algorithm extensively. Our results indicate that the performance of our heuristics is on par with, and in many cases even better than, the ILP-based algorithm of [22].

The rest of this paper is organized as follows. After a brief background and problem formulation in Section II, we discuss our heuristics and the related theoretical findings in Section III. Numerical studies are presented in Section IV, and finally Section V closes the paper.

II. Model and Problem Formulation

Let the IP network substrate be given as a connected undirected graph $G(V, E)$, where V marks physical nodes ($n = |V|$) and E marks physical edges ($m = |E|$). Let IGP link costs be given by the function $c : E \mapsto \mathbb{Z}^+$, with the cost of edge (i, j) denoted as $c(i, j)$. Denote the shortest path distance between nodes i and j with $\mathrm{dist}(i, j)$. For simplicity, we assume that links are point-to-point and there is a well-defined next-hop for each source-destination pair. LANs and multipath issues are for further study. Let $\mathrm{neigh}(v)$ denote the set of neighbors of some node $v \in V$.

In our model, the substrate adopts Loop-Free Alternates for fast failure protection [5]. To understand LFA consider the network in Fig. 1, initially ignoring the virtual router marked by b' and its incident virtual links. Now, if a packet is sent from b to e the next-hop is f. However, should the link between b and f fail, without adequate measures in place all $b-e$ traffic would be lost until the IGP recalculates shortest paths, which can take anything from 150 ms to several seconds [23]. LFA seeks to avoid this by assigning a secondary next-hop to b, in this case node d, which offers a detour to e bypassing the failed component. As d is not notified about the failure (which makes LFA blazingly fast), the alternate next-hop must be such that its shortest path to the destination does not traverse b. Otherwise, the alternate next-hop might blindly send the packets back to b, leading to a routing loop.

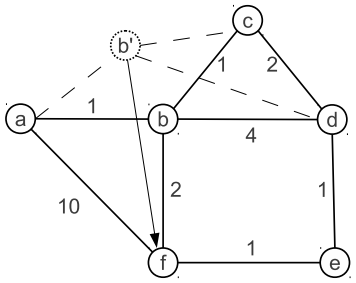


Figure 1: Sample network with IGP link costs. Arrows mark shortest paths, dashed lines are virtual links.

In general, for some source s, destination d, and next-hop t, a neighbor $q \neq t$ is called a *link-protecting $s-d$ LFA* if q is such that the following "loop-free condition" holds [5]:

$$\mathrm{dist}(q, d) < \mathrm{dist}(q, s) + \mathrm{dist}(s, d) \ . \tag{1}$$

Unfortunately, very often it is impossible to satisfy this requirement for all node pairs. In the network of Fig. 1, for instance, c does not have an LFA to d as its only physical neighbor b is upstream. Remarkably, however, if a virtual router b' is established on b and the IGP costs on the virtual links of b' are set so that the shortest path to d traverses f as next-hop, then b' will provide an LFA to $c-d$. As virtual routers are practically indistinguishable from physical ones as far as the IGP is concerned, b' will show up as a legitimate LFA at c. The main observation in [22] is that, by cautiously provisioning a virtual overlay that provides such *virtual LFAs* to unprotected node-pairs, LFA-protection can be improved upon substantially. Here, the level of LFA protection is usually characterized by the *LFA coverage* metric η, defined as the proportion of protected vs. all source-destination pairs [22]:

$$\eta(G, c) = \frac{\text{\# LFA-protected } (s,d) \text{ pairs}}{\text{\#all } (s,d) \text{ pairs}} \ . \tag{2}$$

Note that we consider a source-destination pair protected if *(i)* the source has a link-protecting LFA to the destination and *(ii)* the LFA indeed offers a detour, in that a packet sent from the source via the LFA is guaranteed to reach the destination. This is verified by a simple LFA-aware packet tracer. In the sample network $\eta(G, c) = 0.766$.

A. Problem formulation

In this paper, our main concern is the *Resilient IP Overlay Design* problem. Here, we are given a graph G and costs c and the task is to design an overlay $G_V(V_V, E_V)$ and proper link costs c_V so that LFA coverage $\eta(G_V, c_V)$ is maximized. Easily, we want to achieve this with the minimum number of virtual routers, which translates into minimizing management overhead. Since the full-fledged problem is extremely complex, we describe here a simplified but more workable model [22].

In our model, physical elements (routers and links) are treated the same as virtual elements, with each virtual element having a *default* entity representing the physical element it is

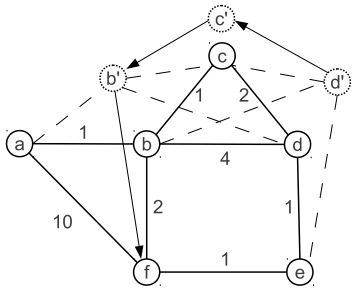


Figure 2: A second sample network with an LFA loop. Arrows mark shortest paths, dashed lines are virtual links.

instantiated on. Formally, let $f_N : V_V \mapsto V$ be a mapping that to each $v \in V_V$ orders the corresponding default node $v \in V$ and similarly, $f_E : E_V \mapsto E$ maps virtual links to default links. Additionally, we require that virtual links connect nodes whose default routers are physically connected. These assumptions generally ensure that the substrate G will be a subgraph of the virtual topology G_V and $V \subseteq V_V$ and $E \subseteq E_V$.

We shall assume single link failures in the physical network, as these make up the majority of unplanned outages in operational IP networks [24]. Crucially, a single physical link failure can cause rolling cascade effects in the overlay, since all the links provisioned on that physical link will go down as well. Formally, for each link $e \in E_V$ we define a Shared Risk Link Group (SRLG) containing all the virtual links sharing the default link for e: $S_E(e) = \{l \in E_V : f_E(l) = f_E(e)\}$. If e fails, all links in $S_E(e)$ fail. Similarly each node $v \in V$ has its own SRLG: $S_N(v) = \{u \in V_V : f_N(u) = f_N(v)\}$.

In order to ensure that no special protocol is needed to distribute SRLGs, currently the LFA specification [5] supports so called *local SRLGs* solely that are restricted to links incident to a single router. Correspondingly, for LFA we cannot configure an SRLG that, say, contains all the virtual links established on the same physical link. As it turns out, this greatly hinders our freedom to shape the virtual overlay.

Consider the topology in Fig. 2 and suppose that b is about to send packets to f. If, for some reason, link (b, f) goes down, b may choose to redirect its traffic to the LFA d'. However, said traffic will never arrive to f as the $d' - f$ detour degrades into the LFA loop $b - d' - c' - b' - c - b$. Here, b' also switches to its LFA c realizing that its link to f has disappeared due the physical failure which b' is unaware of, thanks to the lack of general SRLG support in LFA. What is worse, such *spurious LFAs* can override existing LFAs, like in our case d' can replace the legitimate LFA d. Accordingly, we find that an inadvertent virtualization decision might easily end up *corrupting existing LFAs and decreasing LFA-coverage*, instead of increasing it. Naturally, such cases must be avoided at all costs.

In order to achieve this, it is beneficial to turn our LFA definition SRLG-aware.

Definition 1: For source s, destination d, and $s-d$ next-hop t, node q is an *SRLG-disjoint link-protecting $s-d$ LFA* if

LFA-1: $q \in \text{neigh}(s)$ and $q \neq t$,
LFA-2: $\text{dist}(q, d) < \text{dist}(q, s) + \text{dist}(s, d)$,
LFA-3: $(s, q) \notin S_E(s, t)$ (local SRLG condition), and
LFA-4: each $q \to d$ shortest path is SRLG-disjoint from (s, t).

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.

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: $\text{RIOD}(G, c, U, k)$: given a graph $G(V, E)$, 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 $\eta(G_V, c_v)$ is maximal, where:

- $V_V = V \cup U_V$ with $\forall u \in U_V : f_N(u) \in U$ (*virtual nodes provisioned only inside U*)
- $|U_V| \leq k$ (*no more than k virtual instances*),
- $E_v \subseteq E \cup (U_V \times \text{neigh}(U))$ (*virtual links between physically connected routers*),
- shortest paths in G do not change (*the substrate is unaltered*); and
- LFA-4 is guaranteed (*no spurious LFAs*).

In another version of RIOD, we may ask to reach a certain LFA-coverage threshold while minimizing the size of the overlay $|U_V|$. For the decision version of RIOD, the following complexity characterization was given in [22].

Proposition 1: RIOD is NP-complete.

In fact, [22] shows that RIOD is already NP-complete in the simple case when only a single virtual node is to be provisioned on a fixed physical router $v \in V$ (i.e., the case of $\text{RIOD}(G, c, U, k)$ with $U = \{v\}$ and $k = 1$). We shall refer to this useful special case of RIOD as the *LFA Virtual Router Augmentation Problem* $\text{LFAVirt}(G, c, v)$.

B. *Solving* RIOD

The authors in [22] propose a greedy optimization strategy to solve the special case $\text{RIOD}(G, c, V, \infty)$ (i.e., every router can host virtual routers and there is no limit on the size of the overlay). Their method consists in iteratively provisioning the virtual node whose addition to the network increases LFA-coverage the most, by solving $\text{LFAVirt}(G, c, v)$ for each $v \in V$ and taking the maximum. The iteration terminates when LFA-coverage stops increasing.

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

do: $\eta \leftarrow \eta(G, c)$
  for each $w \in U$: $(c_w, \eta_w) \leftarrow$ solve $\text{LFAVirt}(G, c, w)$
  $(w', \eta') \leftarrow$ choose $w \in U$ that maximizes η_w
  add w' to G and set costs to c_w
while $\eta' > \eta$

Here, $\text{LFAVirt}(G, c, w)$ indicates how LFA-coverage would improve if a new virtual router were instantiated on w. The authors in [22] present an Integer Linear Programming based method to solve this problem. They pre-calculate the following four sets: the set of *eligible node-pairs* $\mathcal{L}$ that can gain an LFA

from a virtual router w' established on w; for each $(s,d) \in \mathcal{L}$ a set of *escape nodes* $\mathcal{E}_{sd}$ consisting of the nodes which, if chosen as a next-hop for w' to d, would render w' an $s-d$ LFA; the *critical node-pairs* $\mathcal{Q}$ that can gain a spurious LFA from w'; and finally the *trap nodes* $\mathcal{T}_{sd}$ which, if chosen as $w'-d$ next-hop, would make w' a spurious $s-d$ LFA. Then, the ILP computes the optimal virtual links and the respective link costs so that the most escape nodes become next-hops for w' and hence LFA-coverage is maximized, while no trap node becomes next-hop and so spurious LFAs never show up.

This ILP, integrated into the greedy optimization strategy of Alg. 1, proved very powerful in practice [22]. In the rest of this paper, we further explore the theoretical and algorithmic aspects of RIOD in order to close the remaining gaps in our understanding of the problem.

III. An Improved Heuristic

If we take a closer look on the greedy optimization strategy as proposed in [22], we find the following issues: optimality is not guaranteed; the complexity of the ILP for solving $\mathrm{LFAVirt}$ can be substantial and unpredictable; and it is unclear whether the above definition for trap nodes indeed eliminates spurious LFAs. In what follows we address these issues one by one.

A. Eliminating spurious LFAs

The key subroutine in Alg. 1 is solving $\mathrm{LFAVirt}(G,c,w)$. This delivers the optimal virtual links and link costs to be set at the virtual router w' added to $w \in V$, so that LFA-coverage is maximized and spurious LFAs are ruled out. The authors in [22] propose to eliminate spurious LFAs using the notion of *trap nodes*: given an $s-d$ pair, the set of trap nodes $\mathcal{T}_{sd}$ contains all the nodes $g \in \mathrm{neigh}(w)$ which, if assigned as a $w'-d$ next-hop, would create a spurious $s-d$ LFA. Easily, we will want to prevent such choices to preclude LFA loops.

The trap node condition is formally described through a so called packet-tracing procedure in [22].

Definition 3: Let $G_{s,d,g}(V_{s,d,g}, E_{s,d,g})$ be the directed graph spanning the links traversed by a packet tracer procedure starting from node g and running at most n steps on a graph with the link between s and the $s-d$ next-hop removed.

Condition 1: $g \in T_{sd}$ if $g \in \mathrm{neigh}(w)$ and $d \notin V_{s,d,g}$.

In other words, a node g is a trap node if a packet, sent from s through the prospective LFA w' and from there through the next-hop g towards d, would get stuck in an LFA loop. Below, we argue that this condition is too weak, in the sense that it does not cover all "bad choices" for the w' next-hops.

Observation 1: There is a graph G, costs c, a node-pair $s-d$, and nodes w and g, so that adding the virtual node w' to w and setting the $w'-d$ next-hop to g results in an LFA-loop, even though $g \notin T_{sd}$ as per Condition 1.

Consider the network in Fig. 3 and, initially, assume that b' is the only virtual router. At this step, we notice that the insertion of d' would protect the (unprotected) $e-f$ node pair, provided that the $d'-f$ next-hop was chosen to, say, node b'. Before making this decision, though, we still have to check whether it would create a spurious LFA to *other* node pairs.

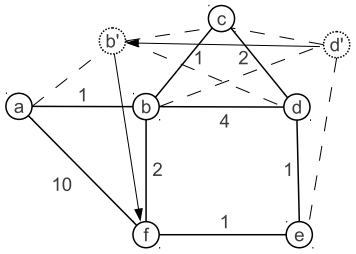


Figure 3: Trap node leakage.

Suppose we are to check Condition 1 for the node-pair $b-f$. As at this point d' is not yet part of the topology, we find that packets from b' reach f along the path $b'-c-b-d-e-f$ (note, that b' also switches to its LFA and so does b) and we conclude that b' is *not* a trap node for $b-f$. Nonetheless, if we indeed provision d' with next-hop b', then we end up with the LFA loop $b-d'-b'-c-b$. The reason for this *trap node leakage* phenomenon is that the spurious LFA d' appears alongside the already existing LFA d, and since LFA implementations do not have a way to configure LFA preference neither they support general SRLGs, it might very well be the case that the spurious LFA overrides the legitimate one. Unfortunately, we do not have a way to anticipate this, since the spurious LFA is still not part of the network when packet tracing is done.

Next, we strengthen the trap node condition to eliminate trap leakage. As it turns out, this comes at the price of ruling out a small portion of options that would yield a legitimate LFA.

Condition 2: $g \in T_{sd}$, if *(i)* $g \in \mathrm{neigh}(w)$ and $d \notin V_{s,d,g}$, or *(ii)* $g \in S_N(s)$ and $\exists l \in S_N(s)$ so that $l \in V_{s,d,g}$.

Here, *(i)* coincides with Condition 1, and *(ii)* ensures that the detour never returns to s or any of the virtual nodes provisioned on s. This modification, as one easily checks, immediately rules out the case demonstrated in Observation 1. The general correctness proof is trivial and so omitted here for brevity.

B. Optimality

So far, it has been an open question whether the greedy optimization algorithm as of Alg. 1 guarantees a fully-protected overlay. Below, we answer this question in the negative.

It could still happen that Alg. 1 fails to find the fully-protected overlay because such overlay may just not happen to exist. First, we prove that this is not the case, showing that $\mathrm{RIOD}(G,c,V,\infty)$ is always solvable to optimality.

Theorem 1: Given a 2-connected graph G with positive costs c, there is an overlay G_V and costs c_V that solve $\mathrm{RIOD}(G,c,V,\infty)$ with $\eta(G_V,c_V)=1$.

Proof: We show that, given any unprotected node-pair $s-d$, there is a proper set of virtual nodes whose addition will create a link-protecting SRLG-disjoint $s-d$ LFA. It follows that if we apply this step to each unprotected node-pair, then LFA-coverage eventually reaches 100%.

We protect the $s-d$ pair by provisioning a "virtual tunnel" between s and d that provides a detour for s bypassing its failed next-hop. Let $s-q-\ldots-r-d$ be an $s-d$ path disjoint from the $s-d$ 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' and the one on r by r'. Connect s to q' and r' to d and set the link cost on $s-q'$ "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' is now an $s-d$ LFA. We still need to show that LFA-4 holds and so no spurious LFAs emerge, but this is guaranteed as there are only two entry points to the virtual tunnel, q' and r', and q' is protected by the local SRLG at s (as of LFA-3) and r' is never an LFA due to its low cost. ■

Next, we show that, even though a solution with complete LFA-coverage always exists, the greedy algorithm of [22] does not necessarily find it.

Theorem 2: Alg. 1 may terminate with $\eta(G_V, c_V) < 1$.

Proof: We show a counter-example in Fig. 4. Here, $\eta = \frac{19}{20}$, as the only unprotected node-pair is $c-d$. 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, Alg. 1 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).

As it turns out, the real shortcoming in Alg. 1 is that it augments the graph in steps too small, with only a single virtual node in each iteration. Instead, one might try to instantiate two virtual routers when adding only 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 $|U_l| = l$.

Our improved algorithm is then based on simply trying increasingly larger connected sets of virtual nodes until LFA-coverage eventually improves. Note that, by Theorem 1, 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 below modified greedy algorithm implements these ideas.

Algorithm 2 Modified algorithm for $\mathrm{RIOD}(G, c, U, \infty)$

```
do η ← η(G, c)
    for l = 1 to k do
        for each connected l-set U_l ⊆ U do
            (c_{U_l}, η_{U_l}) ← solve LFAVirt(G, c, U_l)
        end for
        (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
        end if
    end for
while η' > η
```

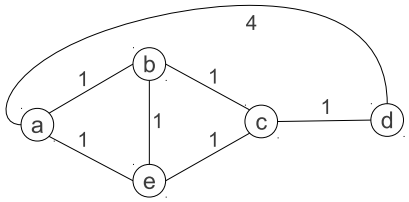


Figure 4: A graph demonstrating that Alg. 1 does not necessarily attain complete LFA-coverage.

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. Also note that the improved trap node characterization of Condition 2 is trivial to generalize to this modified algorithm, and so SRLG-disjointness is guaranteed.

C. Polynomial 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 instead of merely having to add a single virtual router to a known node. In addition, our solution, in contrast to the ILP in [22], must also run in polynomial time. Below, we describe a new heuristics tailored for this setting.

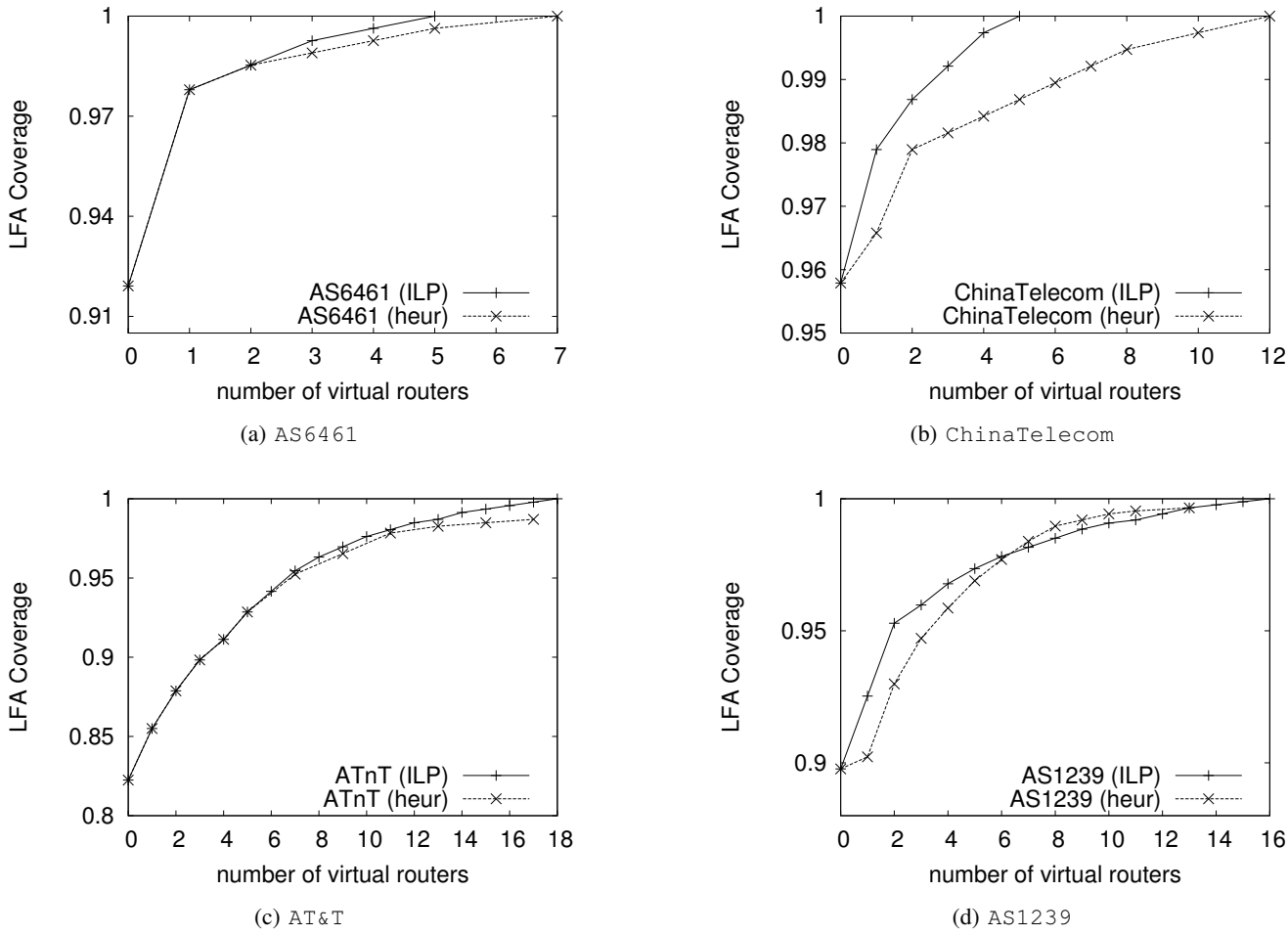
Suppose that we are about to solve $\mathrm{LFAVirt}(G, c, U_l)$ by provisioning a set of virtual routers U_l' on U_l, with each $u \in U_l$ hosting a single virtual instance u'. Also suppose that *eligible node pairs* $\mathcal{L}$, *critical node pairs* $\mathcal{Q}$, and *escape nodes* $\mathcal{E}_{sd} : sd \in \mathcal{L}$ are determined as described in Section II-B and [22], and *trap nodes* $\mathcal{T}_{sd} : sd \in \mathcal{Q}$ are calculated using Condition 2. Finally, denote by $\mathrm{neigh}(U_l)$ the neighbors of nodes in U_l that are outside U_l.

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 node g'. The way we do that is that we explicitly rule out all $g \in \mathrm{neigh}(U_l)$ that happen to be trap nodes, and then we take the node g' which is an escape node for the most eligible node pairs and hence creates the largest number of new LFAs. Formally:

$$g' = \underset{\substack{g \in \mathrm{neigh}(U_l) \\ g \notin \mathcal{T}_{sd}: sd \in \mathcal{Q}}}{\operatorname{argmax}} |sd \in \mathcal{L} : g \in \mathcal{E}_{sd}| \quad . \tag{3}$$

Having chosen g' as of (3), 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 node g' which is again set to low cost. It is now trivial to check that this setting indeed yields that U_l' has a single exit node: g'.

The computational complexity of Alg. 2 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 $\mathcal{L}$



(a) AS6461 (b) ChinaTelecom

(c) AT&T (d) AS1239

Figure 5: LFA-coverage for small and middle-sized topologies as virtual routers are added iteratively with Alg. 1 (ILP) and Alg. 2 (heur). The x axis gives the number of virtual routers instantiated and the y axis marks the LFA-coverage.

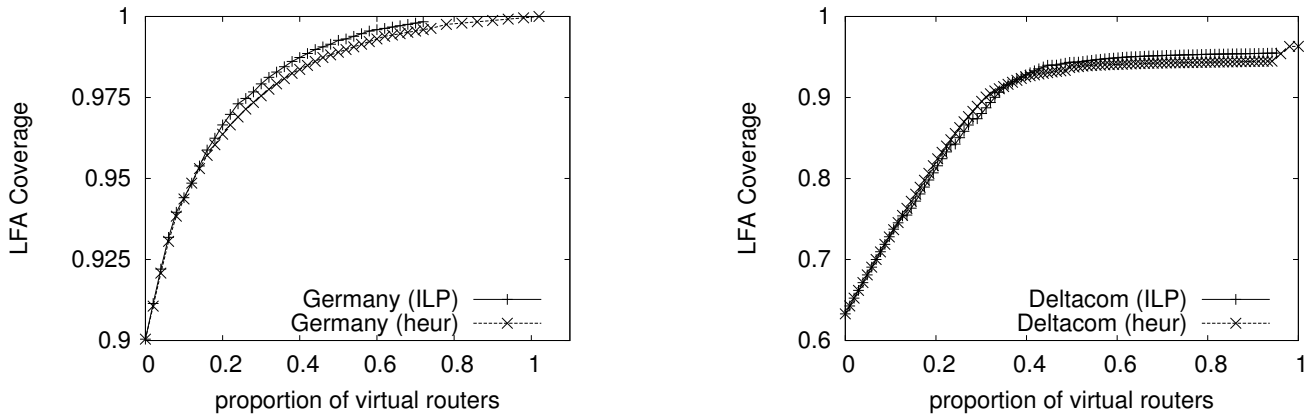
and $\mathcal{Q}$ contain $O(n^2)$ elements and $\mathcal{E}_{sd}$ and $\mathcal{T}_{sd}$ contain $O(n)$ elements, and each can be calculated in $O(n^2)$ steps.

Correspondingly, Alg. 2 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 $k = 2$. In this case the theoretical worst-case complexity of Alg. 2 is $O(n^5)$ 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. 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.

IV. Numerical evaluation

Finally, we evaluated the performance of the improved heuristic algorithm in extensive numerical studies. Our main concern was the performance of Alg. 2 compared to that of the original algorithm of [22] (Alg. 1), measured by the LFA-coverage metric η as an increasing number of virtual routers is provisioned in the network. We did not expect our heuristics to perform better: after all, Alg. 1 obtains an optimal solution for each subproblem using an Integer Linear Program, while Alg. 2 is merely a heuristics. We, however, expected the performance to not be prohibitively worse and the penalty be made up for in the improved running time.

The implementation was done in C++ using the LEMON graph library [25]. For solving the ILPs we employed GLPK, the GNU Linear Programming Kit [26]. We used the same input topologies as [22]. These represent service provider networks, many of which come with real or inferred link costs. For brevity we do not present all the results here. Instead, we choose 6 topologies that sufficiently typify our results, namely, AS6461 and AS1239 from the Rocketfuel dataset [27]; AT&T and the 50-node Germany backbone from SNDlib [28]; and ChinaTelecom and the Deltacom backbone from



(a) Germany (b) Deltacom

Figure 6: LFA-coverage for backbone topologies as virtual routers are added iteratively with Alg. 1 (ILP) and Alg. 2 (heur). The x axis marks the proportion of virtual per physical nodes and the y axis marks the LFA-coverage.

Table I: Evaluated topologies: name, number of nodes n, and number of links m for each topology.

Name	n	m
AS6461 [27]	17	37
ChinaTelecom [29]	20	44
Deltacom [29]	103	151
AT&T [28]	22	38
AS1239 [27]	30	69
Germany [28]	50	88

the Topology-Zoo project [29]. Here, AS6461, AS1239, AT&T, and ChinaTelecom represent small and middle-sized provider topologies, while Germany and Deltacom are larger backbone topologies. Information for the topologies is presented in Table. I. The results themselves for the small and middle-sized networks are given in Fig. 5 and for backbone topologies in Fig. 6.

The topology AS6461 is highly representative for our results. Here, Alg. 1 attains 100% LFA-coverage just by adding 5 virtual routers, while our heuristics matches the performance closely and also reaches full coverage with 7 routers. Observe that the heuristics added two virtual routers in the final step, which is indicated by the missing data point at 6. Similar is the case for ChinaTelecom, Germany, and Deltacom. The latter are backbones, and we observe that in both cases our heuristics eventually even out-performs the ILP, at the price of setting slightly more virtual routers. On the other hand, for the case of AT&T and AS1239 we see a small performance hit. In both cases the heuristics fails to find the fully-protected topology. To obtain the optimal solution, one would need to elevate parameter k in such cases. Nonetheless, the performance penalty never exceeds 0.2%. In summary, we see very little performance lag with our heuristics as compared to the ILP and in some cases we even discover improved performance. Both algorithms are able to bring small and middle-sized networks close to perfect LFA-coverage by provisioning just a couple of virtual routers, and in backbones we see similar improved protection coverage just by provisioning roughly one virtual router per node on average. This suggests the Resilient IP Overlays hold great promise for building robust IP networks.

Notably, we found at least one case (the topology AS1239) where our packet tracer signaled an LFA loop with Alg. 1, as a clear indication of trap node leakage identified in Observation 1. With Alg. 2, however, no such LFA loops arise, thanks to the improved trap node condition. Finally, we also found that the running time of Alg. 2 and Alg. 1 was in the order of mere seconds (with the heuristics being consistently faster than the ILP), except for a few cases when we hit a hard instance that took hours for the ILP solver to work out. The heuristics, on the other hand, always terminated in reasonable and predictable time frame thanks to its guaranteed polynomial complexity.

V. Conclusions

Lately, Loop-Free Alternates for implementing fast protection in IP and MPLS/LDP networks have become an industrial requirement [30]. Unfortunately, LFA in many cases leaves the network vulnerable to certain failure scenarios. In this paper, we have sought to resolve this issue by establishing a Resilient IP Overlay on top of the physical network, which supplies "virtual LFAs" to unprotected node-pairs. We have identified several deficiencies in existing work and we designed an improved heuristics to overcome these. Our algorithm eliminates LFA loops completely and allows to trade-off running time for optimality via setting a simple parameter. As evidenced by numerical evaluations, the performance of our algorithm in some cases even surpasses that of the ILP-based solution of [22]. Moreover, in contrast to that, it also runs in polynomial time. 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 3.7," 2008.

[3] Juniper Networks, "JUNOS 9.6 Routing protocols configuration guide," 2009.

[4] Hewlett-Packard, "HP 6600 Router Series: QuickSpecs," 2008, available online: http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF.

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

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

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

[8] 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.

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

[10] 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.

[11] 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.

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

[13] S. Bryant, M. Shand, and S. Previdi, "IP fast reroute using Not-via addresses," Internet Draft, March 2010.

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

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

[16] C. Filsfils *et al.*, "Loop-Free Alternate (LFA) applicability in Service Provider (SP) networks," RFC 6571, June 2012.

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

[18] H. T. Viet, P. Francois, Y. Deville, and O. Bonaventure, "Implementation of a traffic engineering technique that preserves IP Fast Reroute in COMET," in *Rencontres Francophones sur les Aspects Algorithmiques des Telecommunications, Algotel (2009)*, 2009.

[19] G. Rétvári, L. Csikor, J. Tapolcai, G. Enyedi, and A. Császár, "Optimizing IGP link costs for improving IP-level resilience," in *Proc. International Workshop on Design Of Reliable Communication Networks (DRCN)*, 2011.

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

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

[22] J. Tapolcai and G. Rétvári, "Router virtualization for improving IP-level resilience," in *INFOCOM*, 2013.

[23] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, 2005.

[24] 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.

[25] "LEMON – Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu/, 2009.

[26] GLPK, "GNU Linear Programming Kit," `http://www.gnu.org/software/glpk`.

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

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

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

[30] N. Leymann, B. Decraene, C. Filsfils, and M. Konstantynowicz, "Seamless MPLS architecture," Internet Draft, March 2011.