Safe Routing Reconfigurations with Route Redistribution

Stefano Vissicchio*, Laurent Vanbever†, Luca Cittadini‡, Geoffrey G. Xie§, Olivier Bonaventure*
*Université catholique de Louvain, †Princeton University, ‡RomaTre University, §Naval Postgraduate School

***Abstract*—Simultaneously providing flexibility, evolvability and correctness of routing is one of the basic and still unsolved problems in networking. Route redistribution provides a tool, used in many enterprise networks, to either partition a network into multiple routing domains or merge previously independent networks. However, no general technique exists for changing a live network's route redistribution configuration without incurring packet losses and service disruptions.**

In this paper, we study the problem of how to safely transition between route redistribution configurations. We investigate what anomalies may occur in the reconfiguration process, showing that many long-lasting forwarding loops can and do occur if naive techniques are applied. We devise new sufficient conditions for anomaly-free reconfigurations, and we leverage them to build provably safe and practical reconfiguration procedures. Our procedures enable seamless network re-organizations to accomplish both short-term objectives, such as local repair or traffic engineering, and long-term requirement changes.

I. Introduction

Most networking textbooks explain the classical intradomain routing protocols, i.e., IGPs (RIP, OSPF, EIGRP, ISIS, ...) and mention that each enterprise can select one as its preferred routing protocol. However, routing in enterprise networks is much more complex than that [1], [2]. In many enterprise networks, several intradomain routing protocols have to coexist. This coexistence can be due to several business reasons, e.g., company mergers and acquisitions, or pragmatic reasons, e.g., a specific routing protocol not being supported on routers from a given vendor. To allow different intradomain routing protocols to coexist, network administrators depend on route redistribution (RR). With RR, a router that participates in in several routing protocols can selectively pass routes between the different protocol instances. Although BGP could also be used for this purpose, enterprise network administrators often prefer RR because it supports additional functionality such as efficient routing and domain backup [3].

A network using RR to connect multiple intradomain routing protocols is said to be partitioned into several routing domains. A routing domain (RD) is a connected subgraph of the network that is composed of routers that use the same intradomain routing protocol. Many enterprise networks are divided in routing domains [3]. Using multiple routing domains has several advantages. For example, routing domains can act as administrative boundaries, where different teams separately manage different portions of the network. Moreover, operators can combine the advantages of different routing protocols (or different configuration modes), e.g., optimizing one RD for scalability and another for fine-grained traffic engineering.

The division of an enterprise network in routing domains is not static. Various events can force an enterprise network operator to change the boundaries of its routing domains. Splitting and merging networks, e.g., to accommodate mergers and acquisitions, are two radical examples. Simply replacing a router by another router from a different brand may also force a boundary modification. Finally, the events can be driven by short-term objectives such as a change to the traffic engineering requirements or router maintenance, as well as long-term redesigns such as transitioning part of the network to the new software defined networking (SDN) paradigm.

Reconfiguring an enterprise network divided in several routing domains is challenging. Firstly, route redistribution is prone to routing and forwarding anomalies [3], and designing a correct RR configuration is known to be hard [4], [5]. Secondly, the coexistence of multiple routing domains exacerbates the difficulty of routing reconfiguration. Indeed, while even reconfiguring a single routing protocol can incur routing and forwarding anomalies [6], the presence of RR forces operators to also navigate the intricate interactions between different routing protocols that exchange routes. Unfortunately, operators currently have no methodological support nor tool that help them safely transition between route redistribution configurations. In this paper, we address these challenges by considering the following question: *Given a network using multiple routing domains and route redistribution, how can we reconfigure routing without triggering persistent anomalies?*

We focus on *persistent* routing and forwarding anomalies, i.e., those caused by the reconfiguration process and can only be solved with a configuration change. These long lasting anomalies are much more harmful than transient anomalies that may occur during protocol convergence. We assume that an IGP is the only routing protocol configured in each routing domain. This assumption matches the typical configuration of enterprise networks on which we focus, and avoids anomalies due to a more complex protocol stack [**?**].

This paper is organized as follows. We present the problem of reconfiguring a network with multiple routing domains, and show that persistent anomalies can and do occur during this process in Sec. II. We explain why the presence of multiple routing domains invalidates most of the assumptions made in some previous research in Sec. III. We propose new sufficient conditions for loop-free route redistribution configurations that

Stefano Vissicchio is a postdoctoral researcher of the Belgian fund for scientific research (F.R.S.-FNRS)

are general enough to apply during a reconfiguration process in Sec. IV. Using the new sufficient conditions as a basis, we describe procedures to arbitrarily reconfigure networks with multiple routing domains while guaranteeing the absence of anomalies in Sec. V. Finally, we show the practicality of our procedures through a realistic reconfiguration where a single routing domain is split in two in Sec. VI.

II. Reconfigurations With Route Redistribution

In this section, we introduce the problem of changing the routing configuration of an enterprise network with multiple RDs, and we evaluate its practical impact.

A. *Problem Overview*

The operational community has produced a number of best practices for reconfigurations (e.g., [7], [8]). However, their scope is restricted to very specific use cases and do not guarantee routing safety during the reconfiguration. The problem of disruption-free routing reconfiguration has recently been studied in various contexts, including adjusting IGP link weights [9], changing the IP topology [10], migrating virtual routers [11], [12], reconfiguring an IGP [6] or a BGP [?] network, and updating SDN switches [13]. All these efforts, however, consider networks with a single routing domain. Concurrently, research work [14], [15], [16] has introduced models for studying networks with multiple routing protocols, and sufficient conditions for safe route redistribution configuration. These results do not address the question of how to safely change a route redistribution configuration.

We build upon the algorithmic techniques proposed by Vanbever et al. in [6] to update the routing configuration without causing anomalies. We reconfigure routing in a network in a discrete number of steps. At each step, a single router is reconfigured and then a given time is waited for routing convergence. The configuration of a router can be changed in a limited number of ways, namely, by adding/removing the support of a specific routing instance, changing the administrative distance value of a given routing instance, and activating/deactivating route redistribution. A router is *reconfigured* when it runs with its final configuration. A reconfiguration is completed when all the routers are reconfigured.

Reconfiguring a network router-by-router can incur persistent anomalies during reconfiguration even when the initial and the final configurations are anomaly-free, because of inconsistent protocol preferences across different routers. We refer to those anomalies as reconfiguration anomalies. Previous work [6] showed that reconfiguration anomalies can be prevented in single-domain networks by reconfiguring the routers in a well-chosen ordering. Unfortunately, when route redistribution is enabled, existing techniques are not guaranteed to be correct anymore, since new anomalies can result from the interplay between different routing protocols and route redistribution. Route redistribution is informally defined as the process of receiving a route from one routing domain, discarding all its protocol- and routing domain-specific information, and announcing it in another routing domain with a new set of attributes (e.g., a new metric) [3].

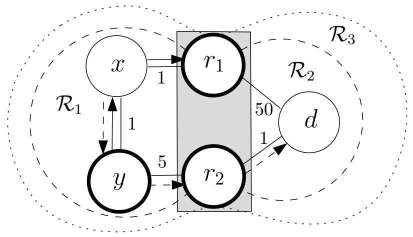


Fig. 1. A forwarding loop during a routing domain splitting reconfiguration.

As an example, Fig. 1 shows a forwarding loop occurring during a RD splitting reconfiguration. The graphic convention in the figure is used throughout the paper. Routers are represented by solid circles, and routing domains are represented by dashed curves. Routers at the border between two routing domains are contained in a shaded box. Solid lines represent links between routers and are annotated with the corresponding link weight. Each figure depicts a specific reconfiguration step. To represent the progress of the reconfiguration, we use thicker lines for routers that have already been reconfigured.

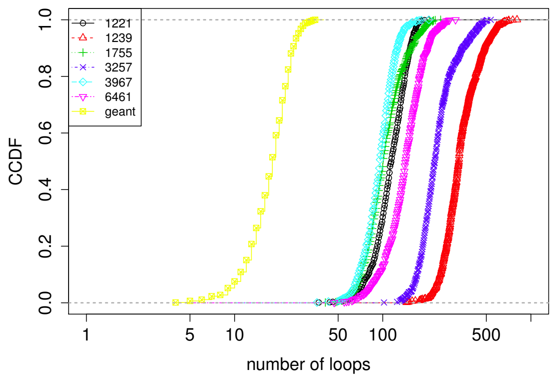
In the example in Fig. 1, routing domain $\mathcal{R}_3$ is being split in two RDs $\mathcal{R}_1$ and $\mathcal{R}_2$. In the initial configuration, only $\mathcal{R}_3$ exists, i.e., all routers in the network run a single protocol p_3. The final configuration has two RDs, $\mathcal{R}_1$ and $\mathcal{R}_2$, connected via route redistribution. Routing protocols p_1 and p_2 are respectively run in $\mathcal{R}_1$ and $\mathcal{R}_2$. Routers r_1 and r_2 run both p_1 and p_2, and redistribute routes from $\mathcal{R}_2$ to $\mathcal{R}_1$. Hence, x and y receive two redistributed routes to d, one from r_1 and the other from r_2. Assuming that the same metric is applied to routes redistributed by both r_1 and r_2, then x and y forward traffic towards d via r_1, since r_1 is the closest between r_1 and r_2 according to the link weights set in $\mathcal{R}_1$. Hence, both the initial and the final configurations are anomaly-free.

The figure shows a snapshot of an intermediate reconfiguration step where r_1, r_2 and y are reconfigured while x is not. Note that there is a persistent forwarding loop. Since r_1 and r_2 are reconfigured, they redistribute routes from $\mathcal{R}_2$ to $\mathcal{R}_1$. Moreover, since y is reconfigured, y selects as best the route redistributed by r_1, represented by the solid arrows in Fig. 1, hence it forwards traffic to d via x. However, x is still in its initial configuration, and uses the route provided by p_3 (dashed arrows in Fig. 1) hence forwarding the traffic to d back to y.

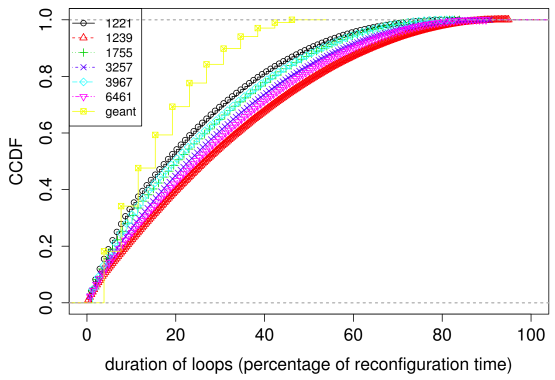
B. *Experimental Analysis*

To assess the likelihood that anomalies occur during reconfigurations of networks with multiple RDs, we simulate a routing domain splitting on the Geant and Rocketfuel topologies, which are publicly available [17], [18].

For each topology, we assume the initial configuration to be a single routing domain, and we build the final configuration as follows. First, we randomly select a connected component composed of half of the routers in the topology. We consider this connected component as one routing domain. Then, we remove this routing domain and we map the remaining connected components to other routing domains. We perform the reconfiguration one RD at a time. Within one routing



(a)



(b)

Fig. 2. Even in small topologies, a large number (left) of long-lasting (right) forwarding loops can appear when splitting a routing domain. Even worse, the number and the duration of the forwarding loops grow with the size of the network.

domain, we compute the order of router reconfigurations as a random permutation. For each topology, we choose 30 different random orderings to ensure statistical significance. Those simulations are intended to mimic the case in which route redistribution is activated first and then routing domains are reconfigured one by one.

Fig. 2 reports the results of the experiments just described. Although our reconfiguration does not modify link weights, we observe a sheer number of long-lasting reconfiguration loops. In particular, Fig. 2(a) and 2(b) respectively report the Cumulative Distribution Functions (CDFs) of the number and of the duration of forwarding loops. Even in a small network like Geant (35 routers), up to 31 loops occur during the reconfiguration *exclusively* because of the activation of route redistribution, with a median around 20 loops per experiment. In addition of being numerous, many loops are also long-lasting. In median, a loop lasts for about 15% of the reconfiguration process, while some loops span about 50% of the reconfiguration. As expected, the number and the duration of loops grow with the size of the network. In Rocketfuel topologies, between 100 and more than 500 loops are raised in the worst case. Even worse, loops also last longer in Rocketfuel topologies, with peaks reaching more than 80% of the reconfiguration process. Depending on the network size, those results imply that reconfiguration loops can last in real networks from minutes to hours, even if the reconfiguration of a single router only takes few seconds.

III. Limitations of Existing Theory

We now show that the presence of route redistribution makes the safe reconfiguration problem much harder.

A. Background and Notation

Routers exchange messages and take decisions on two different levels: the data-plane and the control-plane.

The data-plane level is typically implemented in hardware and is used by a router to efficiently forward packets. Data-plane decisions are based on the Forwarding Information Base (FIB), a data structure that, for each destination, specifies the outgoing interface and the next-hop. The FIB may contain multiple next-hops for the same destination, e.g., in case of Equal Cost Multi-Path (ECMP). In the following, we denote with $fib(r, d, t)$ the set of next-hops of router r at time t for packets to destination d, as stored in the FIB of r. The forwarding paths that are actually followed by data packets are the result of the concatenation of FIB entries of a sequence of routers. More formally, we define the *forwarding paths* followed by packets originated by s and destined to d at time t as the set $\pi(s, d, t)$ of all the sequences of routers of the form $(v_0 \dots v_k)$, where $k \geq 0$, $v_0 = s$, and $\forall i = 0, \dots, k-1$ $v_{i+1} \in fib(v_i, d, t)$.

FIBs are computed by routers on the basis of information exchanged by routers on the control-plane (implemented in software). In particular, routers run routing protocols to agree on the forwarding paths to be used. A routing protocol defines the format of messages and the algorithms to select and distribute *routes*, i.e., forwarding paths available in the network. In the following, we focus on commonly used link-state routing protocols such as IS-IS and OSPF. Basically, link-state protocols are designed to make all routers aware of the network topology. Each router locally computes its best routes relying on some variant of Dijkstra's algorithm.

Routers can run several routing protocols at the same time. In this case, each router instantiates a separate routing process for each routing protocol. In the following we refer to a routing process running on a set of routers as *routing instance*, or simply instance. To solve conflicts in the computation of the FIB, the *Administrative Distance (AD)* establishes a relative preference between routing instances running on the same router. Lower AD values correspond to higher preference. Each routing protocol is normally assigned a default AD value (e.g., 110 for OSPF, 115 for IS-IS, and 1 for static routes on Cisco routers). However, network operators can fine-tune AD values even on a per-destination basis. To simplify the notation, we denote the AD value assigned to a routing instance p as $AD(p)$, and we separately specify routers, destinations and time at which this AD setting applies. We refer to the instance with the lowest AD on r at time t for destination d as $pref(r, d, t)$. For each destination d, each router r selects the best route to d according to instance $pref(r, d, t)$. Then, it stores its best routes in a data structure called Routing Information Base (RIB). Each RIB entry $rib(r, d, t)$ contains information on the forwarding paths $\pi(r, d, t)$ that would be followed if $pref(r, d, t)$ were the only routing instance in the network. The FIB is then derived as a subset of the information in the RIB. In particular, $fib(r, d, t)$ contains the first hop in $rib(r, d, t)$. However, since different AD settings can be configured on different routers, we can have $\pi(r, d, t) \neq rib(r, d, t)$.

We formally define a *routing domain* (RD) as the set of all routers that can exchange routing information using the same routing instance. We distinguish between frontier and internal routers. For each RD $\mathcal{R}$, a router $r \in \mathcal{R}$ is a *frontier router* if it has at least one neighbor $x \notin \mathcal{R}$. Otherwise, r is called *internal* to $\mathcal{R}$. We refer to the routing instance running in RD $\mathcal{R}$ as $proto(\mathcal{R})$.

In a network divided in several routing domains, destinations can be announced in one or more RDs, with route redistribution responsible for propagating a route in the remaining RDs. Observe that some destinations (e.g., loopback interfaces) are typically announced in a single RD, while others (e.g., LANs attached to routers belonging to different routing domains, or a default route) can be announced in several RDs. We define *origin routing domain* of a destination d as the set of disjoint RDs in which d is originated.

B. Known Sufficient Conditions Cannot be Preserved in the Reconfiguration

Route redistribution can cause routing and forwarding anomalies even in a static configuration [3]. In order to focus on anomalies due to the reconfiguration process, we assume that the initial and final configurations are provably anomaly-free, i.e., comply with the following assumptions.

Assumption 1 (Strict Monotonicity): The AD value of a route always increases when it is redistributed, and never decreases when it is propagated inside a RD.

Assumption 2 (Network-Wide Protocol Preference): The AD of each routing instance is consistent across all routers in the instance and is not assigned to any other instance.

Assumption 1 guarantees the absence of routing anomalies [4], while Assumption 2 ensures the absence of forwarding anomalies due to inconsistent route selection [5].

Observe that the forwarding loop in the example of Fig. 1 occurs even if the initial and final configurations are compliant with those assumptions. Even worse, the presence of route redistribution invalidates existing reconfiguration techniques that are provably correct for networks with a single RD.

Consider Fig. 3 and assume that the goal of the reconfiguration is to change the weight of link (x, y) from 1 to 50 and the weight of link (x, r_2) from 50 to 20. We use the IGP reconfiguration technique defined in [6], which introduces

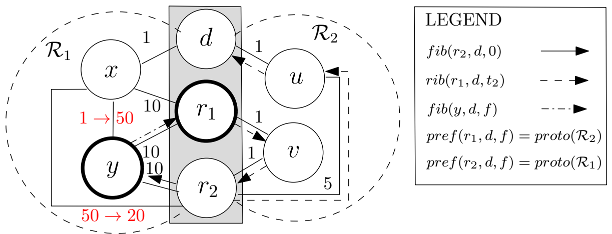


Fig. 3. Reconfiguration techniques which are lossless on a single RD can create forwarding loops in a multi-RD network.

the new configuration in a new routing instance which has an AD value higher than the existing routing instance. The AD of the new instance is then changed on a per-router basis to make it preferred over the old one. Let p_1 and $\bar{p_1}$ be the old and the new routing instances in $\mathcal{R}_1$, respectively, and let $p_2 = proto(\mathcal{R}_2)$. Immediately after having introduced $\bar{p_1}$, we have $AD(p_1) < AD(p_2) < AD(\bar{p_1})$ network-wide. We then increase the AD value of the old routing instance on one router at the time, until we have $AD(p_2) < AD(\bar{p_1}) < AD(p_1)$ on all the routers in the network. Provided that Assumption 1 holds, both the initial and final configurations are compliant with known sufficient conditions that ensure their correctness.

Observe that, as a side effect of the reconfiguration in $\mathcal{R}_1$, the relative preference of $\mathcal{R}_1$ and $\mathcal{R}_2$ is inverted. The technique in [6] does not take the presence of $\mathcal{R}_2$ into account, yielding to possible anomalies in intermediate configurations. For example, according to [6], we can start by reconfiguring routers r_1 and y, as shown in Fig. 3. Consider now destination d, which is announced in routing domains $\mathcal{R}_1$ and $\mathcal{R}_2$. To reach destination d, router r_1 uses p_2 whereas other frontier routers still use p_1. Moreover, as the weight of link (x, y) is changed in $\bar{p_1}$, router y uses r_1 as its next-hop, which causes a loop to occur involving routers y, r_1, v, and r_2.

Similar examples can be shown where the forwarding loops affect destinations with multiple origin RDs (e.g., default routes). Intuitively, forwarding loops appear because Assumption 2 is intrinsically incompatible with reconfiguration techniques that change AD values on a per-router basis.

C. Generalization of Existing Techniques is not Easy

Ideally, we would like to adapt existing techniques for single-RD reconfigurations to account for route redistribution. In particular, the IGP reconfiguration technique defined in [6] only relies on FIB entries. Hence, it might be tempting to simply adapt it to multi-RD reconfigurations. Unfortunately, this is not straightforward.

First of all, there is a mismatch in the model and in the problem statement. In particular, in single-RD reconfigurations, reconfiguring a router can be considered an atomic operation that can be achieved by tweaking the AD value. With RR, we have at least an additional reconfiguration step to consider : activating or deactivating route redistribution. While AD influences what the routers select, RR influences what the routers announce, which might in turn influence the routing decisions of other routers. This distinction does not exist in reconfigurations within a single RD.

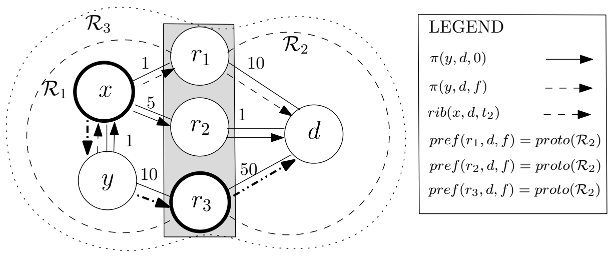


Fig. 4. Route redistribution can induce forwarding loops even when any combination of initial and final nexthops is loop-free.

More importantly, RR invalidates some fundamental properties that make single-RD reconfigurations possible. For example, the IGP reconfiguration technique defined in [6] is based on the property that a router is reconfigured if and only if it is using its final nexthops to any destination. RR invalidates this assumption because it can cause reconfigured routers to select nexthops that are not the final ones.

Consider Fig. 4, where RD $\mathcal{R}_3$ has to be split in $\mathcal{R}_1$ and $\mathcal{R}_2$ connected with RR. Assume that the initial and the final configurations comply with Assumptions 1 and 2. In the initial configuration, the shortest path from router y to destination d is $(y\ x\ r_2\ d)$. In the final configuration, routers inside $\mathcal{R}_1$ are not aware of link weights inside $\mathcal{R}_2$ and use a redistributed route to reach d. Therefore, each router selects the shortest path to the closest frontier router, e.g., router x uses r_1 as its next-hop. Observe that the next-hop of router y is x both in the initial and in the final configurations. For this reason, the technique in [6] would mistakenly assume that there is no ordering constraint between routers x and y. However, Fig. 4 shows that, if router r_3 has been reconfigured, then we cannot reconfigure router x before router y. In fact, router r_3 uses the routing instance in $\mathcal{R}_2$ to reach destination d and redistributes the route in $\mathcal{R}_1$. When router r_3 starts redistributing the route in $\mathcal{R}_1$, there is no other route towards destination d in $\mathcal{R}_1$. Hence, router x starts using y as its next-hop towards destination d, whereas router y keeps using x as next-hop since y still uses $\mathcal{R}_3$. The key observation here is that, despite the fact that router x has been reconfigured, it is not using its final nexthops.

IV. New Sufficient Conditions for Loop-Free Route Redistribution

We know from Sec. III-B that reconfigurations in multiple RDs can violate Assumption 2. Moreover, in intermediate routing configurations multiple routing instances could be configured in the same domain, and RDs might be nested.

To enable anomaly-free reconfigurations, we propose new sets of conditions both for disjoint (Section IV-A) and nested RDs (Section IV-B), that can replace Assumption 2.

A. Sufficient Conditions for Disjoint Routing Domains

We say that RDs are disjoint if frontier routers are the only routers belonging to more than one RD, as in the example in Fig. 3. Let Assumption 1 hold. We now replace Assumption 2 with the following conditions. We refer to a set of contiguous AD values as *AD interval*.

Condition 1 (Network-Wide Domain Preference): Each RD $\mathcal{R}$ is assigned a globally unique AD interval $interval(\mathcal{R})$. Each router sets the AD value of any routing instance p running in $\mathcal{R}$ to a value $AD(p) \in interval(\mathcal{R})$. Routes redistributed in $\mathcal{R}$ are assigned an AD higher than the highest value in $interval(\mathcal{R})$.

Condition 2 (Absence of Internal Loops): Forwarding paths internal to every RD do not contain loops.

Condition 1 is a relaxation of Assumption 2 in that it imposes a partial (rather than total) order on the AD values of routing instances. This relaxation requires adding Condition 2 on the forwarding paths internal to single domains.

Theorem 1: If Assumption 1 and Conditions 1 and 2 hold, the configuration is free from anomalies.

Proof: Assumption 1 ensures the absence of routing anomalies [4]. We now focus on forwarding loops. Let d be any destination, and let $\mathcal{O}_1, \ldots, \mathcal{O}_k$, with $k \geq 1$, be the origin RDs of d. Consider any time t after routing convergence and assume by contradiction that a forwarding loop L exists.

Since RDs are disjoint and Condition 2 holds, a loop cannot occur within a single RD. Hence, L must contain a sequence of frontier routers $(f_1 \ldots f_k)$ such that: i) each pair (f_i, f_{i+1}) belongs to a common RD $\mathcal{R}_i$; and ii) f_{i+1} prefers the path in another routing domain $\mathcal{R}_{i+1}$ over the path in $\mathcal{R}_i$, where subscripts are intended modulo k. For each pair (f_i, f_{i+1}), with $i = 1, \ldots, k-1$, Condition 1 ensures that AD values in $interval(\mathcal{R}_i)$ are greater than those in $interval(\mathcal{R}_{i+1})$. We write $interval(\mathcal{R}_i) > interval(\mathcal{R}_{i+1})$. By definition of L, we then have $interval(\mathcal{R}_i) > interval(\mathcal{R}_{i+1}) > \cdots > interval(\mathcal{R}_{i+k-1}) > interval(\mathcal{R}_i)$, that contradicts Condition 1. ■

B. Sufficient Conditions for Nested Routing Domains

We now consider the case in which i) we have nested RDs, i.e., some routers belong to more than one routing domain; ii) a single routing instance spans each RD; and iii) no route redistribution is configured between nested RDs. Observe that two routing domains cannot be partially overlapping: either one RD is a subset of the other (i.e., nested routing domains), or the two RDs are disjoint. These cases match the examples in Fig. 1 and Fig. 4, respectively. When dealing with nested domains, the notion of origin RD of a destination d is ambiguous because destination d could belong to multiple nested routing domains. We redefine the concept of origin RD for a destination d as follows. Let x be a router announcing d in multiple RDs. Then, each of the smallest disjoint routing domains in which router x announces destination d is an origin RD of d. For example, in Fig. 1, routing domain $\mathcal{R}_2$ is an origin RD of d, while $\mathcal{R}_3$ is not.

We now present sufficient conditions for this setting. Let Assumption 1 hold and let us replace Assumption 2 with the following conditions.

Condition 3 (Local Scope Preference): A router belonging to multiple nested RDs prefers the innermost RD.

Condition 4 (Limited Frontier Traversal): Let $\mathcal{R}$ be any RD and let r be any frontier router in $\mathcal{R}$. In all the routing instances used by at least one router in $\mathcal{R}$, the shortest path from r to any router in $\mathcal{R}$ must only include routers internal to $\mathcal{R}$.

Intuitively, Condition 3 induces a hierarchical preference of routing instances which derives from the hierarchy of RDs. Condition 4 states that at most one frontier router can be traversed in any forwarding path from a source to a destination in the same RD.

Theorem 2: If Assumption 1 and Conditions 3 and 4 hold, the configuration is free from anomalies.

Proof: The absence of routing anomalies is guaranteed by Assumption 1 [4]. Consider any destination d, and any time t after routing convergence. We now show that the forwarding paths from every $r \in \mathcal{R}$ to d contains no loop.

If $\mathcal{R}$ is an origin routing domain and $rib(r,d,t)$ includes only internal routers and frontier routers preferring the instance running in $\mathcal{R}$, then Condition 3 guarantees $\pi(r,d,t) = rib(r,d,t)$. By definition of RIB, this implies the absence of forwarding loops.

Otherwise, let f be the first frontier router (in order of appearance) in $rib(r,d,t)$ such that f prefers an instance running in a RD other than $\mathcal{R}$. Also, let o be a frontier router preferring an instance in an origin RD for d. Then, $\pi(r,d,t)$ can be written as the concatenation of three subpaths $P = (r \dots f)$, $Q = (f \dots o)$ and $R = (o \dots d)$, possibly with P empty (if $f = r$) or Q empty (if $f = o$). P and R are paths internal to a single RD, hence they are loop-free by Condition 3. Consider now Q. If $Q \neq \emptyset$, then f uses a route $\bar{R}$ in a RD $\mathcal{R}_f$, redistributed by another frontier router f_1. By Condition 4, no frontier routers other than f and f_1 are present in $\bar{R}$, i.e., $\bar{R}$ is internal to $\mathcal{R}_f$. By Condition 3, no loop can occur in $\mathcal{R}_f$, hence f_1 is eventually reached. Since Assumption 1 prevents any RD from being traversed twice, the same argument can be iterated until o is reached, proving that Q is loop-free. Since P, Q, and R contain no loops, then $\pi(r,d,t)$ is loop-free. ■

C. Enforcing the Sufficient Conditions in Practice

The conditions described in Sections IV-A and IV-B can be enforced by conveniently tweaking router configurations.

Different AD intervals can be assigned to each RD, as in Condition 1, by carefully configuring the AD of routing instances on all the routers. In order to ensure that redistributed routes have AD values higher than the highest AD in each RD, redistributed routes can be tagged by the router. Other routers can then apply the correct AD value based on the presence of tags (e.g., using route maps). While tags and route maps complicate router configurations, we argue that very similar mechanisms are needed to enforce Assumption 1 [14], and are already applied in some networks [3] to prevent cyclic route redistribution. Moreover, some protocols have specific features meant to natively distinguish redistributed routes (e.g., OSPF external routes).

Condition 2 always holds in the presence of a single routing instance per RD. If multiple instances are present in the same RD, a very simple way to comply with Condition 2 is to make sure that, for each destination, a single instance is the most preferred by all routers in the RD.

Condition 3 can be ensured by careful AD settings.

Finally, Condition 4 imposes constraints on shortest paths as computed by routing instances used inside each RD. Hence, it imposes restrictions on the internal weights and topologies of RDs. Realistic configurations following best practices in network design are likely to comply with such restrictions. For example, in hierarchical topologies like those commonly used in data centers [19] and in Point-of-Presences [20], a frontier can be placed at each level of the hierarchy. More in general, a routing domain $\mathcal{R}$ can be configured to comply with Condition 4 by i) setting the weight of the links incident on frontier routers to a value greater than the highest weight of links internal to $\mathcal{R}$; and ii) setting the weight of links (if any) between $x \in \mathcal{R}$ and $y \notin \mathcal{R}$ to a value greater than any other link weight in the network (including links incident on frontier routers) so that no shortest path can possibly traverse those links.

V. Loop-Free Reconfigurations

In this section, we propose safe procedures to reconfigure multi-RD networks. We assume that the initial and final configurations include only disjoint RDs, as it is typically the case [3]. However, nested routing domains can appear during a reconfiguration. Sections V-A and V-B address reconfigurations in single-RD and multi-RD networks, respectively.

A. Revisiting Single-RD Techniques

We have shown how existing techniques are hard to adapt to multi-RD networks, even when the reconfiguration affects a single-RD (see, e.g., the example reported in Fig. 3). We now revisit those techniques, restricting to those that provably guarantee the absence of anomalies in networks with a single RD. We distinguish between *AD-preserving* and *AD-changing* techniques.

AD-preserving techniques ensure that the AD of all routing instances is left untouched throughout the reconfiguration. Hence, if the initial and final configurations comply with Assumptions 1 and 2, AD-preserving techniques guarantee that any intermediate configuration also complies with those assumptions. Previous results [4], [5] then ensure the absence of routing and forwarding anomalies during the reconfiguration.

Unfortunately, not all AD-preserving techniques are applicable on today's routers (e.g., [21], [22], [23], [24]). Others [9], [25], [26] can only be used in a limited set of reconfiguration scenarios, i.e., link weight changes. AD-preserving techniques cannot be used to introduce or remove routing hierarchies, to activate or deactivate protocol-specific features (like route summarization), or to introduce new routing instances. The latter limitation also prevents them from being applied in

Single-RD Reconfiguration Procedure

Let $\mathcal{R}$ be the RDto be reconfigured. Let p_i and p_f be the routing instances running the initial and the final configurations, respectively.

1) force the interval of AD values assigned to $\mathcal{R}$ to include $[AD(p_i) - 1, AD(p_i) + 1]$.
2) introduce p_f with $AD(p_f) = AD(p_i) + 1$ on all the routers in $\mathcal{R}$ (in any order).
3) set $AD(p_f) = AD(p_i) - 1$ on every router in $\mathcal{R}$ following an ordering computed as in [6].
4) remove p_i from each router in $\mathcal{R}$ (in any order).

Fig. 5. A provably correct procedure for single-RD reconfigurations.

reconfigurations involving splitting, merging or reshaping routing domains.

AD-changing techniques, on the contrary, modify AD values progressively in the network. The new configuration is introduced as a new routing instance and the AD values are progressively changed in order to activate the new instance. By relying on route tags and route-maps, as described in Section IV-C, Assumption 1 can be preserved during the entire reconfiguration process. Unfortunately, Assumption 2 cannot be enforced. However, we can adapt those techniques to make them compliant with Conditions 1 and 2 throughout the reconfiguration process. In particular, we focus on the seamless IGP reconfiguration technique defined in [6].

Let $\mathcal{R}$ be the routing domain to be reconfigured, and let p_i and p_f be the routing instances respectively running the initial and the final configurations on routers in $\mathcal{R}$. In every intermediate configuration, Condition 2 is guaranteed on any routing domain $\mathcal{R}_i \neq \mathcal{R}$ because the initial configuration is anomaly-free and the configuration of routers in $\mathcal{R}_i$ is never changed. Moreover, the reconfiguration technique in [6] guarantees no reconfiguration loop when applied to a single-RD network, so Condition 2 also holds for $\mathcal{R}$.

On the contrary, the technique in [6] is not guaranteed to preserve Condition 1. In fact, when introducing p_f with an AD value which is higher than the one assigned to p_i, the AD value of p_f could violate Condition 1. For example, if $AD(p_i) = \max(interval(\mathcal{R}))$, then we cannot assign to p_f an AD value in $interval(\mathcal{R})$ higher than p_i. Luckily, as long as there is at least one unused AD value, it is always possible to rearrange the AD intervals, and make an arbitrary AD value unused. In fact, the AD value of a routing instance can always be increased by one without invalidating Assumption 1 and Conditions 1 and 2, if the new value was previously unused.

Fig. 5 shows an extension of the technique in [6] that preserves Condition 1 in each intermediate configuration, by

- using $AD(p_i)+1$ and $AD(p_i)-1$ as the initial and final AD for p_f, respectively; and
- rearranging the AD intervals in a preliminary step to make $AD(p_i) + 1$ and $AD(p_i) - 1$ available before the reconfiguration.

Theorem 3: The Single-RD Reconfiguration Procedure in Fig. 5 does not incur reconfiguration anomalies.

Routing Domain Splitting Procedure

Let $\mathcal{R}_U$ be a routing domain to be split in two routing domains $\mathcal{R}_1$ and $\mathcal{R}_2$. Let p_u, p_1 and p_2 be the routing instances running in $\mathcal{R}_U$, $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.

1) (if needed) enforce Assumption 1 by tagging redistributed routes and adding convenient route maps.
2) (if needed) reconfigure $\mathcal{R}_U$ to enforce Condition 4 on $\mathcal{R}_1$ and $\mathcal{R}_2$.
    a) compute new weights to be applied to p_u, p_1, and p_2 (see Section IV-C)
    b) change weights in p_u using the procedure in Fig. 5
3) reconfigure routers in $\mathcal{R}_1$ to prefer p_1 using the procedure in Fig. 5
4) reconfigure routers in $\mathcal{R}_2$ to prefer p_2 using the procedure in Fig. 5
5) activate route redistribution from and to $\mathcal{R}_1$ and $\mathcal{R}_2$. Redistributed routes have lower AD than p_u.
6) remove p_u on all the routers in $\mathcal{R}_U$ (in any order).

Fig. 6. Provably correct procedure for routing domain splitting scenarios.

Proof: Assumption 1 holds in the initial configuration and is never invalidated during the reconfiguration. Moreover, the procedure ensures that all intermediate configurations comply with Conditions 1 and 2. Theorem 1 completes the proof. ■

B. Safe Procedures for Multi-RD Reconfigurations

We now consider reconfigurations that change the organization of the network in RDs. We focus on RD splitting and merging reconfigurations. Any other re-organization of routing domains can be performed by combining splitting and merging.

Fig. 6 describes the procedure that we propose to split an RD $\mathcal{R}_U$ in two RDs $\mathcal{R}_1$ and $\mathcal{R}_2$. For the sake of simplicity, we assume that only one routing instance, in a flat configuration mode, is run in each routing domain. The procedure consists in three main phases. In the first phase, we enforce Assumption 1 by using tags and route-maps. In the second phase, routers in future RDs $\mathcal{R}_1$ and $\mathcal{R}_2$ are reconfigured in order to enforce Conditions 3 and 4. In Step 2, Condition 4 is imposed on $\mathcal{R}_U$ with the single-RD reconfiguration discussed in the previous section (note that routing domains $\mathcal{R}_1$ and $\mathcal{R}_2$ do not exist yet). Then, in Steps 3 and 4, we progressively introduce the routing instances that will run in RD $\mathcal{R}_1$ and $\mathcal{R}_2$. Finally, the third phase activates route redistribution on frontier routers.

Intuitively, the procedure preserves Assumption 1 throughout the process. Conditions 1 and 2 hold in the second phase, and Conditions 3 and 4 are preserved in the third phase. This makes the procedure provably free from anomalies.

Theorem 4: The procedure in Fig. 6 does not incur reconfiguration anomalies.

Proof: Assumptions 1 and 2 hold at Step 1 as they hold in the initial configuration by hypothesis. This guarantees the absence of anomalies [4], [5]. Moreover, this first step enforces Assumption 1 in each intermediate configuration, independently from AD settings.

Routing Domain Merging Procedure

Let $\mathcal{R}_1$ and $\mathcal{R}_2$ be two RDs to be merged in $\mathcal{R}_U$. Let p_u, p_1 and p_2 be the routing instances running in $\mathcal{R}_U$, $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.

1) (if needed) enforce Assumption 1 by tagging redistributed routes and adding convenient route maps.
2) (if needed) reconfigure $\mathcal{R}_1$ and $\mathcal{R}_2$ to enforce Condition 4.
   a) compute new link weights compliant with Condition 4 (see Section IV-C)
   b) apply the procedure in Fig. 5 on $\mathcal{R}_1$ and $\mathcal{R}_2$
3) configure p_u on all the routers in $\mathcal{R}_U$, with an AD higher than every route (either redistributed or non-redistributed) in both $\mathcal{R}_1$ and $\mathcal{R}_2$
4) de-activate route redistribution from and to $\mathcal{R}_1$ and $\mathcal{R}_2$.
5) reconfigure routers in $\mathcal{R}_1$ to prefer p_u using the procedure in Fig. 5.
6) reconfigure routers in $\mathcal{R}_2$ to prefer p_u using the procedure in Fig. 5.
7) remove p_1 and p_2 on all the routers in $\mathcal{R}_U$ (in any order).

Fig. 7. Provably correct procedure for routing domain merging scenarios.

In Step 2, we still have a single RD $\mathcal{R}_U$. Since this step is carried out using the procedure in Fig. 5, Theorem 3 guarantees the absence of reconfiguration anomalies.

Step 3 is a reconfiguration of nested RDs. However, route redistribution between $\mathcal{R}_1$ and $\mathcal{R}_2$ is not activated yet. Moreover, since the link weights are the same in both instances, each router uses the same next-hops for the same destinations, independently of the routing instance it prefers among these in $\mathcal{R}_1$ and in $\mathcal{R}_U$. This ensures the absence of loops for every router reconfiguration ordering. The same argument applies to Step 4. During Step 5, Conditions 3 and 4 are enforced, and Theorem 2 ensures the absence of anomalies.

With respect to other disjoint RDs (if any) in the network, Conditions 1 and 2 hold throughout all the previous steps, hence Theorem 1 ensures the absence of anomalies.

Finally, when Step 5 is completed, all routers in $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) use $\mathcal{R}_1$ (resp. $\mathcal{R}_2$) to reach all destinations in the network, using redistributed routes only for destinations outside $\mathcal{R}_1$ (resp. $\mathcal{R}_2$). Thus, $\mathcal{R}_U$ is not used anymore, hence it can be removed safely at Step 6. ■

Symmetric considerations apply to scenarios in which two RDs $\mathcal{R}_1$ and $\mathcal{R}_2$ have to be merged in a single RD $\mathcal{R}_U$. Fig. 7 shows a procedure to perform the reconfiguration.

Theorem 5: The procedure in Fig. 7 does not incur reconfiguration anomalies.

Proof: Assumptions 1 and 2 hold at Step 1 as they hold in the initial configuration by hypothesis. This guarantees the absence of anomalies [4], [5]. Moreover, this first step enforces Assumption 1 in each intermediate configuration, independently from AD settings.

Step 2 consists of a single-RD reconfiguration on routers in $\mathcal{R}_1$ and $\mathcal{R}_2$. Since this step is carried out using the procedure in Fig. 5, Theorem 3 guarantees the absence of reconfiguration anomalies.

When $\mathcal{R}_U$ is introduced at Step 3, it is not used by any router in the network because of its AD. Hence, this step does not affect configuration correctness.

At the end of Step 3, the configuration complies with both Conditions 3 and 4. Those conditions are preserved throughout Step 4, hence Theorem 2 ensures the absence of anomalies. In steps 5 and 6, the forwarding paths of any router in the network is ensured not to change, since link weights do not change from p_1 and p_2 to p_u. Moreover, in all the previous steps, Conditions 1 and 2 hold, hence Theorem 1 ensures the absence of anomalies involving routers outside $\mathcal{R}_U$.

Finally, no anomaly can occur during Step 7, as routers in $\mathcal{R}_U$ do not use p_1 and p_2 for any destination. ■

VI. Use Case

In this section, we show that our techniques enable lossless reconfiguration *in real networks using real routers*. We created a realistic reconfiguration case study based on the pan-european research network Geant which is composed of 36 routers and 53 links. The full network topology is available at [17]. We deliberately removed stub routers from the topology since they cannot cause any routing anomaly. Also, we assigned link weights to be inversely proportional to bandwidth. We emulated this network on an high-end server using virtualized Cisco IOS images. By using virtual IOS images, we rely on exactly the same control plane (i.e., routing protocols) as real routers. The only difference with a lab of real routers is the forwarding performance and the sharing of CPU between different emulated routers.

For our reconfiguration test case, we divided the Geant network into two routing domains: the backbone RD and the south east RD, composed of 18 and 9 routers, respectively. The initial RD uses OSPF, while the final backbone and south-east RD uses OSPF and IS-IS, respectively. We selected frontier routers based on their geographical location and their links, hereby complying with the best current practices [8].

We implemented the procedures in Fig. 6 by scripting commands on the emulated routers. We ran two experiments. In the first one, we reconfigured the network using the procedure in Fig. 6. In the second one, we used a variant of the procedure (called "naive procedure") in which route redistribution is activated *before* reconfiguring the routers, i.e., we executed Step 4 of Fig. 6 *before* Step 2. In both experiments, we enforced Assumptions 1 and 2 to make sure that the initial and the final configurations are correct and loop-free.

Each router pinged all other routers every second during the entire reconfiguration. These probes allow us to detect routing anomalies that affect the data plane without consuming too much CPU. We repeated each experiment 10 times to improve statistical significance.

Fig. 8 shows the evolution of the number of lost probes at each step of the splitting procedure. The results were similar across all repetitions. The naive procedure created loops that caused up to 23% of all sent probes to be lost between steps 32 and 44. This is not surprising as these reconfiguration steps consist in reconfiguring routers in the south east RD to prefer

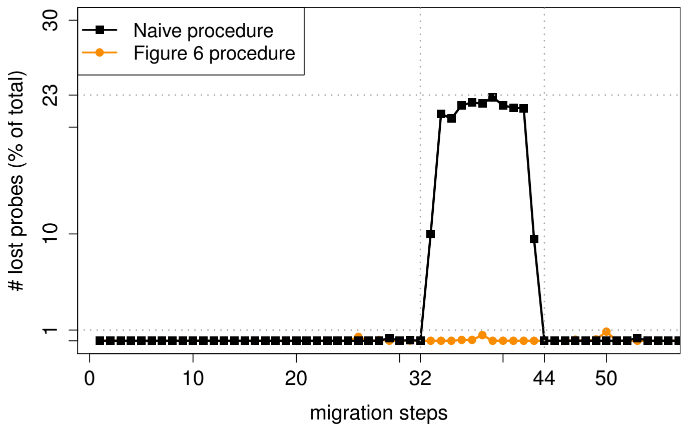


Fig. 8. Comparison between a naive procedure to split a routing domain and the procedure in Fig. 6. In this experiment, using the naive approach results in about 23% of packet loss during 12 consecutive steps (≈ 20% of the migration time). In contrast, virtually no traffic is lost using our procedure.

redistributed routes. At this point, forwarding loops appeared because of the presence of redistributed routes. In contrast, our procedure incurred *no connectivity disruption*. Observe that a few iterations underwent a few losses, namely one or two packets per source-destination pair, summing up to less than 1% of the probes. We double checked the affected steps and verified the absence of persistent loops. We suspect that those few lost packets are due to the virtual environment.

VII. Conclusions

In this paper, we study routing reconfigurations in networks with multiple routing domains (RDs) connected via route redistribution. We show that neither the existing theory on route redistribution nor the state-of-the-art reconfiguration techniques can be applied without causing significant routing anomalies. Unfortunately, this holds even when the desired reconfiguration consists of configuration changes that affect a single routing domain in a multi-RD network.

We have extended the existing theory with new sufficient conditions for routing safety, and from these conditions, we have devised provably correct procedures for reconfiguring networks with multiple RDs. We performed an evaluation of these procedures using realistic network topologies and real router configurations. The results demonstrate the power of our approach in eliminating all the forwarding loops that otherwise cannot be avoided today. Furthermore, our new sufficient conditions can serve as guidelines for making route redistribution configurations more easily evolvable, i.e., through the proposed reconfiguration procedures.

VIII. Acknowledgements

This work has been partially supported by the European Commission's Seventh Framework Programme (FP7/2007-2013) grant no. 317647 (Leone), and by the ARC grant no. 13/18-054 from Communauté française de Belgique.

References

[1] D. A. Maltz, G. Xie, J. Zhan, H. Zhang, G. Hjálmtýsson, and A. Greenberg, "Routing design in operational networks: a look from the inside," in *Proc. SIGCOMM*, 2004.

[2] Y.-W. Sung, X. Sun, S. Rao, G. Xie, and D. Maltz, "Towards systematic design of enterprise networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 3, pp. 695–708, Jun. 2011.

[3] F. Le, G. Xie, D. Pei, J. Wang, and H. Zhang, "Shedding light on the glue logic of the internet routing architecture," in *Proc. SIGCOMM*, 2008.

[4] F. Le and G. Xie, "On Guidelines for Safe Route Redistributions," in *Proc. INM*, 2007.

[5] F. Le, G. Xie, and H. Zhang, "Instability Free Routing: Beyond One Protocol Instance," in *Proc. CoNext*, 2008.

[6] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Lossless Migrations of Link-State IGPs," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1842–1855, 2012.

[7] I. Pepelnjak, "Changing the Routing Protocol in Your Network," 2007. http://stack.nil.com/ipcorner/ChangingRoutingProtocol

[8] G. Herrero and J. van der Ven, *Network Mergers and Migrations: Junos Design and Implementation.* Wiley, 2010.

[9] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful Network State Migrations," *Trans. on Netw.*, vol. 19, no. 4, pp. 1097–1110, 2011.

[10] R. Keralapura, C.-N. Chuah, and Y. Fan, "Optimal Strategy for Graceful Network Upgrade," in *Proc. INM*, 2006.

[11] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: live router migration as a network-management primitive," in *Proc. SIGCOMM*, 2008.

[12] E. Keller, J. Rexford, and J. Van Der Merwe, "Seamless BGP migration with router grafting," in *Proc. NSDI*, 2010.

[13] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, "Abstractions for network update," in *Proc. SIGCOMM*, 2012.

[14] F. Le, G. Xie, and H. Zhang, "Understanding route redistribution," in *Proc. ICNP*, 2007.

[15] ——, "Theory and new primitives for safely connecting routing protocol instances," in *Proc. SIGCOMM*, 2010.

[16] A. Alim and T. Griffin, "On the interaction of multiple routing algorithms," in *Proc. CoNEXT*, 2011.

[17] "GEANT Backbone Topology," http://geant3.archive.geant.net/Network/NetworkTopology/pages/home.aspx, 2012.

[18] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. SIGCOMM*, 2002.

[19] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. SIGCOMM*, 2008.

[20] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "Loop-Free Alternate (LFA) applicability in Service Provider (SP) networks," RFC 6571, June 2012. http://tools.ietf.org/html/rfc6571

[21] R. Alimi, Y. Wang, and R. Yang, "Shadow configuration as a network management primitive," in *Proc. SIGCOMM*, 2008.

[22] P. Francois and O. Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *Trans. on Netw.*, vol. 15, no. 6, pp. 1280–1932, 2007.

[23] J. Fu, P. Sjodin, and G. Karlsson, "Loop-Free Updates of Forwarding Tables," *Trans. on Netw. and Serv. Man.*, vol. 5, no. 1, pp. 22–35, 2008.

[24] L. Shi, J. Fu, and X. Fu, "Loop-Free Forwarding Table Updates with Minimal Link Overflow," in *Proc. ICC*, 2009.

[25] P. Francois, M. Shand, and O. Bonaventure, "Disruption-free topology reconfiguration in OSPF Networks," in *Proc. INFOCOM*, 2007.

[26] F. Clad, P. Mérindol, S. Vissicchio, P. Francois, and J.-J. Pansiot, "Graceful Router Updates in Link-State Protocols," in *Proc. ICNP*, 2013.