Lossless Migrations of Link-State IGPs

Laurent Vanbever, *Student Member, IEEE*, Stefano Vissicchio, Cristel Pelsser, Pierre Francois, *Member, IEEE*, and Olivier Bonaventure, *Member, IEEE*

***Abstract*—Network-wide migrations of a running network, such as the replacement of a routing protocol or the modification of its configuration, can improve the performance, scalability, manageability, and security of the entire network. However, such migrations are an important source of concerns for network operators as the reconfiguration campaign can lead to long, service-disrupting outages. In this paper, we propose a methodology that addresses the problem of seamlessly modifying the configuration of link-state Interior Gateway Protocols (IGPs). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition and the removal of routing hierarchy in a running IGP, and the replacement of one IGP with another. We prove that a strict operational ordering can guarantee that the migration will not create any service outage. Although finding a safe ordering is NP-complete, we describe techniques that efficiently find such an ordering and evaluate them using several real-world and inferred ISP topologies. Finally, we describe the implementation of a provisioning system that automatically performs the migration by pushing the configurations on the routers in the appropriate order while monitoring the entire migration process.**

***Index Terms*—Configuration, design guidelines, Interior Gateway Protocol (IGP), migration, reconfiguration, summarization.**

I. Introduction

AMONG all network routing protocols, link-state Interior Gateway Protocols (IGPs), like IS-IS [1] and OSPF [2], play a critical role. An IGP enables end-to-end reachability between any pair of routers within the network of an autonomous system (AS). Many other routing protocols, like BGP, LDP, or PIM, rely on an IGP to work. As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfigurations [3].

Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Restarting the network with the new configurations does not work since most of the networks carry traffic 24/7. Therefore, IGP migrations have to be performed gradually, while the network is running. Such operations can lead to significant traffic losses if they are not handled with care. Unfortunately, network operators typically lack appropriate tools and techniques to seamlessly perform large, highly distributed changes to the configuration of their networks. They also experience difficulties in understanding what is happening during a migration since complex interactions may arise between upgraded and nonupgraded routers. Consequently, as confirmed by many private communications with operators, large-scale IGP migrations are often avoided until they are absolutely necessary, thus hampering network evolvability and innovation.

Most of the time, network operators target three aspects of the IGP when they perform large-scale migrations. First, they may want to replace the current protocol with another. For example, operators may want to migrate to an IGP that provides more guarantees against security attacks [4], [5], or that allows to integrate new equipments that are not compliant with the adopted one [6], or that is not dependent on the address family (e.g., OSPFv3, IS-IS), e.g., to run only one IGP to route both IPv4 and IPv6 traffic [7], [5]. Second, when the number of routers exceeds a certain critical mass, operators often introduce a hierarchy within their IGP to limit the control-plane stress [8], [9]. Another reason operators introduce hierarchy is to have more control on route propagation by tuning the way routes are propagated from one portion of the hierarchy to another [3]. On the contrary, removing a hierarchy might be needed to better support some traffic engineering extensions [10]. Third, network operators also modify the way the IGP learns or announces the prefixes by introducing or removing route summarization. Route summarization is an efficient way to reduce the number of entries in the routing tables of the routers as IGP networks can currently track as many as 10 000 prefixes [11]. Route summarization also helps improving the stability by limiting the visibility of local events. Actually, some IGP migrations combine several of these scenarios, such as the migration from a hierarchical OSPF to a flat IS-IS [4]. Finally, operators may be forced to revert back to a previous IGP configuration to meet technical requirements [12].

In this paper, we aim at enabling *seamless IGP migrations*, that is, progressive modifications of the IGP configuration of a running network without losing packets. Our contribution is manifold. First, we develop a generic IGP model and use it to analyze in detail various scenarios of link-state IGP migrations. In particular, we show that long-lasting forwarding loops can appear, both theoretically and practically, when changes are made to the IGP hierarchy and when route summarization is introduced or removed. Second, we introduce a methodology that enables seamless IGP migration while minimizing the number of reconfigurations per router. Our methodology leverages the fact that several IGP processes can run at the same time on a router. Each IGP process is assigned with an *administrative distance* (AD), and the IGP process with

Manuscript received October 24, 2011; accepted January 22, 2012; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z. M. Mao. This work was supported in part by Alcatel-Lucent. The work of L. Vanbever was supported by an FRIA scholarship. The work of S. Vissicchio was supported in part by the MIUR PRIN Project ALGODEEP.

L. Vanbever and O. Bonaventure are with the Institute of Information and Communication Technologies, Electronics and Applied Mathematics (ICTEAM), Université catholique de Louvain, Louvain-la-Neuve 1348, Belgium (e-mail: laurent.vanbever@uclouvain.be; Olivier.Bonaventure@uclouvain.be).

S. Vissicchio is with the Dipartimento di Informatica e Automazione, Universita Roma Tre, Rome 00146, Italy (e-mail: vissicch@dia.uniroma3.it).

C. Pelsser is with Innovation Institute, Internet Initiative Japan (IIJ), Tokyo 101-0051, Japan (e-mail: cristel@iij.ad.jp).

P. Francois is with the IMDEA Institute, Madrid 28918, Spain (e-mail: pierre.francois@imdea.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2012.2190767

the lowest distance controls the forwarding. The reconfiguration consists in introducing the final IGP configuration at each router with a high AD, waiting for its convergence, and lowering the distance router by router so that it is used for forwarding. We show that, in real-world networks, it is possible to find an ordering in which to reconfigure the routers while guaranteeing no forwarding loop. Although finding such an ordering is an NP-complete problem, we propose algorithms and heuristics, and we show their practical effectiveness on several ISP networks. We also show how our techniques can deal with advanced reconfiguration scenarios by extending them to support link failures. Finally, we describe the design and the evaluation of a provisioning system that automates the whole migration process according to our methodology. Our system generates router configurations, assesses the proper state of the network, and updates all the routers in an appropriate sequence. As shown in our evaluation and case study, such a provisioning system enables faster and seamless IGP migrations while avoiding human errors due to manual design and application of new configurations on routers.

This paper extends the work that appeared in [13] with additional material, notably including: 1) the generalization of our approach, based on a hybrid per-router and per-destination migration, in case any per-router reconfiguration causes packet losses; 2) a study of how to achieve seamless IGP migrations in presence of link failures; and 3) the proof that deciding if a per-router ordering exists is NP-complete.

The rest of the paper is structured as follows. Section II provides a background on link-state IGPs and presents our abstract model. Section III formalizes the IGP migration problem and describes the migration scenarios we tackle. Section IV presents our methodology. Section V proposes algorithms to compute a loop-free migration ordering. Section VI presents our implementation. Section VII evaluates our migration techniques on both inferred and real-world topologies. Section VIII explains how to deal with network failures. Section IX defines design guidelines that make IGP migrations easier. Section X presents related work. Finally, Section XI concludes the paper.

II. Link-State Interior Gateway Protocols

In this section, we provide some background on link-state IGPs. Also, we present the model we use in the paper.

A. Background

An IGP is a protocol that routers use to decide how to forward packets within an AS. IGPs are divided in two main families: distance-vector and link-state protocols. Although some enterprise networks still use distance-vector protocols, most ISPs and large enterprises deploy link-state IGPs, namely OSPF [2] or IS-IS [1]. In this paper, we focus on network-wide migrations of link-state IGPs.

Link-state IGPs can be configured either in a flat or in a hierarchical *mode*. In flat IGPs, every router is aware of the entire network topology and forwards IP packets according to the shortest paths toward their respective destinations. In hierarchical IGPs, routers are not guaranteed to always prefer the shortest paths. Hierarchical IGP configurations break the whole topology into a set of *zones* (called areas in OSPF and levels in IS-IS), which we denote as $B, Z_1, \ldots, Z_k$. B is a special zone, called *backbone*, that connects all the other *peripheral zones* together, such that packets from a router in the network to a destination inside a different zone always traverse the backbone. IGP routers establish adjacencies over physical links in order to exchange routing information. Each adjacency belongs to only one zone. By extension, we say that a router is in a zone if it has at least one adjacency in that zone. We call *internal routers* the routers that are in one zone only. The *Zone Border Routers* (ZBRs) (e.g., ABRs in OSPF and L1L2 systems in IS-IS) are the routers that are in more than one zone, among which one must be the backbone. Both internal routers and ZBRs *prefer intrazone over interzone paths*. This means that to choose the path on which to forward packets toward a certain destination, each router prefers a path traversing only one zone over a path traversing more than one zone, no matter what is the length of the two paths.

Moreover, in hierarchical IGPs, ZBRs can be configured to perform *route summarization*. In this configuration, ZBRs hide the internal topology of a zone Z to routers in different zones, advertising aggregated prefixes outside Z. In practice, they announce their ability to reach groups of destinations with paths of a certain length. The length announced by a ZBR is the same for all the destinations in an aggregated prefix, and either it is customly configured, or it is decided on the basis of the actual lengths of the preferred paths toward that destinations (e.g., picking the highest one [2]).

B. Abstract Model for Link-State IGPs

In this section, we aim at capturing IGP configurations and forwarding behavior of routers in a model that abstracts protocol-specific details. Transient IGP behavior is not modeled since we ensure that both the initial and the final IGPs have converged before starting the migration process (see Section IV).

We formally define an *IGP configuration* as a tuple (p, G, D, w, m). The tuple reflects configuration knobs available to operators. p is the identifier of an IGP protocol, e.g., OSPF or IS-IS. m is the mode in which the protocol is configured, namely flat or hierarchical. $G = (V, E)$ is the *logical graph*, i.e., a directed graph that represents the IGP adjacencies among routers participating in p. Each node in V represents an IGP router, and each edge in E represents an adjacency between the two routers. Edges are labeled with the name of the zones to which they belong. Moreover, the function $w : E \to \mathbb{N}$ associates a positive integer, called *weight*, to each edge in G. Finally, $D \subseteq V$ is the set of IGP *destinations* for traffic that flows in the network. We associate each destination to a single node in G, assuming that each IP prefix is announced by one router only. This assumption is without loss of generality, as we can use virtual nodes and G can be transformed in a multigraph in order to model peculiarities of the considered IGP. For example, consider how to model the binding of each interface to a given area or the redistribution of external prefixes in OSPF. For each router r that coincides with a traffic destination, we can add a virtual node r_j for each OSPF area j in which r participates, and a virtual node r_{ext} for external destinations injected by r in the IGP. For each r_j, only one edge $(r\ r_j)$, belonging to zone Z_j and weighted 1, is added to the graph. One edge e_j for each OSPF area j is also added between r and r_{ext}. Each e_j is such that it is labeled as belonging to zone Z_j and $w(e_j) = 1$. The destination set D will contain virtual nodes only. Similarly, virtual nodes can be used to model IP prefixes announced by more than one IGP router.

Packets destined to one router $d \in D$ follow forwarding paths. A forwarding path, or simply *path*, P from s to d is a path $P = (s\ r_1 \dots r_k\ d)$ on G in which r_i, with $i = 1, \dots, k$, are routers traversed by the traffic flow. Several forwarding paths can simultaneously be used for the same pair (s, d), e.g., in case of Equal Cost Multi-Path (ECMP). The weight of a path is the sum of the weights of all the links in the path.

According to the IGP configuration, each router chooses its preferred path toward each destination and forwards packets to the next-hops in such preferred paths. To model this behavior, we define the *next-hop* function nh and the *actual path* function $\pi(u, d, t)$. Both functions are derived from the IGP configuration of a network, i.e., an IGP configuration univocally determines the next-hop and the actual path functions. We denote the set of successors (next-hops) of u in the paths router u uses at time t to send traffic destined to destination d by $\mathrm{nh}(u, d, t)$. Notice that $|\mathrm{nh}(u, d, t)|$ is not guaranteed to be equal to 1 since routers can use multiple paths to reach the same destination (e.g., in presence of ECMP). The paths actually followed by packets sent by u toward d at time t can be computed as a function π: $\pi(u, d, t)$ is the set of paths resulting from a recursive concatenation of next-hops. More formally, $\pi(u, d, t)$ is a function that associates to each router u the set of paths $\{(v_0\ v_1\ \cdots\ v_k)\}$, such that $v_0 = u$, $v_k = d$ and $\forall i \in \{0, \dots, k-1\} v_{i+1} \in \mathrm{nh}(v_i, d, t)$. Note that the actual path function does not always coincide with the preferred path of each router since deflections can happen in the middle of a path [14]. A series of deflections can even build a forwarding loop, as shown in different examples in Section III-A. More formally, there exists a *forwarding loop*, or simply a *loop*, for a given destination d at a given time t if $\exists r :\ \pi(r, d, t) = (r\ v_0\ \dots\ v_j\ r)$, with $j \geq 0$.

By appropriately tuning the next-hop function, our model is able to capture specific details of IGP configurations such as the corresponding forwarding rules in hierarchical and flat mode and route summarization. In Section III-A, we provide some examples of next-hop functions, actual path functions, and migration loops in different migration scenarios.

III. IGP Migration Problem

In this section, we study the problem of seamlessly migrating a network from one IGP configuration to another. Both configurations are provided as input (i.e., by network operators) and, by definition, are loop-free.

Problem 1: Given a unicast IP network, how can we replace an initial IGP configuration with a final IGP configuration quickly, with minimal configuration changes and without causing any forwarding loop?

Assuming no network failures and no congestion, solving this problem leads to seamless migrations. Observe that our approach reduces the opportunities for failures during the migration process because of its time efficiency. Furthermore, in Section VIII, we show how to extend our techniques to provide guarantees even in case of network failures. Similar extensions may be used to avoid congestion during the migration. We plan to fully investigate congestion-free migration techniques in future work. However, we argue that congestion issues are less critical, as they can be strongly mitigated by performing the migration during time slots in which traffic is low. Also, large ISPs are normally overprovisioned [15], further reducing the risk of congestion.

TABLE I
IGP Migration Scenarios

scenario	IGP configuration changes
protocol	protocol replacement
flat2hier	zones introduction
hier2flat	zones removal
hier2hier	zones reshaping
summarization	summarization introduction/removal

In this paper, we focus on issues generated by the IGPs themselves, while leaving migration issues due to the presence of additional routing protocols in the network (e.g., BGP) to future work. In the rest of the paper, we call *router migration* the replacement of the initial next-hop function $\mathrm{nh}_{\mathrm{init}}$ with the final next-hop function $\mathrm{nh}_{\mathrm{final}}$ on a given router. Formally, we define the operation of *migrating a router* r at a certain time $\bar{t}$ as the act of configuring the router such that $\mathrm{nh}(r, d, t) = \mathrm{nh}_{\mathrm{final}}(r, d),\ \forall d \in D$ and $\forall t > \bar{t}$. We call *router migration ordering* the ordering in which routers are migrated. A network migration is completed when all routers have been migrated. In this paper, we focus on per-router migrations in which all the destinations are migrated at the same time in order to limit the number of configuration changes and to minimize the migration duration. However, such an ordering might not always exist as described in Section III-A. Only in such cases, we compute and apply separate migration orderings for the troublesome destinations (see Section IV). Results of our evaluation (see Section VII) suggest that troublesome destinations are zero or few in realistic topologies.

Throughout the paper, we focus our attention on *migration loops*, that is, loops arising during an IGP migration because of a nonsafe router migration ordering. Migration loops are not protocol-dependent, and are more harmful than loops that arise during protocol convergence as they last until specific routers are migrated (e.g., see Section III-A). Observe that if $\mathrm{nh}_{\mathrm{init}} = \mathrm{nh}_{\mathrm{final}}$, the π function does not change either, hence any router migration ordering is ensured to be loop-free.

A. IGP Migration Scenarios

Table I presents the IGP migration scenarios we address in this paper. We believe that those scenarios cover most of the network-wide IGP migrations that real-world ISPs can encounter. Each scenario concerns the modification of a specific feature of the IGP configuration. Moreover, different scenarios can be combined if more than one feature of the IGP configuration has to be changed. We do not consider the change of link weights as a network-wide migration. Indeed, traffic matrices tend to be almost stable over time [16], and ISPs typically change the weights of a few links at a time. Moreover, effective techniques have already been proposed for the graceful change link weights [17]–[21]. Nevertheless, our generalized model and the techniques we present in Section V are also applicable to reconfigure link weights. Furthermore, since the addition and the removal of links and devices can be modeled as a change of some link weights from an infinite to a finite value or vice versa, our approach can also be used to guarantee no packet loss during topological changes.

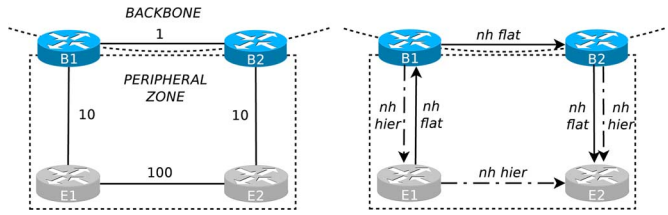


Fig. 1. Bad square gadget. When the IGP hierarchy is modified, a given migration ordering is needed between $B1$ and $E1$ to avoid forwarding loops.

In the following, we describe the issues that must be addressed in each migration scenario we target, using the notation introduced in Section II-B.

1) Protocol Replacement: This migration scenario consists of replacing the running IGP protocol, but keeping the same nh function in the initial and in the final configurations. A classical example of such a scenario is the replacement of an OSPF configuration with the corresponding IS-IS configuration [3]. Since the nh function is the same in both IGPs, routers can be migrated in any order without creating loops.

2) Hierarchy Modification: Three migration scenarios are encompassed by the modification of the IGP hierarchy. First, a flat IGP can be replaced by a hierarchical IGP by introducing several zones. Second, a hierarchical IGP can be migrated into a flat IGP by removing peripheral zones and keeping only one zone. Third, the structure of the zone in a hierarchical IGP can be changed, e.g., making the backbone bigger or smaller. We refer to these scenarios as *flat2hier*, *hier2flat* and *hier2hier*, respectively.

Unlike protocol replacement, changing the mode of the IGP configuration can require a specific router migration ordering. Indeed, the nh function can change in hierarchy modification scenarios because of the intrazone over interzone path preference rule applied by routers in hierarchical IGPs (see Section II). Hence, forwarding loops can arise due to inconsistencies between already migrated routers and routers that are not migrated yet. Consider for example the topology depicted on the left side of Fig. 1. In a *flat2hier* scenario, some routers change their next-hop toward destinations $E1$ and $E2$. In particular, the right side of Fig. 1 shows the next-hop function for all the routers when the destination is $E2$. During the migration process, a forwarding loop arises for traffic destined to $E2$ if $B1$ is migrated before $E1$. Indeed, $B1$ reaches $E2$ via $E1$ in hierarchical mode, and $E1$ reaches $E2$ via $B1$ in flat mode. Hence, for each time t where $B1$ is already migrated and $E1$ is not, the forwarding path used by $B1$ is $\pi(B1, E2, t) = \{(B1\ E1\ B1)\}$ since $\mathrm{nh}_{\mathrm{final}}(B1, E2) = \{E1\}$ and $\mathrm{nh}_{\mathrm{init}}(E1, E2) = \{B1\}$. Notice that such a loop lasts until $E1$ is migrated. A symmetric constraint holds between routers $B2$ and $E2$ for traffic destined to $E1$. A loop-free migration can be achieved by migrating $E1$ and $E2$ before $B1$ and $B2$.

Nevertheless, there are also cases in which it is not possible to avoid loops during the migration. Consider, for example, the topology represented in Fig. 2. In this topology, symmetric constraints between $B1$ and $B2$ for traffic destined to $E2$ and $E3$ imply the impossibility of finding a loop-free ordering. We refer the reader to the central and the right parts of Fig. 2 to visualize the next-hop functions in flat and hierarchical modes.

Similar examples can be found for *hier2flat* and *hier2hier* migrations. They are omitted for brevity. Observe that problems in hierarchy modification scenarios are mitigated in protocols such as IS-IS that natively support multiple adjacencies [1]. In fact, multiple adjacencies belonging to different zones decrease the number of cases in which the nh function changes during the migration. However, migration loops can still arise, depending on the initial and the final configurations.

3) Route Summarization: Introducing or removing route summarization (i.e., *summarization* scenarios) in a network can lead to forwarding loops. For example, consider the topology represented in the left part of Fig. 3. The right part of the figure visualizes the nh functions before and after the introduction of route summarization. In this case, the introduction of route summarization on $B1$ and $B2$ can lead to a forwarding loop between $B3$ and $B4$ for traffic destined to $E2$. Indeed, before summarizing routes, $B3$ and $B4$ prefer to send traffic destined to $E2$ via $B2$. On the other hand, when summarization is introduced, $B1$ and $B2$ propagate one aggregate for both $E1$ and $E2$ with the same weight. Hence, $B3$ and $B4$ change their next-hop since the path to $B1$ has a lower weight than the path to $B2$.

As for hierarchy modifications, no loop-free ordering exists in some cases. An example of such a situation can be built by simply replicating the topology in Fig. 3 so that symmetric constraints on the migration order hold between $B3$ and $B4$.

IV. Methodology

Fig. 4 illustrates the main steps of our methodology. In the first step, we precompute an ordering in which to seamlessly migrate routers with no packet loss (Section V). Migrating all the routers at once is not a viable solution in practice as it can generate protocol-dependent loops and control-plane traffic storms concerning all the protocols (BGP, LDP, PIM, etc.) that rely on the IGP. Moreover, this approach prevents operators from controlling the migration process and from falling back to a previous working state when a problem is detected, e.g., when a router does not receive an intended command. All the discussions that we had with network operators further confirmed that they prefer to gradually migrate their network to have full-control of the process. In the same step, when a per-router ordering does not exist, we identify the set of problematic destinations for which contradictory ordering constraints exists, and we compute a per-destination ordering for each of them. Then, we compute a per-router ordering for the rest of the destinations.

The actual migration process begins in the second step. As basic operation, we exploit a known migration technique called *ships-in-the-night* [3], [4], [7], in which both the initial and the final IGP configurations are running at the same time on each router in separate routing processes. Routing processes are ranked on the basis of their priority, the AD. When a route for a given prefix is available in multiple processes, the one with the lowest AD is installed in the Forwarding Information Base (FIB). In this step, we set the AD of the routing process running the final IGP configuration to 255 since this setting ensures that no route coming from that process is installed in the FIB [22]. All ISP routers typically support this feature.

In the third step of the migration, we wait for network-wide convergence of the final IGP configuration. After this step, both IGPs are in a stable routing state.

In the fourth step, we progressively migrate routers following the ordering precomputed in the first step of the methodology.

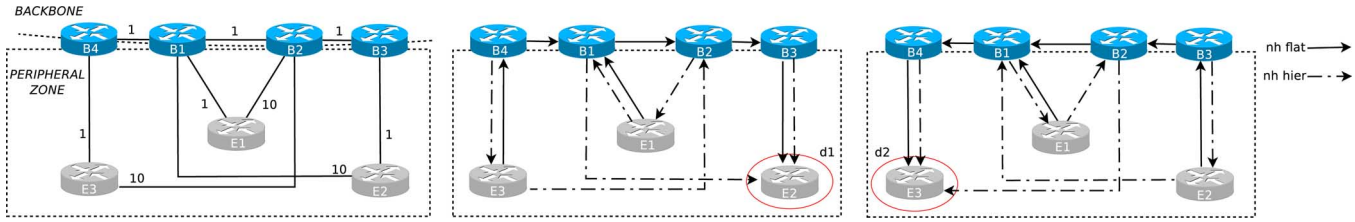


Fig. 2. Loop gadget. No migration ordering is loop-free for *flat2hier* and *hier2flat* scenarios because of contradictory constraints between $B1$ and $B2$.

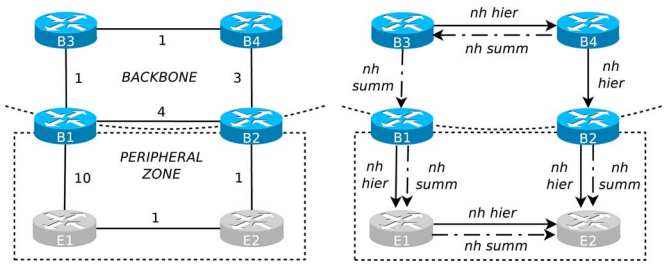


Fig. 3. Route summarization gadget. When summarization is introduced or removed, a specific migration ordering is needed between $B3$ and $B4$ to avoid forwarding loops.

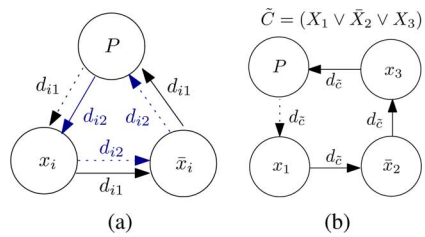


Fig. 5. Gadgets used in the reduction from 3-SAT to RMOP. Solid lines represent $\mathrm{nh}_{\mathrm{init}}$, while dotted lines represent $\mathrm{nh}_{\mathrm{final}}$. Edges are labeled with destinations to which they refer. (a) Variable gadget. (b) Clause gadget.

Seamless IGP Migration Methodology

1) *Compute a lossless router migration order*. In case no per-router ordering exists, compute a per-destination ordering for the troublesome destinations.
2) *Introduce the final IGP configuration*. The final IGP configuration is introduced on all the routers in the network. However, routers continue to forward packets according to the initial IGP configuration.
3) *Monitor the final IGP status*. Wait for the convergence of the final IGP configuration.
4) *Progressively migrate routers*. The pre-computed lossless router migration order is followed. In case no per-router migration ordering exists, a per-destination ordering is also applied for the troublesome destinations.
5) *Remove the initial IGP configuration.* The initial IGP is removed from all the routers in the network.

Fig. 4. Proposed methodology for seamless IGP migrations.

For this purpose, we lower the AD of the routing process running the final IGP such that it is smaller than the AD of the process running the initial configuration. Doing so, the router installs the final routes in its FIB. If a per-destination ordering is required for some destinations, we prevent them from being routed according to the final IGP by keeping the AD of these destinations to a high value. This could be done by using tailored route-maps matching the problematic destinations (see [23] and [24]). After, we migrate the problematic destinations one by one, by lowering their AD following the precomputed per-destination orderings. Since a routing entry change could take about 200 ms before being reflected in the FIB [25], we wait for a given amount time (typically a few seconds) before migrating the next router in the ordering. This step ensures a loop-free migration of the network. Notice that switching the AD and updating the FIB are lossless operations on ISP routers [26].

In the last step, we remove, in any order, the initial IGP configuration from the routers. This is safe since all of the routers are now using the final IGP to forward traffic.

V. Loop-Free Migrations

In this section, we study the problem of migrating a network from one link-state IGP configuration to another without creating any loop. First, we prove that the problem is NP-complete. Then, we present the algorithms we use to compute a loop-free router migration ordering. Finally, we describe how to adapt the algorithms to compute a per-destination ordering to use as fallback when a per-router ordering does not exist.

A. Router Migration Ordering Problem

We now study the following problem from an algorithmic perspective.

Problem 2: Given an initial and a final next-hop function, a logical graph G, and a set of destinations D, compute a router migration ordering, if any, such that no forwarding loop arises in G for any $d \in D$.

Even the problem of deciding if a loop-free router migration ordering exists, which we call Router Migration Ordering Problem (RMOP), is an $\mathcal{NP}$-complete problem. In order to prove the complexity of the RMOP problem, we use a reduction from 3-SAT [27]. In the following, we denote the fact that u is migrated before v with $u < v$. Consider a logical formula F in conjunctive normal form. Let $C_1, \ldots, C_l$ be the clauses in F, $X_1, \ldots, X_h$ be the variables, and X_1 and $\bar{X}_1$ the literals corresponding to X_1. In the following, we build the RMOP instance $S = (G = (V, E), D, \mathrm{nh}_{\mathrm{init}}, \mathrm{nh}_{\mathrm{final}})$ corresponding to F.

As a basis, G contains a single vertex P. For each variable X_i in F, we add to S a variable gadget as depicted in Fig. 5(a). In practice, we add two vertices d_{i1}, d_{i2}, x_i and $\bar{x}_i$ to G, along with edges $(x_i\ P)$, $(\bar{x}_i\ P)$, and $(x_i\ \bar{x}_i)$. d_{i1} and d_{i2} are also added to D. Intuitively, node x_i and $\bar{x}_i$ represent literals x_i and $\bar{x}_i$, respectively. In the following, we call nodes x_i and $\bar{x}_i$ *literal vertices*. Assigning TRUE to X_i corresponds to migrate X_i before P, while assigning FALSE to x_i implies $\bar{x}_i < P$. For each clause $C_j = (L_1 \vee L_2 \vee L_3)$, we add a clause gadget similar to that depicted in Fig. 5(b). For each literal in C_j, we add the corresponding literal vertex, along with edges $(l_1\ l_2)$, $(l_2\ l_3)$,

$(l_3\ P)$, and $(P\ l_1)$. Moreover, a vertex $\tilde{d}_j$ is added to both V and D. After having added all the vertices, one edge is added to E from any vertex to any destination.

Finally, we define the next-hop functions. For each $d_{i1}, \mathrm{nh}_{\mathrm{init}}(u, d_{i1}) = \mathrm{nh}_{\mathrm{final}}(u, d_{i1}) = \{d_{i1}\}\ \forall u \in V$, except $\mathrm{nh}_{\mathrm{init}}(x_i, d_{i1}) = \{\bar{x}_i\}$, $\mathrm{nh}_{\mathrm{init}}(\bar{x}_i, d_{i1}) = \{P\}$, and $\mathrm{nh}_{\mathrm{final}}(x_i, d_{i1}) = \{\bar{x}_i\}$. Similarly, for each d_{i2}, $\mathrm{nh}_{\mathrm{init}}(u, d_{i2}) = \mathrm{nh}_{\mathrm{final}}(u, d_{i2}) = \{d_{i2}\}\ \forall u \in V$, except $\mathrm{nh}_{\mathrm{init}}(P, d_{i2}) = \{\bar{x}_i\}$ $\mathrm{nh}_{\mathrm{final}}(x_i, d_{i2}) = \{\bar{x}_i\}$, and $\mathrm{nh}_{\mathrm{final}}(\bar{x}_i, d_{i2}) = \{P\}$. Finally, for each $\tilde{d}_j$ corresponding to a clause $C_j = (L_{j1} \vee L_{j2} \vee L_{j3})$, $\mathrm{nh}_{\mathrm{init}}(u, \tilde{d}_j) = \mathrm{nh}_{\mathrm{final}}(u, \tilde{d}_j) = \{\tilde{d}_j\}\ \forall u \in V$, except $\mathrm{nh}_{\mathrm{final}}(P, \tilde{d}_j) = \{l_{j1}\}$, $\mathrm{nh}_{\mathrm{init}}(l_{j1}, \tilde{d}_j) = \{l_{j2}\}$, $\mathrm{nh}_{\mathrm{init}}(l_{j2}, \tilde{d}_j) = \{l_{j3}\}$, and $\mathrm{nh}_{\mathrm{init}}(l_{j3}, \tilde{d}_j) = \{P\}$.

Regarding destinations d_{i1} and d_{i2}, it is easy to verify that only x_i,$\bar{x}_i$, and P can be part of a loop since the next-hop of all the other vertices is the destination in both the initial and the final next-hop functions. In particular, a loop arises toward d_{i1} or d_{i2} if and only if P is the first node to be migrated or P is the very last node to be migrated, respectively.

Property 1: A router migration ordering does not create a loop toward destinations d_{i1} and d_{i2} if and only if:

- $x_i < P \rightarrow P < \bar{x}_i$;
- or $\bar{x}_i < P \rightarrow P < x_i$.

As a consequence, only the orders $x_i < P < \bar{x}_i$ and $\bar{x}_i < P < x_i$ are loop-free. This prevents a variable to be TRUE and FALSE at the same time.

Analogously, for destinations d_j, all the routers, except l_{j1}, l_{j2}, l_{j3}, and P, cannot be part of a loop since their next-hop is d_j in both the next-hop functions. The following property holds for l_{j1}, l_{j2}, l_{j3}, and P.

Property 2: A loop arises toward a destination d_j if and only if P is migrated before all the vertices l_i, with $i = 1, 2, 3$, corresponding to a literal in C_j.

It is easy to check that the reduction can be done in polynomial time. We now use such a reduction to prove the complexity of RMOP.

Theorem 1: The Router Migration Ordering Problem is NP-complete.

Proof: Consider a logical formula F in conjunctive normal form. Let S be the instance of the Router Migration Ordering Problem corresponding to F. Then, we have the following.

- If F is satisfiable, then there exists a router migration order in S that does not create any forwarding loop. In fact, if F is satisfiable, then there exists at least one boolean assignment such that for each clause C_j at least one literal l_i is TRUE. This corresponds to $l_i < P$ in the migration order. Such a condition guarantees that no loop arises in the clause gadget corresponding to C_j, by Property 2. The same argument can be iterated on all the clauses in F. Since the boolean assignment that satisfies F is a valid assignment, no variable is assigned both TRUE and FALSE at the same time, hence no loop can be generated in the variable gadget (Property 1).
- If F is not satisfiable, then there does not exist a router migration order in S that does not create any forwarding loop. In fact, if F is not satisfiable, then for each valid boolean assignment, at least one clause $C_n = (L_{j1} \vee L_{j2} \vee L_{j3})$ is not satisfied by the boolean assignment. However, this means that all the literals in C_n are FALSE. This corresponds to migrate P before all the nodes l_{ji}, with $i \in \{1, 2, 3\}$. Hence, a loop arises for destination d_n by Property 2. The same argument can be iterated on all the boolean assignment on the variables in F. As a consequence, every router migration ordering on S contains at least one loop.

The proof is completed by noting that a loop-free router migration order is a succinct certificate for S. ■

1: ***loop_enumeration_run***$(G = (V, E), D, nh_{init}, nh_{final})$
2: $CS \leftarrow \emptyset$
3: **for** $d \in D$ **do**
4: $\bar{G}_d = (V, \bar{E})$, with $\bar{E} = \{(u\ v)\}$ such that $v \in nh_{init}(u, d)$ or $v \in nh_{final}(u, d)$
5: **for** each cycle L in $\bar{G}_d$ **do**
6: $V_{init,L} = \{u \in L : \exists v, (u\ v) \in L, v \in nh_{init}(u, d)$ but $v \notin nh_{final}(u, d)\}$
7: $V_{final,L} = \{u \in L : \exists v, (u\ v) \in L, v \in nh_{final}(u, d)$ but $v \notin nh_{init}(u, d)\}$
8: $CS \leftarrow CS \cup \{u_0 \vee \cdots \vee u_k < v_0 \vee \cdots \vee v_l\}$, where $u_i \in V_{init,L}\ \forall i = 0, \ldots, k$, and $v_j \in V_{final,L}\ \forall j = 0, \ldots, l$.
9: **end for**
10: **end for**
11: $LP \leftarrow$ new LP problem
12: **for** $u_0 \vee \cdots \vee u_k < v_0 \vee \cdots \vee v_l \in CS$ **do**
13: add to LP the following constraints
14: $t_{u_0} - MAX\_INT \times Y_1 < t_{v_0}$
15: $\ldots$
16: $t_{u_0} - MAX\_INT \times Y_l < t_{v_l}$
17: $t_{u_1} - MAX\_INT \times Y_{l+1} < t_{v_0}$
18: $\ldots$
19: $t_{u_k} - MAX\_INT \times Y_{l \times k} < t_{v_l}$
20: $t_{u_0}, \ldots, t_{u_k}, t_{v_0}, \ldots, t_{v_l}$ integer
21: $Y_1, \ldots, Y_{l \times k}$ binary
22: $\sum_{1<i<=l \times k} Y_i < l \times k$
23: **end for**
24: **return** *solve_lp_problem(LP)*

Fig. 6. Loop Enumeration Algorithm.

B. Router Migration Ordering Algorithms

We now present a correct and complete algorithm to find a loop-free ordering. Because of the complexity of the problem, the algorithm is inefficient and can take several hours to run on huge ISP networks (see Section VII). Hence, we also propose an efficient heuristic that is correct but not complete.

In the following, we describe our algorithms in the absence of virtual nodes. Explicit support of virtual nodes is not needed since virtual nodes never change their respective next-hop function. Indeed, consider the logical graph on which the algorithms run. By construction, each virtual node v has only one edge, connecting it to the node u representing the corresponding physical router. Consequently, the next-hop of v is always v when v itself is the destination and it is always u for any other destination.

Loop Enumeration Algorithm: The Loop Enumeration Algorithm (Fig. 6) enumerates all the possible migration loops that can arise during a migration. Then, it outputs the sufficient and necessary constraints that ensure that no loop arises. To identify all possible migration loops, for each destination d, the algorithm builds the graph G_d (line 4) as the union of the actual paths in the initial and in the final configuration. G_d contains all

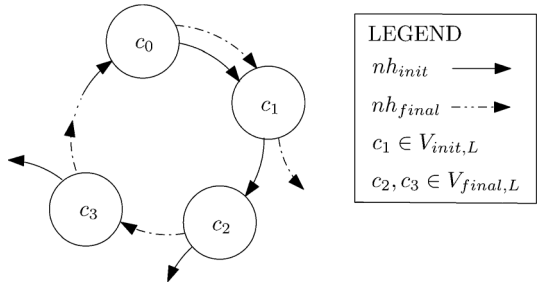


Fig. 7. Abstract representation of a migration loop.

the possible combinations of paths followed by traffic destined to d for any migration order. Then, all the cycles are enumerated, and for each cycle, the algorithm outputs the constraint (line 8) of migrating at least one router that participates in the loop in the initial configuration before at least one router that is part of the loop in the final configuration (lines 5–8). In the example of Fig. 7, indeed, migrating c_1 before at least one among c_2 and c_3 avoids the loop. In the algorithm, $V_{\text{init},L}$ represents the set of routers that participate in the loop when they are in the initial configuration (line 6), and $V_{\text{final},L}$ contains only routers that participate in the loop when they are in the final configuration (line 7). The constraints identified by the algorithm are encoded in an integer linear program (lines 12–22), where the variables t_{u_i} represent the migration steps at which routers can be safely migrated (lines 14–19). Finally, the algorithm tries to solve the linear program and returns a loop-free ordering if one exists (line 24).

We now show correctness and completeness of the Loop Enumeration Algorithm.

Theorem 2: The Loop Enumeration Algorithm is correct and complete.

Proof: We prove the statement by showing that the linear program solved in the Loop Enumeration Algorithm encodes all the sufficient and necessary conditions for any migration loop to not arise. Indeed, let $u_0 \vee \ldots \vee u_k < v_0 \vee \ldots \vee v_l$ be the ordering constraint that the Loop Enumeration Algorithm identifies for a given loop $L = (c_0\ c_1\ \ldots\ c_k\ c_0)$ concerning traffic destined to $d \in D$. We now show that L does not arise at any migration step if and only if the constraint is satisfied.

If the loop does not arise, then the constraint is satisfied. Suppose by contradiction that the constraint is not satisfied. Then, there exists a time $\bar{t}$ such that all the routers in $V_{\text{final},L}$ are migrated while all the routers in $V_{\text{init},L}$ are not migrated. Consider c_0. If $c_0 \in V_{\text{final},L}$, then it is already migrated, i.e., $\text{nh}(c_0, d, \bar{t}) = \text{nh}_{\text{final}}(c_0, d)$, hence $c_1 \in \text{nh}(c_0, d, \bar{t})$, by definition of $V_{\text{final},L}$. If $c_0 \in V_{\text{init},L}$, then $\text{nh}(c_0, d, \bar{t}) = \text{nh}_{\text{init}}(c_0, d)$ and $c_1 \in \text{nh}(c_0, d, \bar{t})$. Finally, if $c_0 \notin V_{\text{init},L}$ and $c_0 \notin V_{\text{final},L}$, then $c_1 \in \text{nh}(c_0, d, t)\ \forall t$. In any case, $c_1 \in \text{nh}(c_0, d, \bar{t})$. Iterating the same argument for all the routers in L, we conclude that $c_{i+1} \in \text{nh}(c_i, d, \bar{t})$, with $i = 0, \ldots, k$ and $c_{k+1} = c_0$. Thus, L arises at time $\bar{t}$.

If the constraint is satisfied, then the loop does not arise. Assume, without loss of generality, that $c_u \in V_{\text{init},L}$ is migrated at time t', while at least one router $c_v \in V_{\text{final},L}$ is migrated at $t'' > t'$. Then, L cannot arise $\forall t < t''$ since $\text{nh}(c_v, d, t) = \text{nh}_{\text{init}}(c_v, d)$ implies that $c_{v+1} \notin \text{nh}(c_v, d, t)$ by definition of $V_{\text{final},L}$. Moreover, L cannot arise $\forall t > t'$ since $\text{nh}(c_u, d, t) = \text{nh}_{\text{final}}(c_u, d)$ implies that $c_{u+1} \notin \text{nh}(c_u, d, t)$ by definition of $V_{\text{init},L}$. Since $t'' > t'$, no time exists such that L arises during the migration. ■

```
1: routing_trees_run(G = (V, E), D, nh_init, nh_final)
2: C ← ∅
3: for d ∈ D do
4:     S_d ← greedy_run(V, d, nh_init, nh_final)
5:     V̄_d ← {v_i : nh_init(v_i, d) ≠ nh_final(v_i, d)}
6:     G_d = (V, E'), with E' = {(u, v) : v ∈ nh_final(u, d)}
7:     for P = (v_0 ... v_k), with v_k = d, (v_i, v_{i+1}) ∈ E', and
       predecessors(v_0) = ⊘ do
8:         last ← Null
9:         for u ∈ P ∩ V̄_d and u ∉ S_d do
10:            if last ≠ Null then
11:                C ← C ∪ {(u, last)}
12:            end if
13:            last ← u
14:        end for
15:    end for
16: end for
17: G_c ← (V, C)
18: return topological_sort(G_c)
19:
20: greedy_run(V, d, nh_init, nh_final)
21: S_d ← ∅
22: N ← {d}
23: while N ≠ ∅ do
24:    S_d = S_d ∪ N
25:    N = ∅
26:    for u ∈ V, u ∉ S_d do
27:        if nh_init(u, d) ∪ nh_final(u, d) ⊆ S_d then
28:            N = N ∪ {u}
29:        end if
30:    end for
31: end while
32: return S_d
```

Fig. 8. Routing Trees Heuristic.

It is easy to verify that the algorithm requires exponential time. Indeed, the algorithm is based on the enumeration of all the cycles in a graph, and the number of cycles in a graph can be exponential with respect to the number of nodes.

Routing Trees Heuristic: The Routing Tree Heuristic is illustrated in Fig. 8. Intuitively, it computes ordering constraints separately for each destination, so that next-hop changing routers are not migrated before their final forwarding path to each destination is established (similarly to what is proposed in [18] and [20]). A router ordering that satisfies all per-destination constraints is then computed. As the first step, for each destination $d \in D$, the heuristic exploits a greedy procedure to compute a set S_d of nodes that are guaranteed not to be part of any loop (line 4). The greedy procedure (lines 20–32) incrementally (and greedily) grows the set S_d, adding a node to S_d at each iteration if and only if all the next-hops of the node in the initial and in the final configurations are already in S_d (lines 27–28). After this step, the Routing Trees Heuristic builds directed graph G_d, which is guaranteed to be acyclic since the final configuration is loop-free. G_d contains only the actual paths followed by packets to reach d in the final configuration (line 6). Then, it generates a constraint for each pair of routers (u, v) such that $(u \ldots v \ldots d) \in \pi_{\text{final}}(u, d)$, and both u and v do not belong to S_d and change at least one next-hop between the initial and the final configuration (lines 7–15). In particular, among the routers that change one or more next-hops

during the migration (set $\bar{V}_d$ at line 5), each router is forced to migrate after all its successors in the actual path toward d (line 11). In the final step, the heuristic tries to compute an ordering compliant with the union of the constraints generated for all the destinations (lines 17–18).

It is easy to check that the algorithm is polynomial with respect to the size of the input. We now prove that the algorithm is correct. First, we show that the routers in S_d can be migrated in any order without creating loops toward d, hence it is possible not to consider them in the generation of the ordering constraints. Then, we prove that the constraints are sufficient to guarantee that the ordering is loop-free.

Lemma 1: If the greedy procedure adds a router u to S_d, then u cannot be part of any migration loop toward destination $d \in D$.

Proof: Suppose, by contradiction, that there exists a router u added to S_d by the greedy procedure at a given iteration i, such that $(u\ v_0\ \ldots\ v_k\ u) \in \pi(u, d, t)$, with $k \geq 0$, at a given time t and for a given migration ordering. By definition of the algorithm, one router is added to S_d if and only if all its next-hops $w_0, \ldots, w_n$ (in both the initial and final IGP configurations) are already in S_d since each node in $\{w_0, \ldots, w_n\}$ is added to S_d at a given iteration before i. Hence, $v_k \notin S_d$ at iteration i because u is one of the next-hops of v_k and it is added to S_d at iteration i by hypothesis. Iterating the same argument, all routers $v_h \notin S_d$ at iteration i, $\forall h = 0, \ldots, k$. As a consequence, GREEDY does not add u to S_d at iteration i, which is a contradiction. ■

Theorem 3: Let $S = x_1, \ldots, x_n$ be the sequence computed by the Routing Tree Heuristic. If the routers are migrated according to S, then no migration loop arises.

Proof: Suppose by contradiction that migration is performed according to S, but migrating a router u creates a loop for at least one destination d. In that case, there exists a set of routers $\tilde{V} = \{v_1, \ldots, v_k\}$, such that $C = (u\ v_0\ \ldots\ v_k\ u) \in \pi(u, d, t)$, at a certain time t. By Lemma 1, all $v_i \notin S_d$. By definition of the heuristic, all routers v_i are such that $\mathrm{nh}(v_i, d, t) = \mathrm{nh}_{\mathrm{final}}(v_i, d)$, with $i = 0, \ldots, k$, because either they do not change their next-hop between the initial and the final configuration or they precede u in S. Hence, at time t, both u and all the routers $v_i \in \tilde{V}$ are in the final configuration. This is a contradiction since we assumed that the final IGP configuration is loop-free. ■

Note that the heuristic is not complete; while the constraints it generates are sufficient to guarantee no forwarding loops, they are not necessary. Indeed, for each destination d, it imposes specific orderings between all the routers (not belonging to S_d) that change one of their next-hops toward d, even if it is not needed. For instance, in the scenario of Fig. 9, the heuristic mandates v to be migrated before u and u before z. However, no loop arises also if v is migrated before z and z before u. Generating unnecessary constraints prevents the heuristic from identifying a loop-free migration ordering every time it exists. Nonetheless, in carefully designed networks [28], such cases are rare. In Section VII, we show that the heuristic found an ordering in most of our experiments on realistic topologies.

C. Per-Destination Ordering

If a per-router ordering does not exist or the Routing Tree Heuristic does not find a solution, a per-destination ordering can be computed. The per-destination ordering is applied to problematic destinations only, that is, to the destinations for which contradictory ordering constraints exist. Per-destination orderings can be computed directly from the ordering constraints identified by our per-router ordering algorithms. Note that it may not be necessary to compute an ordering for every problematic destination since excluding a destination from the per-router ordering may unlock the problem for a set of other problematic destinations. The number of destinations for which a per-destination ordering is required can thus be minimized. However, our experimental evaluation (see Section VII) suggests that potentially troublesome destinations are few in practice, hence the need for minimizing problematic destinations is limited.

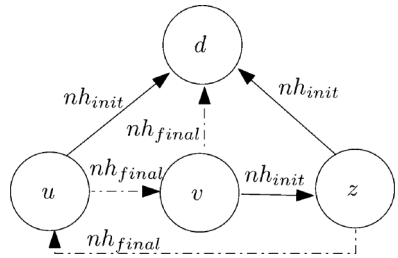


Fig. 9. In some migration scenarios, the Routing Trees Heuristic generates unnecessary constraints.

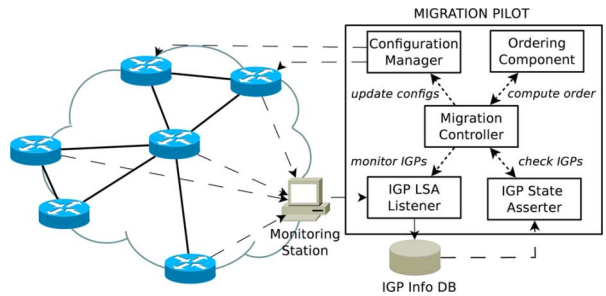


Fig. 10. Provisioning system automates all the reconfiguration process. It computes the ordering, monitors the network, and pushes the intermediate configurations in the appropriate order.

VI. PROVISIONING SYSTEM

We realized a system which computes and automates all the required steps for a seamless migration. The architectural components of our system are depicted in Fig. 10. We now describe how data flows through the system (dashed lines in the figure), while stressing the role of each component.

The main purpose of the monitoring component is to assess properties of intermediate configurations, that is, checking that given routers are correctly migrated and the expected routing state is reached. The monitoring mechanisms also enable failure detection. However, while we discuss how to prevent packet loss due to network failures in Section VIII, we plan to study effective reactive strategies to unexpected failures in future work. The monitoring component encompasses an IGP Link-State Advertisement (LSA) Listener and an IGP State Asserter. The *IGP LSA Listener* collects and parses the IGP LSAs exchanged by routers, storing IGP adjacencies, link weights, and announced IP prefixes in a database. We chose to implement the IGP LSA Listener by using packet-cloning features available on routers, as it is shown to be an effective method to collect all control-plane messages with low resource consumption on monitored

routers [29]. The *IGP State Asserter* queries the database and assesses properties of the monitored IGPs state. The current implementation of the IGP State Asserter can check an IGP for convergence completion. IGP convergence is deduced by stability, over a given (customly set) time, of the expected IGP adjacencies and of the announced prefixes. Moreover, the IGP State Asserter is able to verify the announcement of a given set of prefixes in an IGP, and the equivalence of two IGPs, i.e., the equivalence of the logical graph, and of the forwarding paths toward a given set of destinations.

The IGP State Asserter is triggered at specific moments in time by the *Migration Controller*, which is the central component of the system, responsible for tasks' coordination. Before the actual migration process starts, it delegates the computation of a loop-free router migration ordering to the *Ordering Component*. This component implements the ordering algorithms described in Section V-B. Then, the Migration Controller runs the IGP LSA Listener. When needed (see Section IV), the Migration Controller asks the IGP State Asserter to assess whether it is possible to safely modify the configuration of the devices in the network without incurring transient states. This boils down to checking the stability of the current IGP. At each step of the migration process the controller also requires the *Configuration Manager* to properly update the configuration on the routers as described in Section IV. Based on a network-wide model, the Configuration Manager generates the necessary commands to be sent to routers for each migration step. The Configuration Manager is based on an extended version of NCGuard [30].

VII. Evaluation

In this section, we evaluate the ordering algorithms and the provisioning system. The system is evaluated on the basis of a case study in which a network is migrated from a flat to a hierarchical IGP.

A. Data Set and Methodology

Our data set contains both publicly available and confidential data relative to commercial ISP topologies. Concerning publicly available topologies, we used the inferred topologies provided by the Rocketfuel project [31]. Rocketfuel topologies represent ISPs of different sizes, the smallest one having 79 nodes and 294 edges, while the biggest one contains 315 nodes and 1944 edges. In addition, some network operators provided us with real-world IGP topologies. In this section, we discuss the result of our analyses on all the Rocketfuel data and on the anonymized topologies of three ISPs, namely $tier1.A$, $tier1.B$, and $tier2$. $tier1.A$ is the largest Tier1, and its IGP logical graph has more than 1000 nodes and more than 4000 edges. $tier1.A$ currently uses a flat IGP configuration. The other two ISPs are one order of magnitude smaller, but use a hierarchical IGP.

On this data set, we performed several experiments. We considered the introduction of summarization, as well as *flat2hier* and *hier2hier* scenarios. Since most of the topologies in our data set are flat, we artificially built a hierarchy (i.e., the separation in zones) in order to consider scenarios in which hierarchical configurations are needed. In particular, we grouped routers according to geographical information present in the name of the routers. Doing so, we built two hierarchical topologies out of each flat topology. In the first one, zones are defined per city. In the second one, zones are defined per-continent. In both topologies, we built the backbone by considering routers connected to more than one zone as ZBRs and routers connected only to ZBRs as pure backbone routers. To simulate a *hier2hier* scenario, we artificially enlarged the backbone by moving to it a fixed number (from 1 up to 32) of links. Such links were randomly chosen among the links between a ZBR and a router that does not participate in the backbone. For the summarization scenario, we aggregated all the destinations inside the same zone into a single prefix. This was done for all the zones but the backbone. Our hierarchy construction methodology and the way prefixes are summarized follow the guidelines proposed in [32]. All the tests were run on a Sun Fire X2250 (quad-core 3-GHz CPUs with 32 GB of RAM). We omit the results of some experiments due to space limitations.

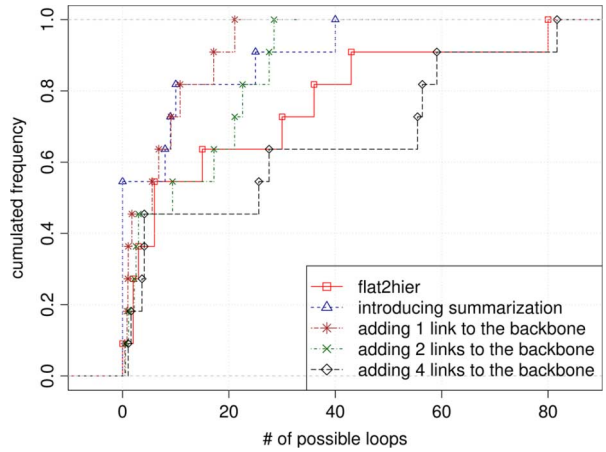


Fig. 11. CDF of the number of loops that can arise on Rocketfuel topologies. In the worst case, up to 80 different forwarding loops can be created during the reconfiguration.

B. Ordering Algorithms

We first evaluate the usefulness and efficiency of the Loop Enumeration Algorithm and Routing Tree Heuristic. Fig. 11 shows the cumulative distribution function (cdf) of the number of loops that can arise in Rocketfuel topologies. Different migration scenarios are considered. Each point in the plot corresponds to a specific topology and a specific scenario. In *flat2hier*, up to 80 different loops can arise in the worst case, and at least 30 loops can arise for four topologies out of 11. Other scenarios follow similar trends. Observe that in the *hier2hier* scenario (curves "adding x links to the backbone"), the number of possible loops significantly increases with the number of links which change zone. In all the scenarios, almost all the loops involve two routers, with a few exceptions of three routers loops. Also, the vast majority of loops concern traffic destined to routers that do not participate in the backbone. These routers are at the border of the network (e.g., BGP border routers or MPLS PEs) and normally attract most of the traffic in ISP networks. Hence, computing an ordering in which they are not involved in loops can be critical. The number of migration loops is topology-dependent, hence it can be influenced by our design approach. However, these results clearly show that migrating routers in a random order is not a viable option in arbitrary networks. Additionally, for practical reasons, it is desirable that migrations of worldwide networks be carried out on a per-zone

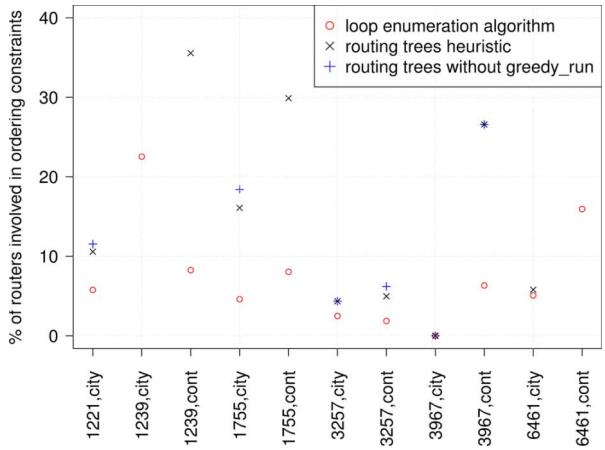


Fig. 12. Percentage of routers involved in the ordering in *flat2hier* (Rocketfuel topologies). Results for other scenarios are similar.

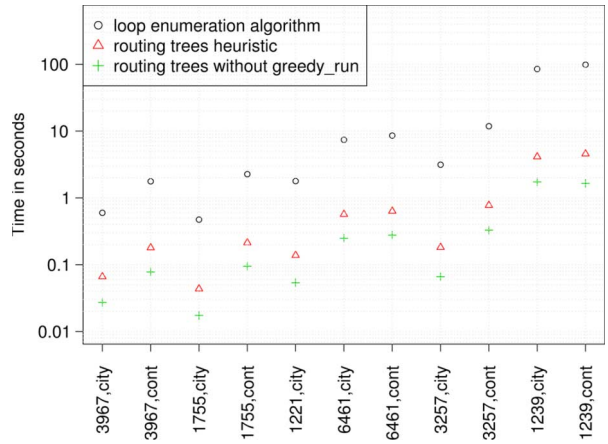


Fig. 13. Time taken to compute an ordering in *flat2hier* (Rocketfuel topologies). Results for other scenarios are similar.

basis, i.e., migrating all the routers in the same zone (e.g., a continent) before routers in other zones. We argue that it is often possible to compute and apply per-zone orderings. Indeed, in both the Rocketfuel and the real-world topologies we analyzed, all the possible loops involve routers in the same zone or backbone routers and routers in a peripheral zone. These considerations further motivate our effort to find a router migration ordering that is guaranteed to be loop-free. We found slightly different results on the real ISP topologies we analyzed. For the two hierarchical ISPs, none or few migration loops can arise in the considered scenarios. This is mainly due to a sensible design of the hierarchy by the ISPs. On the other hand, we found that huge number of problems could arise within $tier1.A$ in the $hier2flat$ scenario where the hierarchy was designed as described in Section VII-A. Indeed, more than 2000 loops might arise, involving up to 10 routers. Again, this stresses the importance of the IGP design on the migration outcome. We discuss simple design guidelines that ease IGP migrations in Section IX.

As a second group of experiments, we ran the ordering algorithms on both the real-world and the Rocketfuel topologies. In the following, we present results for the $flat2hier$ scenario, but similar results and considerations hold for the other scenarios. Fig. 12 shows for each Rocketfuel topology the percentage of routers that need to be migrated in a specific order according to each algorithm (implying that other routers can be migrated in any order). When a point is missing, it means that the corresponding algorithm was not able to find a loop-free ordering for the topology. The enumeration algorithm was always able to find a loop-free ordering in all situations (including the real-world topologies). In the worst case, the computed ordering involves more than 20% of the routers in the network. We believe that finding ordering constraints for such a number of routers is not practical at a glance. This stresses the importance of our algorithms. The Routing Trees Heuristic, instead, found a loop-free ordering on 9 topologies out of 11. In the remaining two cases, the heuristic was not able to find a solution because of contradictory (unnecessary) constraints relative to four and six destinations, respectively. Because of the limited number of destinations involved in contradictory constraints, we propose to apply a per-destination ordering in these cases. Fig. 12 also highlights the gain of relying on the greedy subprocedure, as the heuristic could find a solution for only six topologies without it.

Finally, we evaluated the time taken by our ordering algorithms. Typically, time efficiency of ordering algorithms is not critical in our approach since a loop-free router migration ordering can be computed before actually performing the migration. However, it becomes an important factor to support advanced abilities like computing router migration orderings that ensures loop-free migrations even in case of network failures (see Section VIII). Fig. 13 plots the median of the computation time taken by each algorithm over 50 separated runs. Standard deviation is always under 40 for the loop enumeration algorithm, except for the two cases corresponding to topology 1239, in which standard deviation is around 450. Moreover, the standard deviation of the time taken by the Routing Trees Heuristic is always less than 25. Even if correct and complete, the Loop Enumeration Algorithm is inefficient, especially for large topologies. The heuristic is always one order of magnitude faster. In Fig. 13, the low absolute value of the time taken by the Loop Enumeration Algorithm can be explained by the relatively small size of the Rocketfuel topologies. Nevertheless, for the $tier1.A$ topology, the Loop Enumeration Algorithm took more than 11 h to complete. To further evaluate the performance degradation of the complete algorithm, we enlarged $tier1.B$'s and $tier2$'s topologies. The operation consisted in replicating multiple times the structure of one peripheral zone, and attaching these additional zones to the network in order to reach a size similar to $tier1.A$. In such experiments, we found that the Loop Enumeration Algorithm took several hours even if routers can be migrated in any order, while the heuristics always took less than 1.5 min.

C. Provisioning System

We evaluated the performance of the main components of our provisioning system by means of a case study. In the case study, we performed a $flat2hier$ migration of Geant, the pan-European research network, that we emulated by using a major router vendor routing operative system image. In particular, we simulated the migration from a flat IS-IS configuration to a hierarchical OSPF. Geant's topology is publicly available [33]. It is composed of 36 routers and 53 links. For the case study, we artificially built zones on the basis of the geographical location of the routers and their interconnections [34]. In addition to the backbone (12 routers), we defined three peripheral zones: the

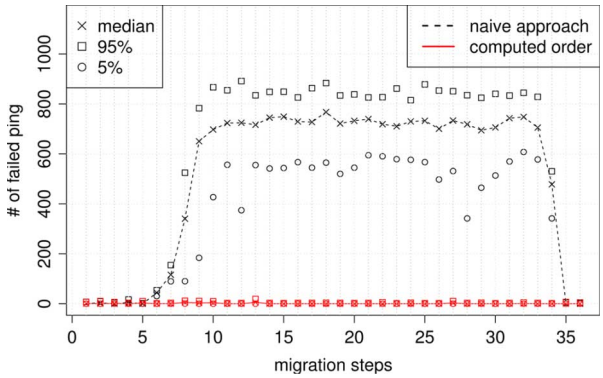


Fig. 14. Our system guarantees that no packet is lost during migration, while long-lasting connectivity disruptions can happen with a naive approach.

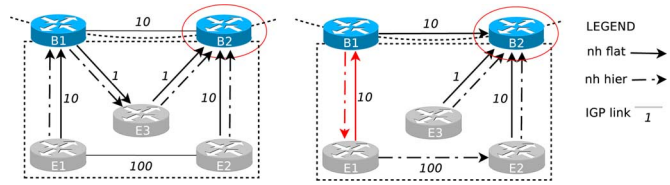


Fig. 15. Link failures can change the reconfiguration ordering to be followed. In a $flat2hier$ scenario on this topology, a forwarding loop can appear between $E1$ and $B1$ if $B1$ is migrated before $E1$ and link ($B1$ $E3$) fails.

southwest area (6 routers), the northeast area (11 routers), and the southeast area (17 routers). We defined the IGP link weights to be inversely proportional to the bandwidth of the links. By executing the Loop Enumeration Algorithm (see Section V-B), we found that eight different loops toward five different destinations could arise on that topology.

To evaluate the cost of not following a proper migration ordering, we counted the number of loops appearing in 1000 random orderings. We observed that more than 50% of the orderings show at least one migration loop for more than 67% of the migration. To further illustrate the effect of not following the ordering, we ran two experiments. In the first experiment, we relied on the ordering computed by the Loop Enumeration Algorithm, while in the second experiment, we adopted an alphabetical order based on the name of the routers. The second experiment mimics a naive approach in which ordering constraints are not taken into account. To minimize the impact of factors beyond our control (e.g., related to the virtual environment), we repeated each experiment 50 times. To measure traffic disruptions due to the migration, we injected data probes (i.e., ICMP echo request) from each router toward the five troublesome destinations. Fig. 14 reports the median, the 5th, and the 95th percentiles of ICMP packets lost that arose after each migration step.

The case study showed the ability of our provisioning system to perform seamless IGP migrations. Following the ordering computed by the Loop Enumeration Algorithm, we were able to achieve no packet loss during the migration (the few losses reported in Fig. 14 should be ascribed to the virtual environment). On the other hand, adopting the naive approach of migrating routers in the random order, forwarding loops arose at step 6 and are only solved at step 34. Thus, the network suffered traffic losses during more than 80% of the migration process.

Our system also enables faster migrations than known migration [4], [7]. The IGP LSA Listener is able to process IGP messages in a few milliseconds. The performance of the module is confirmed by a separate experiment we ran. We forced the Listener to process messages from a pcap file containing 204 LSAs (both OSPF and IS-IS). On 50 runs, the monitor was able to decode and collect each IGP message in about 14 ms on average and 24 ms at most. We evaluated the performance of the IGP State Asserter on the IS-IS and the OSPF DBs generated during the case study. The DBs contained information about 106 directed links and 96 IP prefixes. The IGP State Asserter took about 40 ms to assess equivalence of the logical graph, routing stability, and advertisement of the same set of prefixes in both IGPs. Even if the code could be optimized, current performance is good, also considering that the IGP Asserter does not need to be invoked more than once in absence of network failures (see Section IV). On average, the Configuration Manager took 8.84 s to push one intermediate configuration on a router. The average size of an intermediate configuration is around 38 lines. The entire migration process took less than 20 min. On the contrary, a similar real-world Geant migration took several days to be completed [7].

All the intermediate configurations that our system generated in the case study described above are available online [34].

VIII. Dealing With Network Failures

In this section, we show how to extend the algorithms described in Section III to deal with network failures.

IGP link and node failures modify the IGP topology, which in turn could affect both the nh function and the migration ordering to be followed. Consequently, it may be necessary to adapt the migration ordering to be followed when a failure has been detected in order to not incur long-lasting migration loops. Consider, for example, the topology in Fig. 15 and assume a *flat2hier* migration. The figure shows the initial and the final nh functions toward $B2$, before (left side) and after (right side) the failure of the link between $B1$ and $E3$. Before the failure, any reconfiguration ordering is loop-free since $\mathrm{nh}_{\mathrm{init}} = \mathrm{nh}_{\mathrm{final}}$. However, after the failure of the link between $B1$ and $E3$, $\mathrm{nh}_{\mathrm{init}}$ is no longer equal to $\mathrm{nh}_{\mathrm{final}}$, and a migration loop can be created if $B1$ is migrated before $E1$. To prevent forwarding loops exclusively due to link failures, additional constraints need to be considered during the computation of the migration ordering. For instance, in the example of Fig. 15, $E1$ should be migrated before $B1$ to guarantee loop prevention even in case of failure of link $B1 - E3$.

As a paradigmatic example of how to deal with network failures, we focus on single-link failures. Other kinds of failures (e.g., node and shared risk link group failures) can be similarly addressed. Also, note that single-link failures have been shown to account for the majority of the failures typically occurring in a network [35]. In the following, we refer to a router migration ordering that prevents loop for any single-link failure in the network as a *single-failure compliant ordering*.

For each IGP topology, we computed the additional set of constraints for a single-link failure compliant ordering by iteratively removing single links from the initial topology and running the constraint generation portion of the Loop Enumeration Algorithm or the Routing Trees Heuristic on the topology we obtained. Fig. 16 shows the 50-, 99-, and 100-percentiles of the number of additional forwarding loops that one single-link failure can trigger. Points that do not appear on the figure

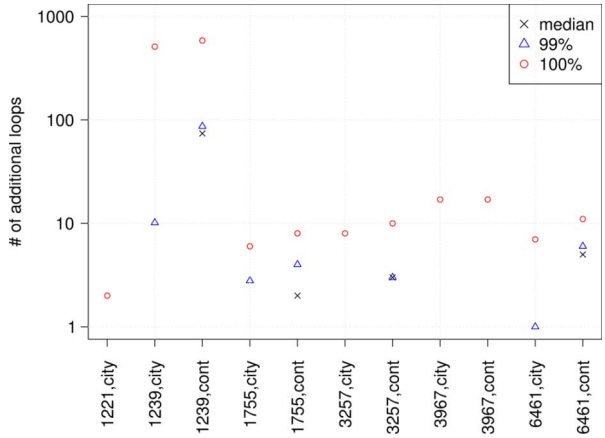


Fig. 16. Average number of additional forwarding loops created by a single-link failure in *flat2hier* reconfiguration scenarios on Rocketfuel topologies. Missing points are equal to 0.

are meant to lay on the x-axis (i.e., no additional loop due to single-link failures). Typically, very few single-link failures are responsible for the vast majority of the additional forwarding loops and ordering constraints. Observe that, in some cases (e.g., $AS1239$), a single-link failure is responsible for more than 500 additional loops. For every network of Fig. 16, we also tried to find single-failure compliant ordering by running the resolver part of either one of the two algorithms on the union of all the constraints. In 9 out of the 11 studied networks, we were able to find such a reconfiguration ordering. Also, in one of the two remaining topologies (namely, $AS1239$, *city*), we computed a migration ordering that prevents loops for 97% of the possible single-link failures. Results on the real ISP topologies are similar. For $tier1.2$, we have been able to find a single-failure compliant ordering, while on $tier2.1$, we were able to find an ordering preventing loops for any single-link failure but one. Our results suggest that finding a single-failure compliant ordering is typically possible on small and medium-sized topologies. For huge networks, finding an ordering is harder as the probability of generating contradictory constraints is higher given the large number of links. In this case, a per-destination ordering (see Section IV) can be precomputed for a subset of the destinations thanks to the efficiency of the heuristic approach.

Regarding the computation time, it took less than 2 h to compute the additional set of constraints on all the topologies but two (namely, $AS1239$, {*city, continent*}). For $AS1239$, {*city, continent*}, it took 46 h. As mentioned earlier, time efficiency is not critical since a single-link failure compliant ordering can be computed offline, before performing the migration. Also, the process is highly parallelizable as every link can be treated separately. Once the additional set of constraints was built, it took less than 1 s in all the topologies to find out an ordering (if any). Given the size of the problem, we did not run our algorithm on $tier1.1$.[1]

IX. Design Guidelines

In this section, we state simple design guidelines that make the entire IGP migration process easier since all the router migration orderings are loop-free. In the following, we consider the design of zones in hierarchical IGPs since the most problematic migrations involve hierarchies.

Guideline A: For each zone Z, the shortest path from each ZBR to any destination in Z is an intrazone path.

Guideline A enables easier $flat2hier$ and $hier2flat$ migrations. In fact, the guideline enforces sufficient conditions to guarantee that the nh function does not change for any router and any destination in any zone Z since an intrazone path is preferred in both flat and hierarchical modes. Since no router in Z changes its path, then $\mathrm{nh}_{\mathrm{init}}(v,d) = \mathrm{nh}_{\mathrm{final}}(v,d)$ also for all routers $v \notin Z$ and $d \in Z$. This implies that no loop can arise during the migration. Notice that Guideline A refers only to ZBRs since if they use intrazone paths, then non-ZBR routers cannot use interzone paths. Establishing multiple adjacencies (e.g., $L1L2$ adjacencies in IS-IS or using OSPF multiarea adjacency extensions [36]) between ZBRs also guarantees the nh function does not change, but backbone links could be unnecessarily traversed in this case.

Guideline B: In each zone Z, the weight of the path from any ZBR to any destination in Z is the same.

Practically, Guideline B can be enforced by organizing routers in each peripheral zone Z in three layers: 1) a core layer, containing ZBRs in Z; 2) an aggregation layer, connecting the access and the core layers; and 3) an access layer, containing destinations in Z. Each ZBR must connect to at least one router in the aggregation layer, and each router in the aggregation layer must connect to all destinations in Z. In addition, all core-to-aggregation links must have the same weight w_1; similarly, all aggregation-to-access layer link weight must be set to the same value w_2 (possibly $w_2 \neq w_1$). Guideline B guarantees easy IGP migrations when route summarization is introduced or removed. We assume that aggregated prefixes are announced with a cost equal to the highest weight among the destinations in the aggregate (as in OSPF, by default [2]). In this case, both with and without summarization, each backbone router chooses the closest ZBR in Z as entry point for destinations in the aggregated prefix. It is easy to check that, as a consequence, the nh function does not change with or without summarization, hence no specific migration ordering is needed during the migration.

X. Related Work

Seamless IGP operation and maintenance have been the focus of several previous studies. For example, several protocol extensions have been proposed [37]–[39] to gracefully restart a routing process. However, few research efforts have been specifically devoted to network-wide seamless IGP migrations, and current best practices [40], [3] are just rules of thumb that do not apply in the general case and do not guarantee lossless reconfiguration processes.

In [17] and [41], Raza *et al.* propose a theoretical framework to formalize the problem of minimizing a certain disruption function (e.g., link congestion) when the link weights have to be changed. The authors also propose a heuristic to find an ordering in which to modify several IGP weights within a network, so that the number of disruptions is minimal. Although their work is close in spirit to ours, the migration scenarios we analyzed cannot always be mapped to a reweighting problem. For example, in hierarchical IGP configurations, both the weight of a link and the zone to which it belongs are considered in

[1]Note that the Routing Tree Heuristic is not usable here as it was not able to find an ordering in the simple case, i.e., without single-link failures.

the computation of the next-hop from a router to a destination, and a unique link weight assignment that generates the same next-hop for each router-to-destination pair may not exist. A more abstract reconfiguration framework on how to transform a feasible solution of a problem into another solution of the same problem is also studied from a purely theoretical point of view (e.g., [42]).

In [43], Keralapura *et al.* study the problem of finding the optimal way in which devices and links can be added to a network to minimize disruptions. Beyond addressing topological changes, our techniques can be used to address several other migration scenarios.

In [44], Chen *et al.* describe a tool that is able to automate status acquisition and configuration change on network devices according to rules specified by domain experts. The tool can be used to automate the ships-in-the-night approach, but not to compute a loop-free ordering. The authors also provide a rule of thumb to avoid problems during IGP migrations, i.e., update edge routers before the backbone ones. However, this rule does not hold in general. For example, migrating $E1$ before $B1$ in Fig. 1 creates a forwarding loop in a $hier2flat$ scenario.

In [45], Alimi *et al.* extend the ship-in-the-night approach by allowing multiple configurations to run simultaneously on a router. They also describe a commitment protocol to support the switch between configurations without creating forwarding loops. While the technique is promising, it cannot be exploited on current routers.

Recently, some techniques [46], [47] have been proposed to enable virtual routers or parts of the configuration of routers (e.g., BGP session) to be moved from one physical device to another. Their works differ from ours as we aim at seamlessly changing network-wide configurations.

In [48], Reitblat *et al.* study the problem of consistent network updates in "Software Defined Networking." They propose a set of consistency properties and show how these properties can be preserved when changes are performed in the network. Unlike our approach, this work only applies to logically centralized networks (e.g., OpenFlow).

Regarding the problem of avoiding forwarding loops in IGPs during transient states, some previous work has also been done. Francois *et al.* propose protocol extensions that allow routers to update their FIB without creating a transient loop after a link addition or removal [19]. Fu *et al.* [20] and Shi *et al.* [21] generalize the results by defining a loop-free FIB update ordering for any change in the forwarding plane and considering traffic congestion, respectively. However, these approaches cannot be used effectively in practice to perform IGP migrations since they only consider updating the FIB on a per-destination basis. Our approach is different as it is aimed at minimizing the number of changes applied to the routers' configurations by searching for a per-router migration ordering. We only apply a per-destination ordering when no per-router migration exists.

IGP migrations could also be performed by using route redistribution. Although new primitives have been recently proposed [49], we believe that relying on a ships-in-the-night approach (when possible) makes the entire migration process easier and more manageable.

XI. Conclusion

Network-wide link-state IGP migrations are a source of concerns for network operators. Unless carried with care, IGP migrations can cause long-lasting forwarding loops, hence significant packet losses. In this paper, we proposed a migration strategy that enables operators to perform network-wide changes on an IGP configuration seamlessly, rapidly, and without compromising routing stability. Our strategy relies on effective techniques for the computation of a router migration ordering and on a provisioning system to automate most of the process. These techniques encompass a complete, time-consuming algorithm and a heuristic. The evaluation we performed on several ISP topologies confirms the practical effectiveness of both the heuristic and the provisioning system. We also evaluated extensions of our techniques that prevent long-lasting loops even in case of network failures. Intuitively, link congestion due to the applied migration ordering can be avoided with similar extensions, i.e., adding constraints to the migration ordering research space. We plan to fully investigate provably congestion-free migrations in future work. In the future, we also plan to study reconfigurations involving different families of routing protocols, like distance-vector IGPs and path-vector policy-based routing protocols (e.g., BGP). Our vision is that network-wide migrations could become a basic operation enabling the seamless replacement or reconfiguration of any network protocol.

Acknowledgment

The authors thank L. Cittadini, R. Bush, G. Xie, and B. Quoitin for their help in improving the paper.

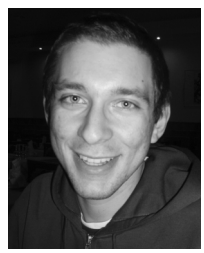
References

[1] D. Oran, "OSI IS-IS intra-domain routing protocol," RFC 1142, 1990.
[2] J. Moy, "OSPF version 2," RFC 2328, 1998.
[3] G. Herrero and J. V. D. Ven, *Network Mergers and Migrations: Junos Design and Implementation*. Hoboken, NJ: Wiley, 2010.
[4] V. Gill and M. Jon, "AOL backbone OSPF-ISIS migration," NANOG29 Presentation, 2003.
[5] North American Network Operators Group, "IPv6: IS-IS or OSPFv3, NANOG thread," 2008 [Online]. Available: http://mailman.nanog.org/pipermail/nanog/2008-December/006194.html
[6] North American Network Operators Group, "OSPF -vs- ISIS, NANOG thread," 2005 [Online]. Available: http://www.merit.edu/mail.archives/nanog/2005-06/msg00406.html
[7] GEANT IPv6 Task Force, "Results of the GEANT OSPF to ISIS migration," presented at the GEANT IPv6 Task Force Meeting 2003.
[8] B. Decraene, J. L. Roux, and I. Minei, "LDP extension for inter-area label switched paths (LSPs)," RFC 5283, 2008.
[9] T. M. Thomas, *OSPF Network Design Solutions*, 2nd ed. San Jose, CA: Cisco Press, 2003.
[10] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for inter-area MPLS traffic engineering," RFC 4105, 2005.
[11] N. Leymann, B. Decraene, C. Filsfils, M. Konstantynowicz, and D. Steinberg, "Seamless MPLS architecture," Internet draft, 2011.
[12] P. Templin, "Small network operator—Lessons learned," NANOG45 Presentation, 2009.
[13] L. Vanbever, S. Vissicchio, C. Pelsser, P. Francois, and O. Bonaventure, "Seamless network-wide IGP migrations," in *Proc. ACM SIGCOMM*, 2011, pp. 314–325.
[14] S. Iasi, P. François, and S. Uhlig, "Forwarding deflection in multi-area OSPF," in *Proc. ACM CoNEXT*, 2005, pp. 254–255.
[15] G. Iannaccone, C. N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier-1 backbone," *IEEE Netw.*, vol. 18, no. 2, pp. 13–19, Mar.–Apr. 2004.
[16] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. ACM SIGCOMM IMW*, 2002, pp. 197–202.
[17] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful network operations," in *Proc. IEEE INFOCOM*, 2009, pp. 289–297.
[18] P. Francois and O. Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1280–1932, Dec. 2007.

[19] P. Francois, M. Shand, and O. Bonaventure, "Disruption-free topology reconfiguration in OSPF networks," in *Proc. IEEE INFOCOM*, 2007, pp. 89–97.
[20] J. Fu, P. Sjodin, and G. Karlsson, "Loop-free updates of forwarding tables," *IEEE Trans. Netw. Service Manage.*, vol. 5, no. 1, pp. 22–35, Mar. 2008.
[21] L. Shi, J. Fu, and X. Fu, "Loop-free forwarding table updates with minimal link overflow," in *Proc. IEEE ICC*, 2009, pp. 1–6.
[22] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making routers last longer with ViAggre," in *Proc. NSDI*, 2009, pp. 453–466.
[23] Cisco, "IP routing protocol-independent commands," Cisco IOS IP Command Reference, 2006, vol. 2 of 3, Routing Protocols.
[24] Juniper Networks, Inc., "Configuring OSPF routing policy," 2010.
[25] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving subsecond IGP convergence in large IP networks," *Comput. Commun. Rev.*, vol. 35, no. 3, pp. 33–44, 2005.
[26] C. Filsfils, P. Mohapatra, J. Bettink, P. Dharwadkar, P. D. Vriendt, Y. Tsier, V. V. D. Schrieck, O. Bonaventure, and P. Francois, "BGP Prefix Independent Convergence (PIC)," Cisco, San Jose, CA, Tech. Rep., 2011.
[27] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1990.
[28] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "LFA applicability in SP networks," Internet Draft, May 2011.
[29] S. Vissicchio, L. Cittadini, M. Pizzonia, L. Vergantini, V. Mezzapesa, and M. L. Papagni, "Beyond the best: Real-time non-invasive collection of BGP messages," in *Proc. INM/WREN*, 2010, p. 9.
[30] L. Vanbever, G. Pardoen, and O. Bonaventure, "Towards validated network configurations with NCGuard," in *Proc. IEEE INM*, 2008, pp. 1–6.
[31] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proc. SIGCOMM*, 2002, pp. 133–145.
[32] J. Yu, "Scalable routing design principles," RFC 2791, 2000.
[33] GEANT, "GEANT backbone topology," 2010 [Online]. Available: http://www.geant.net/network/networktopology/pages/home.aspx
[34] INL, UCL, "Seamless network-wide IGP migrations," 2011 [Online]. Available: http://inl.info.ucl.ac.be/softwares/seamless-network-migration
[35] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. 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, Aug. 2008.
[36] S. Mirtorabi, P. Psenak, A. Lindem, and A. Oswal, "OSPF multi-area adjacency," RFC 5185, 2008.
[37] A. Shaikh, R. Dube, and A. Varma, "Avoiding instability during graceful shutdown of multiple OSPF routers," *Trans. Netw.*, vol. 14, no. 3, pp. 532–542, Jun. 2006.
[38] J. Moy, P. Pillay-Esnault, and A. Lindem, "Graceful OSPF restart," RFC 3623, 2003.
[39] M. Shand and L. Ginsberg, "Restart signaling for IS-IS," RFC 5306, 2008.
[40] I. Pepelnjak, "Changing the routing protocol in your network," 2007.
[41] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful network state migrations," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1097–1110, Aug. 2011.
[42] M. Kaminski, P. Medvedev, and M. Milanic, "Shortest paths between shortest paths," *Theor. Comp. Sci.*, vol. 412, no. 39, pp. 5205–5210, 2011.
[43] R. Keralapura, C.-N. Chuah, and Y. Fan, "Optimal strategy for graceful network upgrade," in *Proc. INM*, 2006, pp. 83–88.
[44] X. Chen, Z. M. Mao, and J. Van der Merwe, "PACMAN: A platform for automated and controlled network operations and configuration management," in *Proc. CoNEXT*, 2009, pp. 277–288.
[45] R. Alimi, Y. Wang, and Y. R. Yang, "Shadow configuration as a network management primitive," in *Proc. ACM SIGCOMM*, 2008, pp. 111–122.
[46] 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. ACM SIGCOMM*, 2008, pp. 231–242.
[47] E. Keller, J. Rexford, and J. Van Der Merwe, "Seamless BGP migration with router grafting," in *Proc. NSDI*, 2010, p. 16.
[48] M. Reitblatt, N. Foster, J. Rexford, and D. Walker, "Consistent updates for software-defined networks: Change you can believe in," in *Proc. HotNets-X*, 2011, pp. 7:1–7:6.
[49] F. Le, G. G. Xie, and H. Zhang, "Theory and new primitives for safely connecting routing protocol instances," in *Proc. ACM SIGCOMM*, 2010, pp. 219–230.

Laurent Vanbever (S'08) received the Master's degree in computer science from the Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 2008, and is currently pursuing the Ph.D. degree in applied sciences at the UCL. He also holds a Master's degree in business management from the Solvay Brussels School of Economics and Management, Brussels, Belgium.

His research interests focus on network management, notably IP routing, network configuration management, and network validation.



Stefano Vissicchio received the Master's degree in computer science from the Roma Tre University, Rome, Italy, in 2008, and is currently pursuing the Ph.D. degree in applied sciences at Roma Tre University.

His research interests are mainly focused on network management. He is currently working on routing configuration design, testing, and deployment.

Cristel Pelsser received the Master's degree in computer science from the Facultés Universitaires Notre-Dame de la Paix (FUNDP), Namur, Belgium, in 2001, and the Ph.D. degree in applied sciences from the Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, in 2006.

From 2007 to 2009, she held a post-doctorate position with NTT Network Service Systems Laboratories, Tokyo, Japan. She is now a Researcher with Internet Initiative Japan (IIJ), Tokyo, Japan. Her current research interests are in Internet routing and privacy in distributed storage systems.

Pierre Francois (M'06) received the B.Sc. degree in economics and management science and Master's degree in computer science from the Facultés Notre Dame de la Paix, Namur, Belgium, in 2000 and 2003, respectively, and the Ph.D. degree in applied sciences from the Université catholique de Louvain, Louvain-la-Neuve, Belgium, in 2007.

He has been a Staff Researcher with Institute IMDEA Networks, Madrid, Spain, since September 2011. He is active in standardization, holding an extensive list of IETF contributions. His main topics of interest are notably IP routing scaling and convergence, Internet governance, Internet routing economics, and network measurements.

Dr. Francois received the IEEE INFOCOM 2007 Best Paper Award.

Olivier Bonaventure (M'92) graduated from the University of Liège, Liège, Belgium, in 1992 and received the Ph.D. degree in applied sciences in 1999.

He spent one year with Alcatel, Antwerp, Belgium. He is now a Full Professor with the Université catholique de Louvain, Louvain-la-Neuve, Belgium, where he leads the IP Networking Lab (http://inl.info.ucl.ac.be) and is Vice-President of the ICTEAM Institute. He has published more than 80 papers, contributes to IETF, was granted several patents.

Prof. Bonaventure served on the Editorial Board of the IEEE/ACM TRANSACTIONS ON NETWORKING. He currently serves as Education Director within ACM SIGCOMM and is a member of the CoNEXT Steering Committee. He received several awards including the IEEE INFOCOM 2005 Best Paper Award.