Seamless Network-Wide IGP Migrations

Technical Report

Laurent Vanbever*, Stefano Vissicchio†,
Cristel Pelsser‡, Pierre Francois*, Olivier Bonaventure*

* Université catholique de Louvain † Roma Tre University ‡ Internet Initiative Japan
*{laurent.vanbever, pierre.francois, olivier.bonaventure} @uclouvain.be
†vissicch@dia.uniroma3.it ‡cristel@iij.ad.jp

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 and service-affecting outages.

In this paper, we propose a methodology which addresses the problem of seamlessly modifying the configuration of commonly used link-state Interior Gateway Protocols (IGP). We illustrate the benefits of our methodology by considering several migration scenarios, including the addition or the removal of routing hierarchy in an existing IGP and the replacement of one IGP with another. We prove that a strict operational ordering can guarantee that the migration will not create IP transit service outages. Although finding a safe ordering is NP-complete, we describe techniques which efficiently find such an ordering and evaluate them using both real-world and inferred ISP topologies. Finally, we describe the implementation of a provisioning system which automatically performs the migration by pushing the configurations on the routers in the appropriate order, while monitoring the entire migration process.

Categories and Subject Descriptors: C.2.3 [Computer-Communication Networks]: Network Operations

General Terms: Algorithms, Management, Reliability

Keywords: Interior Gateway Protocol (IGP), configuration, migration, summarization, design guidelines

1. INTRODUCTION

Among all network routing protocols, link-state Interior Gateway Protocols (IGPs), like IS-IS and OSPF, play a critical role. Indeed, 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, also rely on an IGP to properly work. As the network grows or when new services have to be deployed, network operators often need to perform large-scale IGP reconfiguration [1]. Migrating an IGP is a complex process since all the routers have to be reconfigured in a proper manner. Simple solutions like restarting the network with the new configurations do 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 non-upgraded 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 instance, several operators have switched from OSPF to IS-IS because IS-IS is known to be more secure against control-plane attacks [2, 3]. Operators may also want to migrate to an IGP that is not dependent on the address family (e.g., OSPFv3, IS-IS) in order to run only one IGP to route both IPv4 and IPv6 traffic [4, 3], or to change IGP in order to integrate new equipments which are not compliant with the adopted one [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 [6, 7]. Removing a hierarchy might also be needed, for instance, to better support some traffic engineering extensions [8]. 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 [1]. 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 [9]. Route summarization also

Technical report N.75312
This is an extended version of the publication at SIGCOMM'11. The electronic version of this report is available online `http://hdl.handle.net/2078.1/75312`

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 [2]. There have also been cases where, after having performed a migration, the network no longer met the original requirements, forcing the operators to fallback to the initial configuration [10]. Finally, given the recent trend of deploying virtual networks [11, 12], we believe that the need for reconfiguring the IGP will become more frequent.

In this paper, we aim at enabling *seamless IGP migrations*, that is, progressive modifications of the IGP configuration of a running network without loosing packets. Our main contribution is threefold. First, we analyze in detail the various scenarios of link-state IGP migrations and explain problems that can arise. In particular, we show that long-lasting forwarding loops can appear, both theoretically and practically, when modifications are made to the IGP hierarchy and when route summarization is introduced or removed. To deal with all the identified migration problems, we propose a generic IGP model. Second, we show that, in real-world networks, it is possible to find an ordering for the configuration changes that prevents forwarding loops. 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. Furthermore, we describe how our techniques can be extended to prevent congestion and deal with network failures. Third, we describe the design and the evaluation of a complete system that automates the whole migration process. Our system generates router configurations, assesses the proper state of the network and updates all the routers in an appropriate sequence.

The rest of the paper is structured as follows. Section 2 provides a background on link-state IGPs and presents our abstract model. Section 3 formalizes the IGP migration problem and describes the migration scenarios we tackle. Section 4 presents our methodology. Section 5 proposes algorithms to compute a loop-free migration ordering. Section 6 presents our implementation. Section 7 evaluates our migration techniques on both inferred and real-world topologies. Section 8 defines design guidelines that make IGP migrations easier. Section 9 presents related work. Section 10 discusses the limitation of the approach and the impact on BGP. Finally, Section 11 contains the conclusions.

2. LINK-STATE INTERIOR GATEWAY PROTOCOLS

In this section, we summarize the most important features of link-state IGPs. Then, we present the model we use throughout the paper.

2.1 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 [13] or IS-IS [14]. 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 accCamera-ready main conference papers dueording to the shortest-paths towards 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, that could be represented as links in a logical graph. Each link in the logical graph belongs to only one zone. By extension, we say that a router is in a zone if it has at least one link in that zone. We call *internal routers* the routers that are in one zone only and *Zone Border Routers* (ZBRs) (e.g., ABRs in OSPF and L1L2 systems in IS-IS) the routers that are in more than one zone, among which there must be the backbone. Both internal routers and ZBRs *prefer intra-zone over inter-zone paths*. This means that, to choose the path on which to forward packets towards a certain destination, each router prefers a path in which all the links belong to the same zone over a path containing at least one link belonging to a different zone, no matter the weight 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 weight. The weight 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 weights of the preferred paths towards that destinations (e.g., picking the highest one [13]).

2.2 An Abstract Model for 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 behaviors are not modeled since we ensure that both the initial and the final IGPs have converged before starting the migration process (see Section 4).

We formally define an *IGP configuration* as a tuple $< p, G, D, w, m >$. In such a tuple, p is the identifier of an IGP protocol, e.g., OSPF or IS-IS, and 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 on which the two routers are allowed to exchange protocol-specific messages. Edges are labeled with additional information. In hierarchical configurations they are labeled with the name of zones they belong to. Moreover, $D \subseteq V$ is a 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 to model peculiarities of the considered IGP see Appendix A. To be as generic as possible, we consider as destinations a subset of the IGP routers. Finally, the function $w : E \rightarrow \mathbb{N}$

associates a positive integer, called *weight*, to each edge in G.

Packets destined to one router $d \in D$ follow paths on G. A forwarding path or simply *path* P on G is denoted as $P = (s\ r_1\ \ldots\ r_k\ d)$, where s is the first router in G that is required to forward packets destined to d, and routers r_i, with $i = 1, \ldots, k$, are routers traversed by the traffic flow. 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 towards each destination and forwards packets to the next-hops in such preferred paths. To capture this behavior, we define the *next-hop* function nh, and the *actual path* function $\pi(u, d, t)$. We denote with $nh(u, d, t)$ the set of successors (next-hops) of u in the paths router u uses at time t to send traffic destined to destination d. Notice that $|nh(u, d, t)|$ is not guaranteed to be equal to 1, since our model encompasses the case in which routers uses multiple paths to reach the same destination, e.g., Equal Cost Multi-Path (ECMP). The paths actually followed by packets sent by u towards 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 $\{(v_0\ v_1\ \ldots\ v_k)\}$, such that $v_0 = u$, $v_k = d$ and $v_{i+1} \in nh(v_i, d, t)$, $\forall i \in \{0, \ldots, k-1\}$. Notice 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 [15]. A series of deflections can even build a forwarding loop, as shown in different examples described in Section 3.1. More formally, we say that there exists a *forwarding loop*, or simply a *loop*, for a certain destination d at a certain time t if $\exists r$ such that $\pi(r, d, t) = (r\ v_0\ \ldots\ v_j\ r)$, with $j \geq 0$.

By appropriately tuning the next-hop function, our model is able to represent specific details of IGP configurations such as the corresponding forwarding rules in hierarchical and flat mode, and route summarization. In Section 3.1, we provide some examples of next-hop functions, actual path functions, and migration loops in different migration scenarios.

3. THE 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 an input (i.e., by network operators) and are loop-free.

PROBLEM 1. *Given a unicast IP network, how can we replace an initial IGP configuration with a final IGP configuration without causing any forwarding loop?*

Assuming no congestion and no network failures, solving this problem leads to seamless migrations. These assumptions are reasonable, since management operations are typically performed during convenient time slots, in which traffic is low. Moreover, our approach is time efficient, reducing the opportunities for failures during the migration process. Also, we discuss how to extend our techniques to remove these assumptions in Section 5.2.

In the rest of the paper, we call *router migration* the replacement of nh_{init} with nh_{final} on one router. Formally, we define the operation of *migrating a router* r at a certain time $\bar{t}$, the act of configuring the router such that $nh(r, d, t) = nh_{final}(r, d)$, $\forall d \in D$ and $\forall t > \bar{t}$. Since only one IGP can be configured to control the forwarding of each router (i.e., either the initial or the final), routers cannot be migrated on a per-destination basis. We call *router migration ordering* the ordering in which routers are migrated. A network migration is completed when all routers have been migrated.

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

Table 1: IGP Migration Scenarios.

Throughout the paper, we consider only *migration loops*, that is, loops arising during an IGP migration because of a non-safe router migration ordering. Migration loops are not-protocol dependant, and can be longer and more harmful than loops that arise during protocol convergence, since migration loops last until specific routers are migrated (e.g., see Section 3.1). Observe that, if the nh function does not change, the π function does not change either, hence any migration order does not create loops during the migration.

3.1 IGP migration scenarios

Table 1 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 have to be changed. We do not consider the change of link weights as a network-wide migration. Since traffic matrices tend to be almost stable over time [16], ISPs can prefer to progressively change the weight of few links at a time. Effective techniques have been already proposed for the graceful change of few link weights [17, 18, 19, 20, 21]. Nevertheless, our generalized model and the techniques we present in Section 5 can also be used when link weights have to be changed. In the following, we describe the issues that must be addressed in each scenario using the notation introduced in Section 2.2.

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 [1]. Since the nh function is the same in both IGPs, routers could be migrated in any order without creating loops.

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

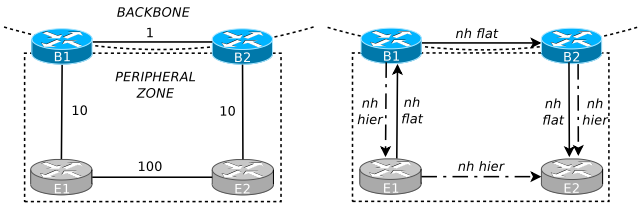


Figure 1: Bad Square Gadget. When the IGP hierarchy is modified, a given migration ordering is needed between $B1$ and $E1$ to avoid forwarding loops.

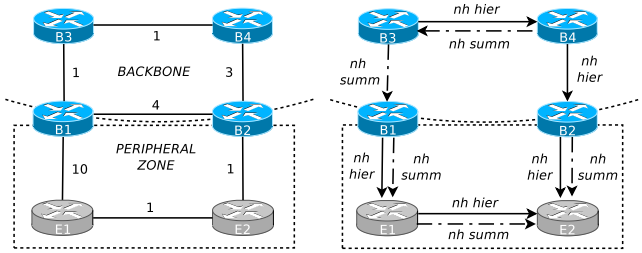


Figure 3: Route summarization gadget. When summarization is introduced or removed, a specific migration ordering is needed between $B3$ and $B4$ to avoid forwarding loops.

refer to these scenarios as *flat2hier*, *hier2flat* and *hier2hier*, respectively.

Unlike to 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 intra-zone over inter-zone path preference rule applied by routers in hierarchical IGPs (see Section 2). 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 towards 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 $nh_{final}(B1, E2) = \{E1\}$ and $nh_{init}(E1, E2) = \{B1\}$. 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 [14]. 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.

Route summarization

Introducing or removing route summarization (i.e., *summ* 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. It is evident that 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 $B4$.

4. METHODOLOGY

Fig. 4 illustrates the main steps of our methodology. In the first step, we pre-compute an ordering in which to seamlessly migrate routers, with no packet loss (Section 5). When such an ordering does not exist, we use technical fallback solutions see Appendix C. Fallback solutions are only exploited as a backup since they make the whole migration process slower and harder to pilot. As we always could find an ordering for all the ISP topologies we analyzed (Section 7), we believe that fallback solutions will be rarely needed in practice.

The actual migration process begins in the second step of our methodology. As a basic operation, we exploit a known migration technique called *ships-in-the-night* [1, 2, 4], 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 Administrative Distance (AD). When a route for a given prefix is available from several processes, the one with the lowest AD is installed in the 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 have reached a stable routing state. In the fourth step, we progressively migrate routers following the ordering pre-computed in the first Step of the methodology. 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. Since a rout-

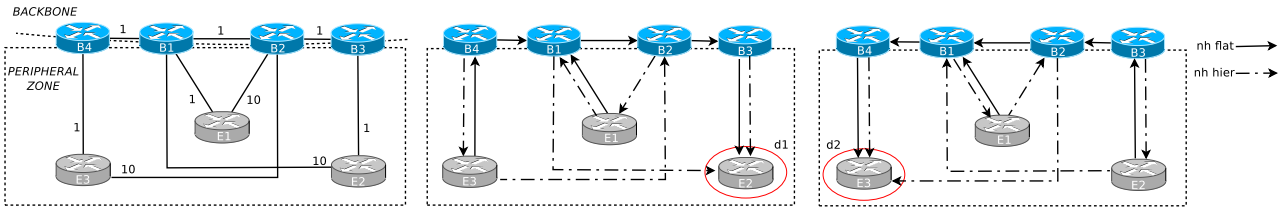


Figure 2: Loop Gadget. No migration ordering is loop-free for $flat2hier$ and $hier2flat$ scenarios because of contradictory constraints between $B1$ and $B2$.

Steps for a Seamless IGP Migration

1. *Compute a loop-free order* in which routers can be migrated without creating loops.
2. *Introduce the final IGP configuration network-wide.* In this step, routers still forward packets according to the initial IGP configuration only.
3. *Monitor the status* of the IGP configurations. Wait for the convergence of the final IGP configuration.
4. *Migrate routers* following the pre-computed loop-free order. Migrating a router means configuring it to forward packets according to the final IGP configuration.
5. *Remove the initial IGP configuration* from all the routers.

Figure 4: Proposed methodology for seamless IGP migrations.

ing entry change takes about $100ms$ before being reflected in the FIB [23], we wait for a given amount time (typically 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 [24]. Lowering the AD on 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 backtracking to a previously working state when a problem is detected, e.g., a router that does not receive an intended command. All the discussions that we had with network operators further confirm that they prefer to gradually migrate their network to have full-control of the process.

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

5. 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. Firstly, we present the algorithms we use to compute a loop-free router migration ordering. Then, we discuss how to adapt the algorithms to address congestion and network failures.

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

Figure 5: Loop Enumeration Algorithm.

5.1 Migration Ordering Computation

We now study the following problem from an algorithmic perspective.

PROBLEM 2. *Given an initial and a final next-hop functions, 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, that we call *RMOP*, is an $\mathcal{NP}$-complete problem. Indeed, a reduction from the well-known 3-SAT problem [25] can be built in polynomial time. The complete proof is described in Appendix B.

In the following, we present an algorithm to find a loop-free ordering (when it exists). Because of the complexity of the problem, the algorithm is inefficient and can take several hours to run on very large ISP networks (see Section 7). We also propose an efficient heuristic that is correct but not complete.

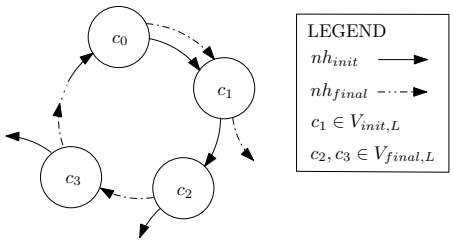


Figure 6: Abstract representation of a migration loop.

Loop Enumeration Algorithm

The Loop Enumeration Algorithm (Fig. 5) enumerates all the possible migration loops that can arise during a migration, and 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 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. 6, indeed, migrating c_1 before at least one among c_2 and c_3 avoids the loop. In the algorithm, $V_{init,L}$ represents the set of routers that participate in the loop when they are in the initial configuration (line 6), and $V_{final,L}$ 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 prove that the loop enumeration algorithm is correct and complete. This depends on the fact that the constraints it outputs encode the sufficient and necessary conditions to prevent loops during the migration.

LEMMA 1. *Let $u_0 \vee \cdots \vee u_k < v_0 \vee \cdots \vee v_l$ be the ordering constraint that the Loop Enumeration Algorithm identifies for a loop L concerning traffic destined to $d \in D$. Then, L does not arise during the migration if and only if the constraint is satisfied.*

PROOF. Let $L = (c_0\ c_1\ \ldots\ c_k\ c_0)$. We prove the statement in two steps.

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

```
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
```

Figure 7: Routing Trees Heuristic.

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

THEOREM 1. *The Loop Enumeration Algorithm is correct and complete.*

PROOF. The statement follows by Lemma 1, since the linear program encodes all the sufficient and necessary conditions for any migration loop to not arise. □

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. 7. 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 the directed acyclic graph G_d [1] containing 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 $(u \dots v \dots d) \in \pi_{final}(u, d)$, 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 towards 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 towards 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 2. *If the greedy procedure adds a router u to S_d, then u cannot be part of any migration loop towards 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 $(u\ v_0\ \dots\ v_k\ u) \in \pi(u, d, t)$, 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 $w_0, \dots, w_n$ (in both the initial and final IGP configurations) are already in S_d, since each node in $\{w_0, \dots, w_n\}$ 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, $\forall h = 0, \dots, k$. As a consequence, GREEDY does not add u to S_d at iteration i, which is a contradiction. □

THEOREM 2. *Let $S = x_1, \dots, x_n$ 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 $\tilde{V} = \{v_1, \dots, v_k\}$, such that $C = (u\ v_0\ \dots\ v_k\ u) \in \pi(u, d, t)$, at a certain time t. By Lemma 2, all $v_i \notin S_d$. By definition of the heuristic, all routers v_i are such that $nh(v_i, d, t) = nh_{final}(v_i, d)$, with $i = 0, \dots, k$, 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 towards d, even if it is not needed. For instance, in the scenario of Fig. 8, 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, if state of the art best practices for PoP design are followed [26], such cases are rare. In Section 7, we show that the heuristic found an ordering in most of our experiments on realistic topologies.

[1] G_d is an acyclic since the final configuration is loop-free

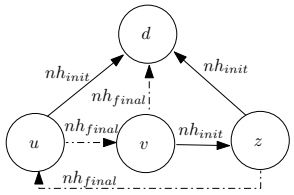


Figure 8: In some migration scenarios, the Routing Trees Heuristic generates unnecessary constraints.

5.2 Dealing with Congestion and Failures

Even if there is no congestion in both the initial and the final IGP configurations, congestion could transiently appear during the migration because of forwarding paths in temporary states in which only some routers are migrated. To deal with congestion, we can add constraints (e.g., routers u and v must not be migrated both before z) that the algorithms must take into account in looking for a proper router migration ordering. Assuming that the traffic matrix does not consistently change during the migration, such constraints can be statically computed, given the traffic matrix, the capacity of the links, and the nh function. Note that the assumption on the stability of the traffic matrix is reasonable since traffic shifts are rare for the most popular destinations [16] and our approach requires a short time to complete the migration process (see Section 7).

On the other hand, link or node failures modify the topology, hence they may modify the nh function and the loop-free migration ordering to be followed. Thanks to the high time efficiency of our heuristic (see Section 7), we can precompute loop-free router orderings and the corresponding ordering constraints that are needed for seamless migrations in the most important failure scenarios (e.g., all possible single link failures). When a failure happens, we can use such constraints to minimize the duration of the loops generated by the failure, and to dynamically adapt the order in which migration steps are performed. Because of the high efficiency of the heuristic on small and medium-sized topologies, we can even directly recompute the ordering just after the failure, taking into account the fact that some routers could have already been migrated.

6. THE PROVISIONING SYSTEM

We implemented a software system which is able to compute and automate all the required steps for a seamless migration. The main architectural components of our system are represented in Fig. 9. In the following, we describe how data flow through the system (dashed lines in the figure), by stressing the role of each component.

In order to assess the properties of the initial and the final IGPs, we rely on a monitoring system which collects the IGP Link-State Advertisements (LSAs) circulating in the net-

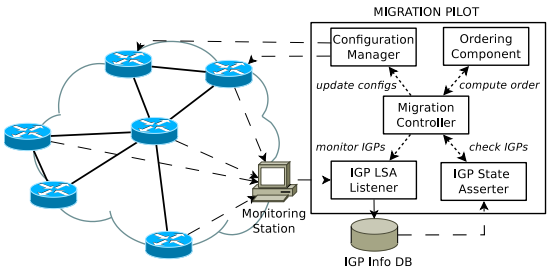


Figure 9: System architecture.

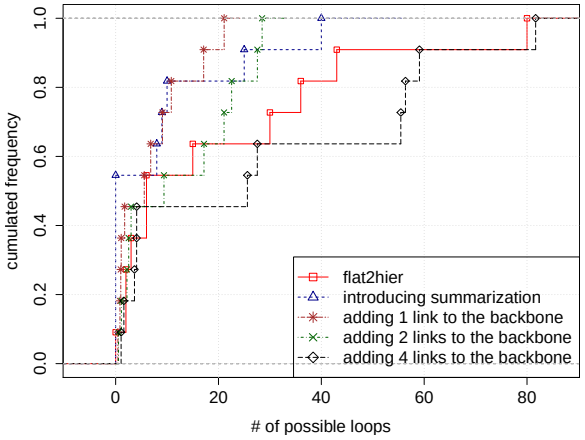


Figure 10: CDF of the number of loops that can arise on Rocketfuel topologies.

work. The *IGP LSA Listener* component parses the LSAs, continuously filling a database (DB) with data on the IGP adjacencies, on the weight of the links, and on the announced IP prefixes. We implemented the IGP LSA Listener by using packet-cloning features available on routers [27]. The *IGP State Asserter* component is responsible for querying the DB and assessing properties of the monitored IGPs state. The current implementation of the IGP State Asserter is able to check an IGP for convergence completion by evaluating the stability over time of the IGP adjacencies and of the prefixes announced by each router. A custom time threshold can be set to assess the stability of the IGP. 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 towards 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 5.1. Then, the Migration Controller runs the IGP LSA Listener. When needed (see Section 4), 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 4. 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 [28].

7. 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.

7.1 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 [29]. 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 adding 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 [30]. All the tests were run on a Sun Fire X2250 (quad-core 3GHz CPUs with 32GB of RAM). We omit the results of some experiments due to space limitations.

7.2 Ordering Algorithms

We first evaluate usefulness and efficiency of the Loop Enumeration Algorithm and of the Routing Tree Heuristic. Fig. 10 shows the cumulative distribution function 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

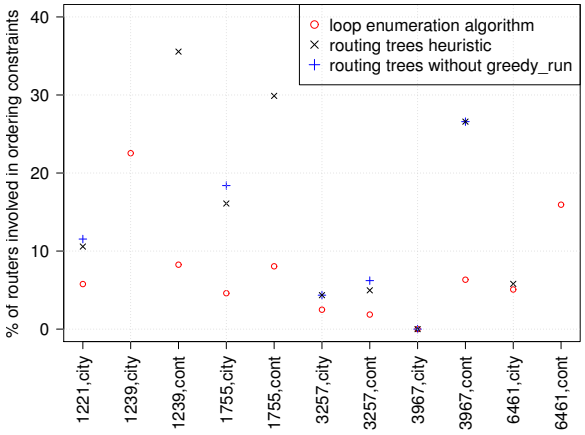


Figure 11: Percentage of routers involved in the ordering in *flat2hier* (Rocketfuel topologies). Results for other scenarios are similar.

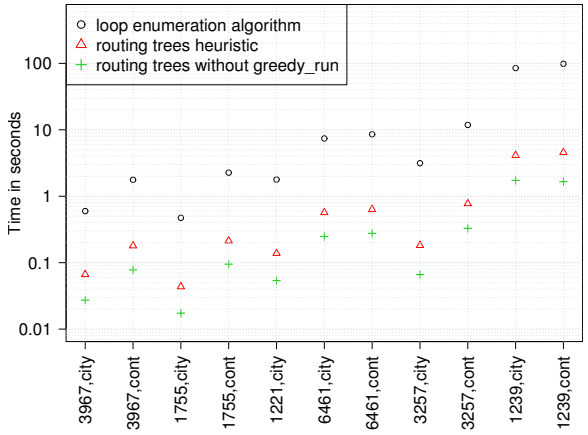


Figure 12: Time taken to compute an ordering in *flat2hier* (Rocketfuel topologies). Results for other scenarios are similar.

worst case and at least 30 loops can arise for 4 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 concerns 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. It is, thus, important to compute an ordering in which they are not involved in loops. 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, it is desirable that migrations of world-wide networks be carried out on a per-zone basis, that is, migrating all the routers in the same zone (e.g., a continent) before routers in other zones. We observe that this is indeed possible since all the loops that occur, in both Rocketfuel and real-world topologies, arise between routers in the same zone or between backbone routers and routers in a peripheral zone. Thus, it is often possible to compute per-zone orderings. These considerations further motivate our effort to find a router migration ordering which 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. We discuss simple design guidelines that ease IGP migrations in Section 8. On the other hand, we found that huge number of problems could arise in a migration from a poor design to a neat one. In the $hier2flat$ scenario, more than 2000 loops, involving up to 10 routers, might arise within the $tier1.A$. Such a large number of loops is mainly a consequence of the way we built the hierarchy.

As a second group of experiments, we ran the ordering algorithms on the Rocketfuel topologies. In the following, we present results for the $flat2hier$ scenario but similar results and considerations hold for the other scenarios. Fig. 11 shows for each 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. 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. Fig. 11 also highlights the gain of relying on the greedy subprocedure, as the heuristic could find a solution for only 6 topologies without it.

Fig. 12 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 that cases, the 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. 12, 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 hours 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 minutes. Typically, the time taken by the ordering algorithm is not a critical factor in our approach, since a loop-free router migration ordering can be computed before actually performing the migration. However, time efficiency is highly important to support advanced

abilities like promptly reacting to failures that could happen during the migration (see Section 5.2).

7.3 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 [31]. 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 [32]. In addition to the backbone (12 routers), we defined three peripheral zones: the south west area (6 routers), the north east area (11 routers) and the south east 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 5.1), we found that 8 different loops towards 5 different destinations could arise on that topology.

We ran two experiments. In the first experiment, we relied on the ordering computed by the Loop Enumeration Algorithm, while in the second we adopted a random order. In order to have statistically relevant data, 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 towards the 5 troublesome destinations. Fig 13 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. 13 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. Finally, we observe that, even migrations on a per-zone basis require the use of an ordering algorithm because all the ordering constraints are among routers in the same zone.

Our system also enables faster migrations than known migration [2, 4]. 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 milliseconds on average and 24 milliseconds 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 milliseconds 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 4). On average, the Configuration Manager took 8.84 seconds 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 minutes. On the contrary, a similar real-world Geant migration took several days to be completed [4].

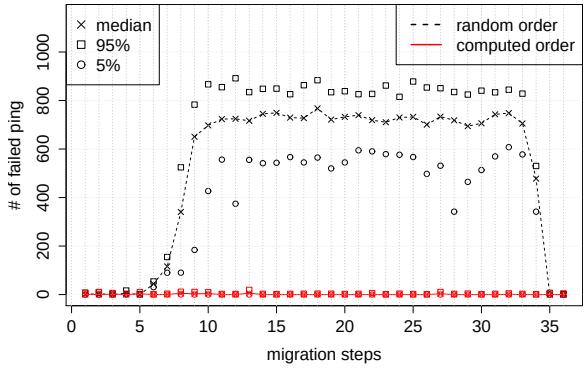


Figure 13: Our system guarantees that no packet is lost during migration while long-lasting connectivity disruptions can happen with a naive approach.

All the intermediate configurations that our system generated in the case study described above are available online [32].

8. 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 intra-zone 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 intra-zone path is preferred in both flat and hierarchical modes. Since no router in Z changes its path, then $nh_{init}(v,d) = nh_{final}(v,d)$ 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 intra-zone paths, then non-ZBR routers cannot use inter-zone paths. Establishing multiple adjacencies (e.g., $L1L2$ adjacencies in IS-IS) between ZBRs also guarantees the nh function does not change, but backbone links could be unnecessarily traversed in this case.

GUIDELINE B. *For each zone Z, an aggregation layer of routers connects the $ZBRs$ to the destinations in Z (e.g., BGP border routers or MPLS PE). Link weights are set so that the weight of the path from any ZBR to any destination in Z is the same.*

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 [13]). 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.

9. RELATED WORK

Seamless IGP operation and maintenance have been the focus of many previous studies. For example, several protocol extensions have been proposed [33, 34, 35] to gracefully restart a routing process. However, few research effort has been specifically devoted to network-wide IGP migrations.

In [17], Raza *et al.* propose the Graceful Network Operations (GNO) framework which formalizes the problem of minimizing a certain disruption function (e.g., link congestion) when the link weights change. They also describe an algorithm to find a congestion-free ordering when several IGP weights have to be modified. 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 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 could not exist. In [36], Keralapura *et al.* study the problem of finding the optimal way in which to add nodes and links to a network to minimize disruptions. Even if our techniques can be adapted to address topological changes, this problem is beyond the focus of this paper.

In [37], 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 [38], 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 looks promising, it cannot be exploited on current routers and a commitment ordering could still be needed.

Recently, some techniques [12, 39] 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 work differ from ours as we aim at seamlessly changing network-wide configurations.

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 [40]. 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 in practice to carry out IGP migrations since they assume that the FIB can be updated on a per-destination basis which is not the case on current routers.

IGP migrations could also be performed by using route redistribution. Although new primitives have been recently proposed [41], we believe that relying on a ships-in-the-night approach (when possible) makes the entire migration process easier and more manageable.

10. DISCUSSION

In this section, we discuss the limitations of our methodology, especially in terms of its application to other types of migrations.

First of all, our methodology is adapted to link-state IGP migrations and it cannot be directly applied to other IGPs (e.g., distance-vector IGPs). Contrary to link-state protocols, where routers always have a global view of the topology and can take decisions autonomously, in distance-vector protocols a change of the next-hop of one router can affect the visibility other routers have of some destinations. This poses different problems with respect to those tackled by our techniques.

Moreover, our methodology does not take into consideration the interactions between the changing IGP and the protocols relying on it. In particular, our approach is not suitable for all the scenarios in which BGP is deployed on the migrated network. BGP uses the IGP to both discriminate among several exit-points and to learn how to reach the preferred exit-point [42]. Migrating the underlying IGP can thus cause BGP routers to change their preferred exit-point which can lead to forwarding loops. Currently, our algorithms ensure that no loop occurs during the migration towards any internal destination of an AS. For all the networks whose forwarding is based on tunneling or encapsulation mechanisms like MPLS, this property is sufficient to guarantee loop-free forwarding towards inter-domain destinations as well. Indeed, no loop occurs in the IGP and the tunneling mechanism ensures that the BGP traffic will reach the proper exit-point. Though, in the migration of a pure IP network, the exclusive presence of BGP can induce forwarding loops due to conflicting BGP decisions between updated routers and non-updated routers. Theoretically, our ordering algorithms can be adapted to deal with BGP-induced loops. However, the ordering problem is much more complicated since it needs to consider: (i) the fact that BGP prefixes could be reached via any combination of exit-points, (ii) the iBGP topology and its relationship to the IGP [43], and (iii) BGP dynamism. For these reasons, we expect that a loop-free migration ordering for all the BGP prefixes does not exist in most of the cases. We believe that finding an effective technique to prevent BGP-induced loops during the migration of pure IP networks is an interesting open problem raised by this paper.

11. CONCLUSIONS

Network-wide IGP migrations are a source of concerns for network operators. Unless carried on with care, IGP migrations can cause long-lasting forwarding loops and thus significant packet losses. In this paper, we proposed a migration strategy that enables network operators to migrate an entire IGP configuration seamlessly, rapidly, and without compromising routing stability. Our strategy relies on effec-

tive 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.

Although we focused on link-state IGP migrations, the applicability of our techniques is broader since it can encompass any migration issues involving changes of next-hops. We plan to make our approach suitable for seamless migrations involving distance-vector IGPs. Also, we plan to study seamless migrations of other routing protocols (e.g., MPLS or BGP). Our vision is that network-wide migrations could become a basic operation enabling the seamless replacement or reconfiguration of any protocol.

12. ACKNOWLEDGEMENTS

We thank Luca Cittadini, Randy Bush, Bruno Quoitin, Virginie van den Schriek and our shepherd, Walter Willinger, for their help in improving the paper. This work was partially supported by Alcatel-Lucent. Laurent Vanbever is supported by a FRIA scholarship. Stefano Vissicchio is partially supported by ESF project 10-EuroGIGA-OP-003 GraDR and MIUR PRIN Project ALGODEEP. Pierre Francois is supported by the "Fonds National de la Recherche Scientifique", Belgium.

13. REFERENCES

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

[2] V. Gill and M. Jon, "AOL Backbone OSPF-ISIS Migration," NANOG29 Presentation, 2003.

[3] NANOG thread, "IPv6: IS-IS or OSPFv3," http://mailman.nanog.org/pipermail/nanog/2008-December/006194.html, 2008.

[4] "Results of the GEANT OSPF to ISIS Migration," GEANT IPv6 Task Force Meeting, 2003.

[5] NANOG thread, "OSPF -vs- ISIS," http://www.merit.edu/mail.archives/nanog/2005-06/msg00406.html, 2005.

[6] B. Decraene, J. L. Roux, and I. Minei, "LDP Extension for Inter-Area Label Switched Paths (LSPs)," RFC 5283 (Proposed Standard), Jul. 2008.

[7] T. M. Thomas, *OSPF Network Design Solutions, Second Edition*. Cisco Press, 2003.

[8] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for Inter-Area MPLS Traffic Engineering," RFC 4105, Jun. 2005.

[9] N. Leymann, B. Decraene, C. Filsfils, M. Konstantynowicz, and D. Steinberg, "Seamless MPLS Architecture," Internet draft, 2011.

[10] P. Templin, "Small Network Operator - Lessons Learned," NANOG45 Presentation, 2009.

[11] "Virtual Network Operator (VNO) hosting opportunities with orbit research." [Online]. Available: http://www.orbitresearch.co.uk/services/virtual-network-operator

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

[13] Moy, J., "OSPF Version 2," RFC 2328, 1998.

[14] Oran D., "OSI IS-IS Intra-domain Routing Protocol," RFC 1142, 1990.

[15] S. Iasi, P. François, and S. Uhlig, "Forwarding deflection in multi-area OSPF," in *CoNEXT*, 2005.

[16] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP Routing Stability of Popular Destinations," in *Proc. IMW*, 2002.

[17] S. Raza, Y. Zhu, and C.-N. Chuah, "Graceful Network Operations," in *Proc. INFOCOM*, 2009.

[18] P. Francois and O. Bonaventure, "Avoiding transient loops during the convergence of link-state routing protocols," *Trans. on Netw.*, vol. 15, pp. 1280–1292, December 2007.

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

[20] J. Fu, P. Sjodin, and G. Karlsson, "Loop-free updates of forwarding tables," *Trans. on Netw. and Serv. Man.*, vol. 5, no. 1, pp. 22–35, 2008.

[21] L. Shi, J. Fu, and X. Fu, "Loop-free forwarding table updates with minimal link overflow," in *Proc. ICC*, 2009.

[22] H. Ballani, P. Francis, T. Cao, and J. Wang, "Making routers last longer with ViAggre," in *Proc. NSDI*, 2009.

[23] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving sub-second IGP convergence in large IP networks," *Comput. Commun. Rev.*, vol. 35, no. 3, pp. 33–44, 2005.

[24] 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) Technical Report," Cisco, Tech. Rep., 2011.

[25] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. Freeman, 1990.

[26] C. Filsfils, P. Francois, M. Shand, B. Decraene, J. Uttaro, N. Leymann, and M. Horneffer, "LFA applicability in SP networks," Internet Draft, May 2011.

[27] 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*, 2010.

[28] L. Vanbever, G. Pardoen, and O. Bonaventure, "Towards validated network configurations with NCGuard," in *Proc. of Internet Network Management Workshop 2008*, Orlando, USA, October 2008, pp. 1–6.

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

[30] J. Yu, "Scalable Routing Design Principles," RFC 2791, 2000.

[31] "GEANT Backbone Topology," 2010, http://www.geant.net/network/networktopology/pages/home.aspx.

[32] "Seamless Network-Wide IGP Migrations," 2011, http://inl.info.ucl.ac.be/softwares/seamless-network-migration.

[33] A. Shaikh, R. Dube, and A. Varma, "Avoiding instability during graceful shutdown of multiple OSPF routers," *Trans. on Netw.*, vol. 14, pp. 532–542, June 2006.

[34] J. Moy, P. Pillay-Esnault, and A. Lindem, "Graceful OSPF Restart," RFC 3623, 2003.

[35] M. Shand and L. Ginsberg, "Restart Signaling for IS-IS," RFC 5306, 2008.

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

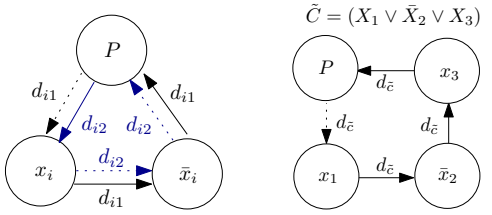
[37] X. Chen, Z. M. Mao, and J. Van der Merwe, "PACMAN: a platform for automated and controlled network operations and configuration management," in *Proc. CONEXT*, 2009.

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

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

[40] P. Francois and O. Bonaventure, "Avoiding transient loops during IGP Convergence in IP Networks," in *Proc. INFOCOM*, 2005.

[41] F. Le, G. G. Xie, and H. Zhang, "Theory and new primitives for safely connecting routing protocol instances," in *Proc. SIGCOMM*, 2010.



(a) Variable Gadget. (b) Clause Gadget.

Figure 14: Gadgets used in the reduction from 3-SAT to RMOP. Solid lines represent nh_{init}, while dotted lines nh_{final}. Edges are labeled with destinations they refer to.

[42] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, 2006.

[43] T. Griffin and G. Wilfong, "On the Correctness of IBGP Configuration," *SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 17–29, 2002.

[44] S. Mirtorabi, P. Psenak, A. Lindem, and A. Oswal, "OSPF Multi-Area Adjacency," RFC 5185, 2008.

[45] S. Mirtorabi, P. Psenak, and A. Lindem, "OSPF Tunnel Adjacency," Internet draft, 2005.

APPENDIX

A. MODELING OSPF DESTINATIONS

In hierarchical OSPF, each interface is bound to a specific OSPF area, which means that OSPF routers can prefer different paths to reach different interfaces of the same destination router. Moreover, when a router r redistributes an external prefix in OSPF, traffic towards the external prefix can be forwarded to any of the interfaces of r, since r injects the same OSPF LSAs in all the areas it participates in.

In order to represent these peculiar features of OSPF, we can enrich our model with virtual nodes. This requires the logical graph to become a multi-graph. In particular, for each router r that coincides with a destination for some (control-plane or data-plane) traffic, we can add a virtual node r_j for each OSPF area j in which r participates, and a virtual node r_{ext} 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 $w(e_j) = 1$. Finally, only virtual nodes are then added to the destination set D.

Virtual nodes can also be used to model IP prefixes announced by more than IGP routers. In this case, it is sufficient to add to the logical graph one node x for the IP prefix and edges from each router that announces the prefix to x. Again, only virtual nodes are added to D.

B. RMOP IS NP-COMPLETE

In order to prove the complexity of the RMOP problem, we use a reduction from 3-SAT [25]. 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 $S =< G = (V, E), D, nh_{init}, nh_{final} >$ 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. 14(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 14(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}, $nh_{init}(u, d_{i1}) = nh_{final}(u, d_{i1}) = \{d_{i1}\}\ \forall u \in V$, except $nh_{init}(x_i, d_{i1}) = \{\bar{x}_i\}$, $nh_{init}(\bar{x}_i, d_{i1}) = \{P\}$, and $nh_{final}(x_i, d_{i1}) = \{\bar{x}_i\}$. Similarly, for each d_{i2}, $nh_{init}(u, d_{i2}) = nh_{final}(u, d_{i2}) = \{d_{i2}\}\ \forall u \in V$, except $nh_{init}(P, d_{i2}) = \{\bar{x}_i\}$, $nh_{final}(x_i, d_{i2}) = \{\bar{x}_i\}$, and $nh_{final}(\bar{x}_i, d_{i1}) = \{P\}$. Finally, for each $\tilde{d}_j$ correspondingt to a clause $C_j = (L_{j1}\ L_{j2}\ L_{j3})$, $nh_{init}(u, \tilde{d}_j) = nh_{final}(u, \tilde{d}_j) = \{\tilde{d}_j\}\ \forall u \in V$, except $nh_{final}(P, \tilde{d}_j) = \{l_{j1}\}$, $nh_{init}(l_{j1}, \tilde{d}_j) = \{l_{j2}\}$, $nh_{init}(l_{j2}, \tilde{d}_j) = \{l_{j3}\}$, and $nh_{init}(l_{j3}, \tilde{d}_j) = \{P\}$.

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 a destination among d_{i1} and d_{i2} if and only if P is the first node to be migrated or P is the very last one in the migration order, respectively.

PROPERTY 1. *A router migration ordering does not create a loop towards 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 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 3. *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,

- 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, all router migration orderings on S contain at least one loop.

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

C. FALLBACK SOLUTIONS

Although unlikely, in some situations no ordering exists or could take to much time to be computed. In this appendix, we explain the intuition behind fallback solutions that, when applicable, still guarantee seamless migration. The idea behind these fallback solutions is to make the initial and the final next-hop functions equal, so that routers could be migrated in any order. Nonetheless, these operations consume more time and are more complicated than the ordering described in Section 5.

To equate the next-hop functions in migration scenarios involving hierarchical IGPs, we propose to establish additional IGP adjacencies between ZBRs, one per shared zone. These adjacencies ensure that all the paths available to ZBRs are intra-area paths. If the IGP does not support multiple adjacencies either natively or by the help of extensions [44, 45], additional adjacencies could be established over virtual links (e.g. IP tunnel). Notice that links could be added and removed safely by using reweighting techniques such as [19]

When no ordering is possible in summarization scenarios, we propose to modify the cost of the aggregate propagated by the ZBRs to preserve the next-hop function for each router outside the summarized zone. Such costs can be automatically computed using Linear Programming (LP) techniques. If no suitable costs assignment can be computed (i.e., the problem is unfeasible), the aggregate can be splitted in sub-aggregated. Again, LP techniques can be exploited to compute the optimal grouping of prefixes and their associated costs such that no router outside the zone change its path towards any aggregated destinations.