Correctness of Routing Vector Protocols as a Property of Network Cycles

João Luís Sobrinho, *Senior Member, IEEE, Member, ACM*

***Abstract*— Most analyses of routing vector protocols, such as the Border Gateway Protocol (BGP), are conducted in the context of a single destination in a given network. In that context, for arbitrary routing policies, it is computationally intractable to determine whether or not a routing vector protocol behaves correctly. In this paper, we consider the common scenario where routing policies are specified independently of the destination. In this scenario, we demonstrate that the correctness of a routing vector protocol for all destinations in a given network equates to a property of routing policies around its cycles, designated strict absorbency, similarly to the way that the correctness of a distance vector protocol equates to cycles of positive length. A number of pragmatic conclusions can be derived from this theoretical result. For example, we show that all next-hop routing policies, which are popular in inter-domain routing and in the interconnection of routing instances, cannot fully exploit the physical redundancy of a network. As another example, we show how sibling autonomous systems of the Internet can share all routes between them without introducing oscillations into BGP.**

***Index Terms*— Correctness of routing vector protocols, routing algebra, inter-domain routing, BGP, routing policies.**

I. Introduction

ROUTING vector protocols are distributed algorithms that find paths in a network over which data-packets are subsequently forwarded. These protocols instantiate a separate computation process for each destination. Such a process starts when the destination originates a route that it sends to all its neighbors. Nodes iteratively extend the routes received from neighbors into candidate routes, elect a route from among the candidate routes, and send the elected route to all their neighbors. Routing policies determine how an elected route sent by one node extends to a candidate route at a neighbor node and which route is elected from among a set of candidate routes.

The Border Gateway Protocol (BGP) [1] is the preeminent routing vector protocol, allowing for the implementation of a broad class of routing policies. It is the inter-domain routing protocol of the Internet, deployed as well in enterprise networks [2] and in data-centers [3]. Other routing vector protocols in current usage are associated with restricted classes of routing policies. For example, the Routing Information Protocol (RIP) [4] is a classical distance vector protocol. It implements routing policies based on lengths, finding shortest paths in a network. The Interior Gateway Routing Protocol (IGRP) [5] and the Enhanced IGRP (EIGRP) [5] implement routing policies based both on lengths and capacities, finding paths according to a combination of these two metrics. A less obvious example of a routing vector protocol lies in the interconnection of routing instances by border routers through configuration of the Administrative Distance (AD) parameter and of route redistribution [6]–[8]. The routing policies in this case are called next-hop [9]–[11]. They are characterized by the fact that the only routing information learned by one node from another is whether or not a destination is reachable. Routing vector protocols also have a presence in wireless networks [12], where routing policies typically combine many metrics that reflect distinct properties of wireless links [13], [14].

Routing vector protocols are simple and they scale well. Nonetheless, it is straightforward to devise routing policies that make them oscillate indefinitely or that trap data-packets in forwarding loops [7], [15], [16]. Such behaviors are undesirable. A routing vector protocol is *correct* if it always terminates in a stable state devoid of forwarding loops. The design, configuration, and analysis of routing vector protocols rests much on our ability to characterize the routing policies that lead to correctness. For a given destination, Griffin *et al.* [16] showed that the existence of a stable routing state is an NP-hard problem. Later, Engelberg *et al.* [17] showed that termination of a routing vector protocol is an undecidable problem.

In the present paper, we demonstrate that if routing policies are set *independently* of the destination, then correctness of a routing vector protocol for *all* destinations equates to a property of routing policies around the cycles of the network, which we call *strict absorbency*: a routing vector protocol is correct *if and only if* all cycles in the network are strictly absorbent. This result can be regarded as a generalization to arbitrary routing policies of the well-known fact that a distance vector protocol is correct if and only if all cycles in the network have positive length [18], with the concept of strictly absorbent cycle generalizing that of a cycle of positive length. However, the behaviors of a routing vector protocol under arbitrary routing policies are far more diversified than those of a distance vector protocol. The equivalence between correctness of a routing vector protocol and strictly absorbent cycles is harder to prove that its specialization to shortest paths. The sufficiency of strict-cycle-absorbency for correctness follows from a conclusion of [19]. The necessity is emphasized in this paper.

Manuscript received September 24, 2014; revised October 13, 2015 and April 26, 2016; accepted May 9, 2016; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Y. Ganjali. This work was supported in part by the Fundação para a Ciência e Tecnologia through the Portuguese Ministry of Science and Higher Education under Project UID/EEA/50008/2013.

The author is with Instituto de Telecomunicações and Instituto Superior Técnico, Lisbon 1049-001, Portugal (e-mail: joao.sobrinho@lx.it.pt).

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.2016.2567600

It bears stressing that the premise of routing policies being set independently of the destination is common in practice. For example, lengths in RIP, and lengths and capacities in IGRP and EIGRP, are assigned to links no matter what the destination IP prefix. In the case of inter-domain routing, network operators typically configure their BGP routers as a function of economic relationships their Autonomous Systems (ASs) have with their neighbor ASs, regardless of destination IP prefixes. For example, the Gao-Rexford routing policies [20], which provide a baseline of understanding for how network operators configure BGP, prescribe that routes learned from customers are preferred to routes learned from either peers or providers, and that routes learned from customers are exported to all neighbor ASs and all routes learned from neighbor ASs are exported to customers—no mention is made to destination IP prefixes. The same independence between routing policies and destinations occurs with next-hop routing policies [10], policies including sibling-sibling relationships [21], and backup policies [22]. Indeed, with more than 600,000 globally routed IP prefixes and more than 50,000 ASs as of today [23], it would hardly be realistic or conducive to a scalable Internet routing system to configure BGP per destination IP prefix.

We clarify and justify two hypotheses of our routing model. First, we assume a setting that includes anycast routing, whereby a destination is a set of nodes each of which originates a route. Several applications and services in the Internet use anycast routing, such as DNS [24] and some Content Delivery Networks [25]. Anycast routing also models cases where an IP prefix appears to be originated at multiple ASs, for example, because the IP prefix is originated at an exchange point or at a network that multi-homes to several providers without the intervention of BGP [26]. Second, our use of the term correctness implies termination and the absence of forwarding loops in stable state in the network and in all sub-networks obtained after an arbitrary set of link failures. This is a reasonable requirement since link failures are mostly unpredictable, while routing protocols should remain operational despite them.

Our main result thus reduces the correctness of routing vector protocols to a structural property of routing policies in a network. Tests for strict-cycle-absorbency can be incorporated in tools for automatic specification and verification of routing policies [27], [28], endowing the latter with the ability to always tell whether or not input routing configurations lead to correctness. The tools, and the tests for strict-cycle-absorbency, presuppose access to the routing policies of a multitude of nodes. This information may be difficult to gather in some cases, especially in inter-domain routing, given the reluctance of network operators to share the routing configurations of their ASs. Nevertheless, even in the case of inter-domain routing, the tools and tests serve to check presumed routing policies and to assist in designing guidelines for correct configuration of BGP.

Here, we derive two implications of our main result. First, we show that correctness of next-hop routing policies is at odds with resiliency to failures. Correct configuration of next-hop routing policies necessarily excludes some of the paths physically existing in the network from the possibility of carrying data-packets. Second, we present a model of BGP for when the routing policies set through BGP's parameter LOCAL-PREF do not depend on BGP's parameter AS-PATH. With this model, we present inter-domain routing policies for sibling-sibling relationships between ASs that always preserve correctness of BGP.

The remainder of the paper is organized as follows. In the way of motivation, Section II discusses three examples of routing policies that lead to different types of undesirable behaviors. Section III presents the routing model. Section IV states and proves the main result, equating the correctness of routing vector protocols to strictly absorbent cycles. Section V examines classes of routing policies and assesses their impact on strict-cycle-absorbency. Next-hop routing policies are debated in this section. Section VI presents a model of BGP and provides a discussion of inter-domain routing policies with sibling-sibling relationships. Section VII further discusses related work, and Section VIII contains a final summary and appraisal.

II. Correctness by Example

We start with a discussion of three simple classes of routing policies, emphasizing the different types of undesirable behavior each may induce on a routing vector protocol.

Shortest Paths With Steady Lower Bound: The first example concerns shortest-path routing with lengths represented by integers in some finite range that includes negative values.[1] Routing policies are such as to find distances (lengths of shortest paths) in a network: the routing protocol is a standard distance vector protocol. Links are assigned lengths and routes associate destinations to lengths. The length of a route is an estimate of the distance toward the destination. An elected route at a node v is extended to a candidate route at a neighbor node u by adding the length of the link from u to v to the length of the elected route at v, possibly truncated to fit into the range of allowed lengths. Among the candidate routes for the same destination, a node elects the one with the shortest length. It is well-known that a distance vector protocol terminates in a stable state devoid forwarding loops if and only if all cycles in the network have positive length.

We investigate in more detail the behavior of a distance vector protocol when there is a cycle of negative length. In Figure 1, links point in the direction of the flow of data-packets; routes travel in the opposite direction. The integers next to links represent their lengths. Lengths are truncated from below at -100 with the understanding that a route of length -100 can be used to forward data-packets. Cycle $uvwu$ has a negative length of $1 - 4 + 1 = -2$. Let node y be the destination originating a route of length 0. That route extends to a route of length 1 at w which is then propagated counterclockwise around the cycle, reaching again node w via node u as a candidate route of length $-2+1 = -1$. The latter route is elected at w and further propagated counterclockwise around the cycle, arriving back at w as a candidate route of length $-2 - 1 = -3$. Routes of ever more negative lengths

[1]The finite range avoids trivial count-to-infinity problems and is always verified in practice.

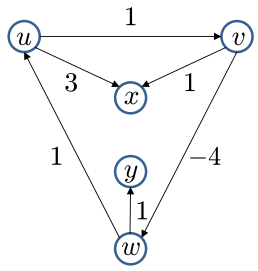

Fig. 1. Links point in the direction of data-packet flow. The length of a link is indicated next to it. Cycle $uvwu$ has a negative length of -2.

continue to be propagated through w around the cycle until a stable state is reached. In the stable state, w elects a route of length -98, learned from u; v elects a route of length -100 learned from w ($-4-98=-102$ which is truncated to -100); and u elects a route of length -99, learned from v. The stable state encloses a forwarding loop around cycle $uvwu$: the forwarding neighbor of w is u; the forwarding neighbor of u is v; and the forwarding neighbor of v is w.

Suppose, instead, that the destination is node x, rather than node y, originating a route of length 0. That route extends to a route of length 1 at v and to a route of length 3 at u. These two routes will propagate several times counterclockwise around the cycle until the distance vector protocol terminates in exactly the same stable state as when the destination was y. Namely, w elects a route of length -98, learned from u; v elects a route of length -100, learned from w; and u elects a route of length -99 learned from v.

In short, for both destinations x and y the distance vector protocol terminates in a stable state that encloses a forwarding loop.

Shortest Paths Without Steady Lower Bound: The routing policies of the second example are similar to those of the first example, only now a node establishes that the destination cannot be reached when it receives a route that, according to standard addition, would yield a candidate route with length less than -100. The condition for termination of a distance vector protocol in a stable state devoid of forwarding loops is the same as before, but the manifestation of undesirable behavior is different.

Consider again Figure 1 and let y be the destination originating a route of length 0. That route extends to a route of length 1 at w which circulates 49 times counterclockwise around the cycle before arriving at w via u as a route of length -98. Node w extends the route received from u into a candidate route of length $1-98=-97$, elects that route, and sends it to v. Node v attempts to compute a candidate route from the route received from w by adding -4 to -97. Since $-4-97=-101<-100$, v establishes that destination y cannot be reached and sends this information to u which propagates it further to w. Unable to reach destination y via u, node w again elects the route of length 1 learned directly from y. The process repeats itself without the distance vector protocol ever reaching a stable state—there is no such state. The same conclusion is valid if the destination were x rather than y originating a route of length 0.

Shortest-Paths With Path-Complements: In the third example, we refine the distance vector routing policies of the first example by including a path-complement in routes, used in the same way as the parameter AS-PATH is used in BGP [1]. In particular, the path-complement provides local loop-detection. Routes associate destinations to couplets containing a length and a path-complement. For instance, assuming that routes are originated with couplet $(0,\varepsilon)$, where ε is the empty string, from the presence of a route with couplet $(10,vwx)$ at node u we deduce that a route was originated at x, propagated through w and v before reaching u, and that the length of path $uvwx$ is 10. An elected route at a node v extends to a candidate route at neighbor node u by adding the length of the link from u to v to the length of the elected route at v and prefixing v to the path-complement of the elected route at v, *except* that if u is in the path-complement of the elected route at v, then the route is discarded at u. For instance, if link uv has length 5, then an elected route with couplet $(10,wx)$ at v extends to a candidate route with couplet $(15,vwx)$ at u; an elected route with couplet $(10,wux)$ at v is discarded at u. Among the candidate routes to the same destination, a node elects the one with the shortest length. If there are multiple routes with that length, then a node elects one with the shortest (fewest nodes in the) path-complement. Further tie-breaks are broken arbitrarily. It is easy to show that if all cycles in the network have positive *or* zero length, then the routing vector protocol terminates in a stable state devoid of forwarding loops.

It is less obvious what happens when there is a cycle of negative length, which is the condition we now explore. Consider Figure 1 in light of the new routing policies. Let y be the destination originating a route with couplet $(0,\varepsilon)$. That route extends to a route with couplet $(1,y)$ at w, which extends to a route with couplet $(-3,wy)$ at v, which extends to a route with couplet $(-2,vwy)$ at u. The latter route is sent from u to w, but, since w belongs to path-complement vwy, the route is discarded at w. A stable state for destination y is reached whereby w elects a route with couplet $(1,y)$, learned from y; v elects a route with couplet $(-3,wy)$, learned from w; and u elects a route with couplet $(-2,vwy)$, learned from v. That stable state does not harbor a forwarding loop. It is shown graphically in Figure 2.

Suppose, instead, that the destination is node x, rather than node y, originating a route with couplet $(0,\varepsilon)$. That route extends to a route with couplet $(1,x)$ at v and to a route with couplet $(3,x)$ at u. Assume that it takes no time for a route to travel from u to w, it takes one unit of time for a route to travel from w to v, and it takes one unit of time for a route to travel from v to u. At time $T=0$, nodes u, v, and w elect routes with couplets $(3,x)$, $(1,x)$, and $(4,ux)$, respectively, as shown on the left-hand side of Figure 3. At time $T=1$, u elects a route with couplet $(2,vx)$, learned from v; w elects a route with couplet $(3,uvx)$, learned from u; and v elects a route with couplet $(0,wux)$, learned from w, as shown on the right-hand side of Figure 3. Route with couplet $(3,uvx)$, elected at w, is sent to v, but is discarded there because v is in path-complement uvx. Route with couplet $(0,wux)$, elected at v, is sent to u, but is, likewise, discarded because u is

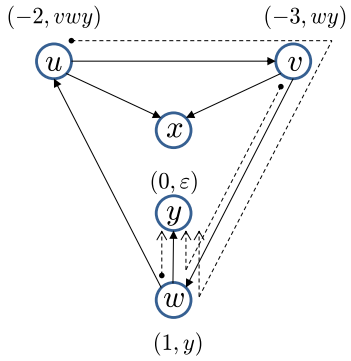

Fig. 2. Destination is y. Couplets of the routes elected by the nodes in the stable state are indicated next to them. As a visualization aid, the dashed paths mark the path-complement of couplets. The stable state does not contain a forwarding loop.

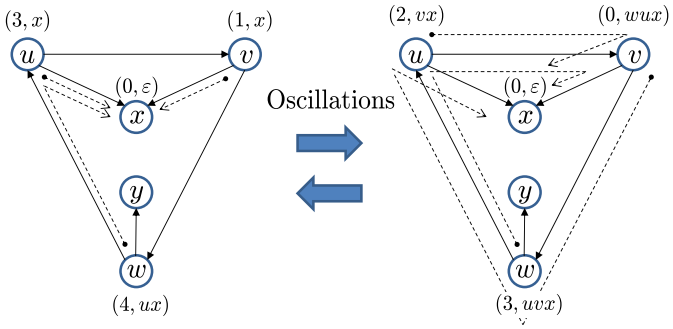

Fig. 3. Destination is x. Couplets of the routes elected by the nodes are indicated next to them. As a visualization aid, the dashed paths mark the path-complement of couplets. The routing vector protocol oscillates indefinitely between the routing state depicted on the left-hand side and the routing state depicted on the right-hand side.

in path-complement wux. Therefore, at time $T = 2$, nodes u, v, and w revert to the election of routes with couplets $(3, x)$, $(1, x)$, and $(4, ux)$, respectively, as shown on the left-hand side of Figure 3. The routing vector protocol oscillates indefinitely.[2]

In summary, for destination y the routing vector protocol terminates in a stable state devoid of forwarding loops whereas for destination w the routing vector protocol does not terminate. Our later results will imply that, for the general case of the distance vector routing policies refined with path-complements, if at least one cycle in the network has negative length, then there will be a destination for which the routing vector protocol does not terminate.

A Note Regarding Destinations: With generality, a destination is understood in an anycast sense. Consider the network consisting solely of the cycle of Figure 1, with x and y removed, and the distance vector routing policies refined with path-complements. In order to show that the routing vector protocol does not terminate, the destination must comprise at least two nodes. For instance, let the destination consist of the two nodes u and v, with u originating a route with couplet $(3, \varepsilon)$ and v originating a route with couplet $(1, \varepsilon)$. Keeping the previous assumptions on the traveling times of routes, at time $T = 0$, u elects the route originated locally with couplet $(3, \varepsilon)$; w elects a route with couplet $(4, u)$, learned from u; and v elects the route originated locally with couplet $(1, \varepsilon)$. At time $T = 1$, u elects a route with couplet $(2, v)$, learned from v; w elects a route with couplet $(3, uv)$, learned from u; and v elects a route with couplet $(0, wu)$, learned from w. At time $T = 2$, local loop-detection takes effect and nodes re-elect the routes they elected at $T = 0$. The routing vector protocol oscillates indefinitely.

[2]The conditions on route travel times that lead to oscillations are broad. As long as u leaks a route with couplet $(3, x)$ into the cycle and v leaks a route with couplet $(1, x)$ into the cycle, the routing vector protocol oscillates.

III. Routing Model

A routing vector protocol is a distributed algorithm to find paths in a network according to routing policies configured at various nodes. Section III-A reviews an algebraic framework that models arbitrary routing policies. Section III-B discusses the operation and correctness of a routing vector protocol.

A. Attributes and Labels

The operation of a routing vector protocol relies on iterations of two elementary processes: election of a route at a node from a set of candidate routes; and extension of an elected route at a node into a candidate route at a neighbor node.

A route is a piece of state information associating a destination to an *attribute* [19]. The set of attributes is denoted by Σ, assumed finite. Attributes are totally ordered by $\preceq$. Given two attributes α and β, $\alpha \succeq \beta$ is the same as $\beta \preceq \alpha$; $\alpha \prec \beta$ is the same as $\alpha \preceq \beta$ and $\alpha \neq \beta$; and $\alpha \succ \beta$ is the same as $\beta \prec \alpha$. If $\alpha \prec \beta$ ($\alpha \succ \beta$), then we say that α is *preferred* to β (α is *less preferred* than β).

From $\prec$, we define an *election* operation $\sqcap$ that yields the most preferred of two attributes:

$$\alpha \sqcap \beta = \begin{cases} \alpha, & \text{if } \alpha \preceq \beta; \\ \beta, & \text{if } \alpha \succeq \beta. \end{cases}$$

The election operation $\sqcap$ is associative, commutative, and selective [29], so that we can talk without ambiguity about the elected attribute from among a set of attributes $\{\alpha_1, \ldots, \alpha_n\}$, which we denote by $\sqcap\{\alpha_1, \ldots, \alpha_n\}$. There is a special attribute $\bullet$ to indicate that a destination is not reachable. Attribute $\bullet$ is the least preferred of all attributes. Attributes other than $\bullet$ are called *proper*. The preference and election of routes is the preference and election, respectively, of their attributes. A route is proper if its attribute is proper.

A link uv in a network represents the possibility of forwarding data-packets from u to v. Routes travel in the opposite direction. The routing policies of link uv tell how the attribute of a route elected at v is *extended* into the attribute of a candidate route stored at u to reach the destination via v. They subsume the export policy of v with regard to u and the import policy of u with regard to v. We assume that routing policies do not depend on the destination. Therefore, the routing policies of link uv are modeled by a map on the set of attributes which we denote by $L[uv]$ and call the *label* of uv. Label $L[uv]$ extends attribute α into attribute $L[uv](\alpha)$. If v cannot reach a destination, then u cannot reach that same destination via v. Therefore, $L[uv](\bullet) = \bullet$. The extension of a route into another is the extension of the attribute of the former route into the attribute of the latter.

More generally, we call label to every map L on Σ such that $L(\bullet) = \bullet$. A walk $u_0u_1 \dots u_n$ in a network is associated with a label, denoted by $L[u_0u_1 \dots u_n]$, that results from the composition of the labels of its constituent links: $L[u_0u_1 \dots u_n] = L[u_0u_1] \cdots L[u_{n-1}u_n]$. A route with attribute α that propagates along walk $u_0u_1 \dots u_n$, from u_n to u_0, becomes a route at u_0 with attribute $L[u_0u_1 \dots u_n](\alpha)$.

B. Operation and Correctness of Routing Vector Protocols

We discuss the operation and correctness of a routing vector protocol in a network starting from an *initial assignment* R^* of attributes to the nodes of the network, with $R^*[t]$ denoting the attribute assigned to node t. The destination of initial assignment R^* is the set D of nodes which have been assigned a proper attribute:

$$D = \{t \mid R^*[t] \prec \bullet\}.$$

If D has multiple nodes, then routing is anycast.

A node t of D originates a route with attribute $R^*[t]$ that it sends to all its neighbors. Every node u stores, for every one of its neighbors, a candidate route to reach D via that neighbor. Whenever u receives a route with attribute α from its neighbor v, it replaces the previous candidate route to reach D via v with a new candidate route having attribute $L[uv](\alpha)$, and then uses operation $\sqcap$ among the attributes of all candidate routes to elect a new route. If the newly elected route is different from the one previously elected, then it is sent to all of u's neighbors. Node v is a forwarding neighbor of u if the elected route at u is the candidate route learned from v. The model admits equal-route multi-path routing, so that any or all of the forwarding neighbors of u can be used to forward data-packets to D.

A *stable state for initial assignment* R^* is a state without routes for D in transit in the network. Such a state is represented by a *final assignment* R of attributes to nodes, with $R[u]$ denoting the attribute of the elected route at u to reach D in the stable state. Without routes in transit from v to u, the last route received at u from v is the route elected at v: the candidate route at u via v is $L[uv](R[v])$. Hence, final assignment R satisfies the following system of fixed-point equations [29]:

$$R[u] = \sqcap\{L[uv](R[v]) \mid v \text{ a neighbor of } u\} \sqcap R^*[u]. \quad (1)$$

A routing vector protocol can be regarded as a distributed algorithm to solve for R given R^* in the above system of fixed-point equations.

A final assignment R contains a forwarding loop around cycle $C = u_0u_1 \cdots u_{n-1}u_0$ if every node of the cycle has the next for forwarding neighbor, that is, if $R[u_0] \prec \bullet$ and $R[u_i] = L[u_iu_{i+1}](R[u_{i+1}])$ for all i, $0 \le i < n$, where subscripts are interpreted modulus n. If a forwarding loop exists around C, then data-packets with destination in D are forever trapped around C.

A routing vector protocol *terminates for initial assignment* R^* if, eventually, it reaches a stable state for R^*, whatever the initial routing state and whatever the travel times of routes across links, in the network and in every sub-network obtained after an arbitrary set of link failures. A routing vector protocol is *correct for initial assignment* R^* if it terminates and no final assignment, in the network or a sub-network, contains a forwarding loop. Last, a routing vector protocol is *correct* if it is correct for all initial assignments.

IV. Relationship Between Cycles and Correctness

The correctness of routing vector protocols is intimately related to how routing policies are configured at the various nodes around the cycles of a network. In Section IV-A, we present the concept of strictly absorbent cycle, which equates to the correctness of a routing vector protocol. In the same section, we also present the slightly broader concept of absorbent cycle, which will play a critical role in our model of BGP. In Section IV-B, we show that if all cycles in a network are strictly absorbent, then a routing vector protocol is correct. In Section IV-C, we show the converse result: if at least one cycle in the network is not strictly absorbent, then a routing vector protocol is not correct.

A. Strictly Absorbent Cycles and Absorbent Cycles

A cycle is *strictly absorbent*[3] if, for every combination of proper routes learned by its nodes externally to the cycle and sent to their neighbors, at least one node prefers the route learned externally to the route learned from the neighbor. In symbols, cycle $C = u_0u_1 \cdots u_{n-1}u_0$ is strictly absorbent if

$$\forall_{\alpha_0 \prec \bullet, \alpha_1 \prec \bullet, \dots, \alpha_{n-1} \prec \bullet} \ \exists_{0 \le i < n} \quad \alpha_i \prec L[u_iu_{i+1}](\alpha_{i+1}), \quad (2)$$

where indexes are interpreted modulus n. In Condition (2), attributes $\alpha_0, \alpha_1, \dots, \alpha_{n-1}$ represent external routes. The preference of the external route over the route learned from the neighbor at node i is expressed by $\alpha_i \prec L[u_iu_{i+1}](\alpha_{i+1})$.

A cycle is *absorbent* if, for every combination of routes learned by its nodes externally to the cycle and sent to their neighbors, at least one node prefers the route learned externally to the route learned from the neighbor *or*, for all nodes, the route learned externally equals the route learned from the neighbor. In symbols, cycle $C = u_0u_1 \cdots u_{n-1}u_0$ is absorbent if

$$\begin{aligned} \forall_{\alpha_0, \alpha_1, \dots, \alpha_{n-1}} \quad & \exists_{0 \le i < n} \ \alpha_i \prec L[u_iu_{i+1}](\alpha_{i+1}) \\ \vee \ & \forall_{0 \le i < n} \ \alpha_i = L[u_iu_{i+1}](\alpha_{i+1}). \end{aligned} \quad (3)$$

Every strictly absorbent cycle is absorbent.

B. Strictly Absorbent Cycles Imply Correctness

In a strictly absorbent cycle, the preference for the external route at some node being assured for every combination of external routes suggests that neither permanent route oscillations nor forwarding loops ever form around the cycle. Indeed, we have the following theorem.

Theorem 1: If all cycles in the network are strictly absorbent, then the routing vector protocol is correct.

[3] In [19], a strictly absorbent cycle is called "free."

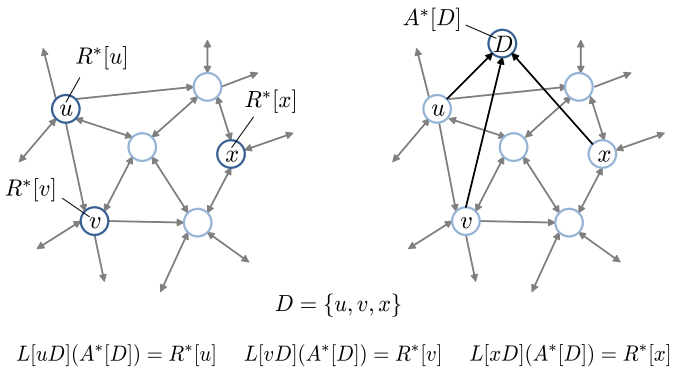

Fig. 4. A routing vector protocol terminates for initial assignment R^* in the network to the left if and only if it terminates for initial assignment A^* in the augmented network to the right. The destination of initial assignment R^* comprises nodes u, v, and x, whereas the destination of initial assignment A^* comprises only node D.

Proof (sketch): Let R^* be any initial assignment of attributes to nodes in a network where all cycles are strictly absorbent. The absence of a forwarding loop in a stable state derives directly from the definition of strict-cycle-absorbency. Indeed, let R be a final assignment of elected attributes in a stable state. A forwarding loop around cycle $C = u_0u_1 \cdots u_{n-1}u_0$ would mean that there were attributes $R[u_i]$, with $R[u_0] \prec \bullet$, such that $R[u_i] = L[u_iu_{i+1}](R[u_{i+1}])$, for $0 \leq i < n$. Attributes $R[u_i]$ invalidate the strict absorbency of C.

In order to show termination for initial assignment R^*, we rely heavily on [19, Th. 1]. That theorem implies that given a network where all cycles are strictly absorbent and a destination in that network *comprising a single node*, a routing vector protocol with path-complements as part of their attributes terminates. In the proof of that theorem, the purpose of path-complements is only to make the set of possible attributes finite and, thus, prevent count-to-infinity problems. In our case, the set of attributes is finite by hypothesis.

Thus, the only aspect left to be proven is that the routing vector protocol terminates even for an initial assignment R^* with a destination D that *comprises multiple nodes*. Indeed, we can equate termination in the network with initial assignment R^* to termination in an augmented network with an initial assignment A^* having a single node for destination. The augmented network has a new node $\{D\}$ and new links tD, for every t in D. Initial assignment A^* satisfies $A^*[D] \prec \bullet$ and $A^*[u] = \bullet$ for every u in the network. Labels of links tD are such that $R^*[t] = L[tD](A^*[D])$ for every t in D. Figure 4 illustrates the relationship between a network and its augmented network. The added node and links do not create a cycle. Hence, the routing vector protocol terminates for initial assignment R^* in the network if and only if it terminates for initial assignment A^* in the augmented network. ■

C. Correctness Implies Strictly Absorbent Cycles

The definition of strictly absorbent cycle, Condition (2), compares routes to other routes that traverse one single link of a cycle. In order to show undesirable routing behaviors when cycles are not strictly absorbent, we need to be able to compare routes to other routes that propagate several hops around a cycle. We first establish a notation to deal with the propagation of such routes. In relation to cycle $C = v_0v_1 \cdots v_{n-1}v_0$, we let, for $0 \leq i, j < n$,

$$v_iCv_j = v_iv_{i+1} \cdots v_j,$$

with indexes interpreted modulus n. If $i \neq j$, then v_iCv_j is a path from v_i to v_j along C; if $i = j$, then v_iCv_j is a circuit around C starting and ending at node v_i.[4] Recall from Section III that the label of v_iCv_j, denoted by $L[v_iCv_j]$, is defined as the composition of the labels of the links of v_iCv_j:

$$L[v_iCv_j] = L[v_iv_{i+1}] \cdots L[v_{j-1}v_j],$$

with indexes interpreted modulus n. A route with attribute α expedited by v_j around the cycle becomes a candidate route with attribute $L[v_iCv_j](\alpha)$ at v_i.

We say that the m $(1 \leq m \leq n)$ nodes $u_0, u_1, \ldots, u_{m-1}$ are disposed around C if the u_jCu_{j+1}, for $0 \leq j < m$ with indexes interpreted modulus m, are pairwise link-disjoint. Informally, nodes $u_0, u_1, \ldots, u_{m-1}$ are disposed around C if on traveling around the cycle starting at u_0 we visit the other nodes in the order $u_1, u_2 \ldots, u_{m-1}$ before returning to u_0.

The propagation of routes several hops around cycle C is well described by the following n conditions, each of which may or may not be true of C:

Condition $\mathbf{T}(m)$, for $1 \leq m \leq n$: There are m nodes $u_0, u_1, \ldots, u_{m-1}$ disposed around C and m proper attributes $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ such that

$$\forall_{0 \leq i < m} \quad \alpha_i \succeq L[u_iCu_{i+1}](\alpha_{i+1}).$$

In Condition $\mathbf{T}(m)$, the inequality $\alpha_i \succeq L[u_iCu_{i+1}](\alpha_{i+1})$ means that when the external route with attribute α_{i+1} is propagated from u_{i+1} to u_i, it becomes equal or preferred to the external route at u_i. For m the number of nodes of C, $m = n$, we have that u_iCu_{i+1} is a link of C, that is, $u_iCu_{i+1} = v_jv_{j+1}$, for some index j. Therefore, $\mathbf{T}(n)$ is the statement that cycle C is not strictly absorbent.

Consider the next two conditions on C:

Condition S: There is a node u of C and a proper attribute α such that

$$\alpha \succeq L[uCu](\alpha).$$

Condition M: For some integer m greater than one, there are m nodes $u_0, u_1, \ldots, u_{m-1}$ disposed around C and m attributes $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ such that

$$\forall_{0 \leq i < m} \quad \alpha_{i+1} \succ L[u_{i+1}Cu_{i+2}](\alpha_{i+2}) \\ \wedge\ \alpha_i \prec L[u_iCu_{i+2}](\alpha_{i+2}).$$

In Condition **S**, the inequality $\alpha \succeq L[uCu](\alpha)$ means that a route with attribute α expedited by u around C arrives back at u as a candidate route with the same or a preferred attribute. In Condition **M**, the two inequalities $\alpha_{i+1} \succ$

[4] As is usually the case, the string $v_0v_1 \cdots v_{n-1}v_0$ denotes both a cycle and a circuit around that cycle starting and ending at v_0. The context makes clear which is the case.

$L[u_{i+1}Cu_{i+2}](\alpha_{i+2})$ and $\alpha_i \prec L[u_iCu_{i+2}](\alpha_{i+2})$ mean that when the external route with attribute α_{i+2} is propagated from u_{i+2} to u_{i+1}, it becomes preferred to the external route there, $\alpha_{i+1} \succ L[u_{i+1}Cu_{i+2}](\alpha_{i+2})$, but, when this preferred route at u_{i+1} is propagated further to u_i, it becomes *less* preferred than the external route there, $\alpha_i \prec L[u_iCu_{i+2}](\alpha_{i+2})$.

In the next theorem, we show that if C is not strictly absorbent, then, by taking the smallest integer for which $\mathbf{T}(m)$ is valid, C satisfies either Condition **S** or Condition **M**. Then, we show that each of these two conditions drives a routing vector protocol into undesirable behaviors.

Theorem 2: If cycle $C = v_0v_1\cdots v_{n-1}v_0$ is not strictly absorbent, then it satisfies Condition **S** or Condition **M** (or both).

Proof: Let m^* be the smallest integer for which Condition $\mathbf{T}(m)$ is satisfied, and let $u_0, u_1, \ldots, u_{m^*-1}$ be m^* nodes disposed around C and $\alpha_0, \alpha_1, \ldots, \alpha_{m^*-1}$ be m^* proper attributes that together validate $\mathbf{T}(m^*)$:

$$\forall_{0\leq i<m^*} \quad \alpha_i \succeq L[u_iCu_{i+1}](\alpha_{i+1}). \tag{4}$$

Condition $\mathbf{T}(1)$ is exactly Condition **S**. Thus, if $m^* = 1$, then the theorem is proven.

Suppose, instead, that $m^* \geq 2$. We show that Condition **M** is validated with the m^* nodes $u_0, u_1, \ldots, u_{m^*-1}$ disposed around C and the m^* attributes $\alpha_0, \alpha_1, \ldots, \alpha_{m^*-1}$. In order to obtain a contradiction, assume otherwise, that is, assume that

$$\begin{aligned}\exists_{0\leq i<m^*} \ & \alpha_{i+1} \preceq L[u_{i+1}Cu_{i+2}](\alpha_{i+2}) \\ & \vee\ \alpha_i \succeq L[u_iCu_{i+2}](\alpha_{i+2}).\end{aligned} \tag{5}$$

Let i^* be the index identified by the existential quantifier:

$$\begin{aligned}&\alpha_{i^*+1} \preceq L[u_{i^*+1}Cu_{i^*+2}](\alpha_{i^*+2}) \\ &\vee\ \alpha_{i^*} \succeq L[u_{i^*}Cu_{i^*+2}](\alpha_{i^*+2}).\end{aligned} \tag{6}$$

If the first disjunct above is satisfied, $\alpha_{i^*+1} \preceq L[u_{i^*+1}Cu_{i^*+2}](\alpha_{i^*+2})$, then, combining with (4), we deduce that

$$\alpha_{i^*+1} = L[u_{i^*+1}Cu_{i^*+2}](\alpha_{i^*+2}). \tag{7}$$

Therefore, we write

$$\begin{aligned}\alpha_{i^*} &\succeq L[u_{i^*}Cu_{i^*+1}](\alpha_{i^*+1}) && \text{(from (4))} \\ &= L[u_{i^*}Cu_{i^*+1}](L[u_{i^*+1}Cu_{i^*+2}](\alpha_{i^*+2})) && \text{(from (7))} \\ &= L[u_{i^*}Cu_{i^*+2}](\alpha_{i^*+2}).\end{aligned}$$

The set of nodes $u_0, u_1, \ldots, u_{i^*}, u_{i^*+2}, \ldots, u_{m^*-1}$ and the set of attributes $\alpha_0, \alpha_1, \ldots, \alpha_{i^*}, \alpha_{i^*+2}, \ldots \alpha_{m^*-1}$ validate $\mathbf{T}(m^*-1)$, contradicting the choice of m^* as the smallest integer for which Condition $\mathbf{T}(m)$ is satisfied.

If the second disjunct in (6) is satisfied, $\alpha_{i^*} \succeq L[u_{i^*}Cu_{i^*+2}](\alpha_{i^*+2})$, then the set of nodes $u_0, u_1, \ldots, u_{i^*}, u_{i^*+2}, \ldots, u_{m^*-1}$ and the set of attributes $\alpha_0, \alpha_1, \ldots, \alpha_{i^*}, \alpha_{i^*+2}, \ldots \alpha_{m^*-1}$, validate $\mathbf{T}(m^*-1)$, again contradicting the choice of m^* as the smallest integer for which Condition $\mathbf{T}(m)$ is satisfied.

In conclusion, Condition **M** is validated with the m^* nodes $u_0, u_1, \ldots, u_{m^*-1}$ disposed around C and the m^* attributes $\alpha_0, \alpha_1, \ldots, \alpha_{m^*-1}$. ■

We separate the proofs of incorrectness under Condition **S** and under Condition **M**, as they correspond to distinct types of undesirable behaviors.

Theorem 3: If a network has a cycle that satisfies condition **S**, then a routing vector protocol is not correct in that network.

Proof: Since the only thing we know about the network is that it contains a cycle C that satisfies condition **S**, we consider the sub-network consisting of C alone. There is a node u and a proper attribute α such that $\alpha \succeq L[uCu](\alpha)$. We subdivide the proof into two cases. In the first case, there is an attribute β such that $\beta \preceq \alpha$ and $\beta = L[uCu](\beta)$. The state whereby u elects a route with attribute β and every node x of the cycle other than u elects a route with attribute $L[xCu](\beta)$ is stable and encloses a forwarding loop (see the first example of Section II).

In the second case, for all attributes $\beta \preceq \alpha$, we have $\beta \neq L[uCu](\beta)$. In particular, $\alpha \succ L[uCu](\alpha)$. Consider the initial assignment that maps α to u. That is, the destination is composed of u alone originating a route with attribute α. No stable state exists implying that the routing vector protocol oscillates forever (see the second example of Section II). ■

Theorem 4: If a network has a cycle that satisfies condition **M**, then a routing vector protocol is not correct in that network.

Proof: Again, since the only thing we know about the network is that it contains a cycle C that satisfies condition **M**, we consider the sub-network consisting of C alone. Let $u_0, u_1, \ldots, u_{m-1}$ be the m nodes, $m \geq 2$, disposed around C and $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ be the m attributes that validate Condition **M**.

Consider the initial assignment that maps α_j to u_j, $0 \leq j < m$. The destination is the set of nodes $\{u_0, u_1, \ldots, u_{m-1}\}$ with u_j originating a route with attribute α_j. Suppose that it takes one unit of time for a route to propagate from u_{j+1} to u_j around C. Initially, u_j elects the route with attribute α_j originated locally. After one unit of time has elapsed, u_j elects route $L[u_jCu_{j+1}](\alpha_{j+1})$, learned from around the cycle, since $\alpha_j \succ L[u_jCu_{j+1}](\alpha_{j+1})$. After one more unit of time elapses, u_j learns candidate route $L[u_jCu_{j+2}](\alpha_{j+2})$ which propagated from u_{j+2} through u_{j+1} to u_j. Since $\alpha_j \prec L[u_jCu_{j+2}](\alpha_{j+2})$, node u_j reverts its election to the route with attribute α_j originated locally. The routing vector protocol oscillates forever. These events are illustrated in Figure 5 for a cycle with six nodes and $m = 3$ (see the third example of Section II). ■

The next theorem combines the previous three theorems into the converse of Theorem 1.

Theorem 5: If the routing vector protocol is correct, then all cycles in the network are strictly absorbent.

Proof: We prove the contrapositive statement. If the network has at least one cycle that is not strictly absorbent, then, by Theorem 2, that cycle satisfies either Condition **S** or Condition **M**. If it satisfies Condition **S**, then, by Theorem 3, the routing vector protocol is not correct; if it satisfies Condition **M**, then, by Theorem 4, the routing vector protocol is also not correct. ■

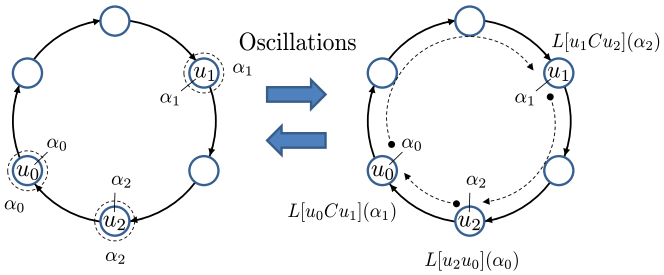

Fig. 5. Nodes u_0, u_1, and u_2, and attributes α_0, α_1, and α_2 assert Condition **M**. A routing vector protocol oscillates when u_0, u_1, and u_2 originate routes α_0, α_1, and α_2, respectively. Attributes of elected routes are shown next to nodes, outside of the circle. A dashed circle indicates election of the originated route (left-hand side). A dashed arrow points to the origin of the route that is currently elected (right-hand side).

V. Link Properties and Correctness

Routing policies are typically configured from a small set of parameters and rules in a way that must be intelligible to a network operator. Thus, it is not surprising that many routing policies used in practice end up having some structure which translates into algebraic properties of the labels of links. We present three such algebraic properties and relate them to strict-cycle-absorbency. In Section V-A, we review absorbency as a property of links, and in Section V-B, we discuss isotonicity. These two properties will be used later in the analysis of concrete routing policies for inter-domain routing. In Section V-C, we present next-hop routing policies and show why they constrain the usability of a network.

A. Absorbency

A label is *absorbent* if it maps every attribute either into itself or into a less preferred attribute [19]. In symbols, label L is absorbent if

$$\forall_\alpha \quad \alpha \preceq L(\alpha). \tag{8}$$

A label is *strictly absorbent* if it maps every proper attribute into a less preferred attribute. In symbols, label L is strictly absorbent if

$$\forall_{\alpha \prec \bullet} \quad \alpha \prec L(\alpha). \tag{9}$$

A link is (strictly) absorbent if its label is (strictly) absorbent. Clearly, if all links of a cycle are absorbent, then the cycle is absorbent. The following theorem is easily derived from [19, Th. 2].

Theorem 6: Let $C = u_0u_1 \cdots u_{n-1}u_0$ be a cycle all links of which are absorbent. Then, C is strictly absorbent if and only if for every proper attribute, there is a link whose label maps that attribute into a less preferred one. In symbols, C is strictly absorbent if and only if

$$\forall_{\alpha \prec \bullet} \exists_{0 \le i < n} \quad \alpha \prec L[u_iu_{i+1}](\alpha).$$

In particular, if one link of the cycle is strictly absorbent, then the cycle is strictly absorbent.

Strict-*link*-absorbency is the sufficient condition for correctness of routing vector protocols most invoked by researchers [7], [11], [28], [30]. However, it fails to represent many routing policies used in practice (see Section VI) and it unnecessarily constrains the design of routing vector protocols. In addition, absorbency is a somewhat artificial property for links in that it compares two routes, the original one and its extension, held at distinct nodes. In the operation of routing vector protocols, routes are compared at nodes, not across them.

B. Isotonicity

A label is *isotone*[5] if it does not invert the preference between any two attributes [19], [29], [31]. In other words, a label is isotone if it is an increasing map on the ordered set of attributes. In symbols, label L is isotone if

$$\forall_{\alpha,\beta} \quad \alpha \preceq \beta \Rightarrow L(\alpha) \preceq L(\beta). \tag{10}$$

A link is isotone if its label is isotone. The following theorem is easily obtained from [19, Th. 4].

Theorem 7: Let $C = u_0u_1 \cdots u_{n-1}u_0$ be a cycle all links of which are isotone. Then, C is strictly absorbent if and only if the label of the circuit around C starting and ending at an arbitrary node u_i is strictly absorbent. In symbols, C is strictly absorbent if and only if:

$$\forall_{\alpha \prec \bullet} \quad \alpha \prec L[u_iCu_i](\alpha).$$

The next theorem is proven in the appendix.

Theorem 8: Let $C = u_0u_1 \cdots u_{n-1}u_0$ be a cycle all links of which are isotone. Then, C is absorbent if and only if the labels of all circuits around C are absorbent. In symbols, C is absorbent if and only if:

$$\forall_{0 \le i < n} \forall_\alpha \quad \alpha \preceq L[u_iCu_i](\alpha).$$

Theorems 7 and 8 significantly reduce the complexity of testing a cycle for strict absorbency and absorbency, respectively, to a verification on the labels of the circuits around the cycle.

C. Next-Hop

With next-hop routing policies, the only routing information that a node learns from a neighbor node is whether or not a destination is reachable [9]–[11]: the node has no information about other properties of the route elected at the neighbor. We formulate next-hop routing policies in algebraic terms, show that these policies constrain the usability of a network, and provide a brief discussion of their deployment in current networks. Consider a link uv with next-hop routing policies. When u learns from v that the destination is reachable, it elects a route with an attribute which may depend on v, but not on the attribute of the route elected at v. Therefore, given attributes α and β of elected routes at v, if $L[uv](\alpha) \prec \bullet$ and $L[uv](\beta) \prec \bullet$, then we must have $L[uv](\alpha) = L[uv](\beta)$. In general, label L is next-hop if

$$\forall_{\alpha,\beta} \quad L(\alpha) \prec \bullet \wedge L(\beta) \prec \bullet \Rightarrow L(\alpha) = L(\beta). \tag{11}$$

A link is next-hop if its label is next-hop.

The next theorem gives an implication of strict-cycle-absorbency for when routing policies are next-hop.

Theorem 9: Let $C = u_0u_1 \cdots u_{n-1}u_0$ be a cycle all links of which are next-hop. If C is strictly absorbent, then there

[5] Several authors use the term monotonicity rather than isotonicity.

is a node u_i of C that does not propagate any proper route around C, from its neighbor u_{i+1} to its neighbor u_{i-1}. In symbols, if C is strictly absorbent, then

$$\exists_{0\leq i<n}\forall_{\alpha} \quad L[u_{i-1}u_iu_{i+1}](\alpha) = \bullet,$$

where indexes are interpreted modulus n.

Proof: We prove the contrapositive statement. Assume that for every i, $0 \leq i < n$, there is a proper attribute α_{i+1} such that u_i propagates a route with attribute α_{i+1} from u_{i+1} to u_{i-1}:

$$L[u_{i-1}u_iu_{i+1}](\alpha_{i+1}) \prec \bullet, \tag{12}$$

with indexes interpreted modulus n. Inequality (12) implies that $L[u_iu_{i+1}](\alpha_{i+1}) \prec \bullet$. Define $\gamma_i = L[u_iu_{i+1}](\alpha_{i+1})$ for every i, $0 \leq i < n$.

Inequality (12) also implies

$$\begin{aligned} L[u_{i-1}u_i](\gamma_i) &= L[u_{i-1}u_i](L[u_iu_{i+1}](\alpha_{i+1})) \\ &= L[u_{i-1}u_iu_{i+1}](\alpha_{i+1}) \prec \bullet. \end{aligned}$$

Since $L[u_{i-1}u_i](\alpha_i) \prec \bullet$, $L[u_{i-1}u_i](\gamma_i) \prec \bullet$, and $u_{i-1}u_i$ is next-hop, we have

$$L[u_{i-1}u_i](\gamma_i) = L[u_{i-1}u_i](\alpha_i) = \gamma_{i-1},$$

for every i, $0 \leq i < n$. Therefore, attributes $\gamma_0, \gamma_1, \ldots, \gamma_{n-1}$ assert that C is not strictly absorbent. ■

We now see how *all* next-hop routing policies constrain the paths that can be used in a network to transport data-packets. From Theorem 5, correctness requires all cycles to be strictly absorbent. From Theorem 9, there is one node in every strictly absorbency cycle that does not propagate *any* proper route around the cycle. Let C be a strictly absorbent cycle with at least three nodes. If node v of C does not propagate any proper route from its neighbor w to its neighbor u, then path uvw—and any path containing uvw—is never discovered by the routing vector protocol and, thus, it can never be used for forwarding data-packets, not even after link failures in other parts of the network. Reference [32] provides a polynomial-time algorithm to quantify how much of the physical connectivity of a network is lost for any fixed, but arbitrary, next-hop routing policies, showing that the losses can be significant.

There is intuition behind the reduced usability of a network operated with next-hop routing policies. For correctness, every proper route that tours around a cycle must either be filtered somewhere along the cycle or arrive back at the origin node with a proper attribute that is less preferred than the one it started out with. Next-hop routing policies exclude the second possibility, as they do not allow the attribute of a route to build up to a less preferred proper attribute when it arrives back at the origin node.

Next-hop routing policies are common in inter-domain routing, because they confer secrecy to the routing configuration of an AS, in that the only routing information conveyed from the AS to a neighbor AS is the set of destinations that it can reach—which is minimal information required for communication. The popular Gao-Rexford routing policies [20] are a particular case of next-hop routing policies (see Section VI-C). The structural constraint accompanying them, stating that there should not exist a cyclic dependency of ASs where each AS is a customer of the next in that dependency, is exactly the statement that all cycles should be strictly absorbent, and it constrains the paths that can used in the Internet for carrying data traffic [33].

Next-hop routing policies are the class of policies that one obtains when interconnecting routing instances at border routers through configuration of parameter Administrative Distance (AD) and of Route Redistribution (RR), a form of interconnection that has been comprehensively studied in a number of papers by Le, Xie, and other co-authors [7], [8], [34]. Each routing instance that a border router is attached to has an AD value that establishes a preference for candidate routes learned from that routing instance. Routing instances with smaller AD values are preferred. Parameter AD sidesteps the difficult problem of comparing and transforming attributes of routes belonging to distinct routing instances. RR designates which routes are exported from one routing instance to another. Reference [34] proposes strict-*link*-absorbency as a condition for correct configuration of interconnected routing instances. Our conclusions in this paper show that the condition for correctness can be widened to strict-*cycle*-absorbency. The resiliency of a network composed of interconnected routing instances improves if parameter AD and RR can be configured per destination IP prefix. However, Reference [35] shows, by example, that, even in this case, not all paths in the network can be used for carrying data traffic.

VI. Application to BGP and Inter-Domain Routing

A route in BGP is associated with a number of parameters, of which we consider two: LOCAL-PREF and AS-PATH. LOCAL-PREF indicates a level of preference assigned by a node to a route learned from a neighbor. Given two candidate routes at a node, learned from different neighbors, the one with largest value of LOCAL-PREF is preferred. AS-PATH is the sequence of nodes traversed by a route from where it was originated up to the node currently holding it. AS-PATH avoids looping routes and it implements shortest-path tie-breaking for candidate routes with the same value of LOCAL-PREF. We present a model of BGP that decouples the routing policies realizable with LOCAL-PREF from the far more restricted routing policies realizable with AS-PATH and show that, according to this model, BGP is correct if and only if all cycles are absorbent in terms of the LOCAL-PREF routing policies. Section VI-A provides an algebraic model for the AS-PATH routing policies. Section VI-B expounds our model of BGP and derives its correctness condition. The model is used in Section VI-C to investigate the impact routing policies for siblings have in the correctness of inter-domain routing.

A. AS-PATH Routing Policies

Nodes have unique identifiers taken from some finite set. The default routing decisions involving AS-PATH at a node consist in: (i) invalidating a route that already references the node; (ii) preferring routes with smaller numbers of identifiers; and (iii) adding the node's identifier to the elected route before sending it to neighbors. These decisions have been exploited in

two primitive types of routing policies: source-poisoning and node prepending. With source-poisoning [36], a destination originates a route containing a sequence of identifiers, thereby ensuring that none of the nodes identified in the sequence are ever present in a path to the destination. With node prepending [37], a node sends an elected route to a neighbor with its own identifier inserted multiple times, thereby inflating the length of the path announced in the route and discouraging the neighbor from electing it.

The algebraic model for the AS-PATH routing policies consists of P-attributes and P-labels. The set Σ^P of P-attributes comprises strings of nodes where a node may appear more than once in a string only in consecutive positions, together with $\bullet$. For example, if u, v, and x are nodes, then $uuvvvx$ and $uvxxx$ are P-attributes whereas $uvxxxv$ is not (v appears in non-consecutive positions). Given two strings ϕ and ψ, ϕ is preferred to ψ if the length of ϕ is shorter than that of ψ, $|\phi| < |\psi|$.

The label of link uv is of the form $\langle uv^l \rangle$, where l is an integer determining the amount of node pre-pending. The case $l = 1$ corresponds to the operation without pre-pending. We have

$$\langle uv^l \rangle(\phi) = \begin{cases} v^l\phi, & \text{if } u \text{ is not in } \phi; \\ \bullet, & \text{otherwise.} \end{cases} \tag{13}$$

All links are strictly absorbent.

B. Model and Correctness of BGP

We consider the case where the routing policies implemented with LOCAL-PREF can be decoupled from AS-PATH. Concretely, this means that the value of LOCAL-PREF of a candidate route at a node can be regarded as a function of the value of LOCAL-PREF of the route elected at the neighbor wherefrom the candidate route was learned. LOCAL-PREF may be a constant for a given neighbor, may be set through BGP-communities, or even through AS-PATH. What is excluded from our assumption is two routes with the same value of LOCAL-PREF at a neighbor giving rise to two candidate routes with different values of LOCAL-PREF at the node.

The routing policies realized with LOCAL-PREF alone are modeled by L-attributes and L-labels. The set of L-attributes is denoted by Σ^L and their order by $\preceq^L$. Proper BGP-attributes (our use of the term) are pairs composed of a proper L-attribute and a proper P-attribute:

$$\Sigma = \underbrace{(\Sigma^L - \{\bullet\})}_{\text{LOCAL-PREF}} \times \underbrace{(\Sigma^P - \{\bullet\})}_{\text{AS-PATH}} \cup \{\bullet\},$$

with lexicographic ordering among them. For all proper BGP-attributes (α, ϕ) and (β, ψ),

$$(\alpha, \phi) \prec (\beta, \psi), \quad \text{if } \alpha \prec^L \beta, \text{ or } \alpha = \beta \text{ and } |\phi| < |\psi|. \tag{14}$$

The BGP-label of link uv is a pair composed of an L-label, $L^L[uv]$, acting on L-attributes and a P-label, $\langle uv^l \rangle$, acting on P-attributes, except that we only obtain a proper BGP-attribute if both the L-attribute and the P-attribute are themselves proper:

$$(L^L[uv], \langle uv^l \rangle)(\alpha, \phi) = (L^L[uv](\alpha), v^l\phi), \tag{15}$$

if $L^L[uv](\alpha) \prec \bullet$ and u is not in ϕ; otherwise,

$$(L^L[uv], \langle uv^l \rangle)(\alpha, \phi) = \bullet. \tag{16}$$

Because AS-PATH routing policies implement shortest-path tie-breaking on LOCAL-PREF routing policies, we expect (non-strict) absorbency of a cycle according to LOCAL-PREF routing policies to be sufficient for strict absorbency of the cycle according to the combined routing policies. The following theorem certifies this expectation.

Theorem 10: A cycle is strictly absorbent according to BGP-attributes and BGP-labels if and only if it is absorbent according to L-attributes and L-labels.

Proof: Let $C = u_0u_1 \cdots u_{n-1}u_0$ be a cycle with n nodes and let l_i be the number of times node u_i is pre-pended in a route sent by u_i to u_{i-1}, $0 \leq i < n$, with indexes interpreted modulus n. Let $(\alpha_0, \phi_0), (\alpha_1, \phi_1), \ldots, (\alpha_{n-1}, \phi_{n-1})$ be proper BGP-attributes. Because C is absorbent according to L-attributes and L-labels, we have

$$\begin{array}{l} \exists_{0 \leq i < n} \quad \alpha_i \prec^L L^L[u_iu_{i+1}](\alpha_{i+1}) \\ \vee \ \forall_{0 \leq i < n} \ \alpha_i = L^L[u_iu_{i+1}](\alpha_{i+1}) \end{array} \tag{17}$$

We need to show that there is an index i such that

$$(\alpha_i, \phi_i) \prec (L^L[u_iu_{i+1}], \langle u_iu_{i+1}^{l_{i+1}} \rangle)(\alpha_{i+1}, \phi_{i+1}). \tag{18}$$

Clearly, if there is an index i such that u_i is in ϕ_{i+1}, then, from (16),

$$(\alpha_i, \phi_i) \prec \bullet = (L^L[u_iu_{i+1}], \langle u_iu_{i+1}^{l_{i+1}} \rangle)(\alpha_{i+1}, \phi_{i+1}).$$

So, assume that u_i is not in ϕ_{i+1}, for all i. From (17), either: (i) there is an index i such that $\alpha_i \prec^L L^L[u_iu_{i+1}](\alpha_{i+1})$; or (ii) for all i, we have $\alpha_i = L^L[u_iu_{i+1}](\alpha_{i+1})$. From (14), case (i) readily implies

$$\begin{aligned} (\alpha_i, \phi_i) &\prec (L^L[u_iu_{i+1}](\alpha_{i+1}), u_{i+1}^{l_{i+1}}\phi_{i+1}) \\ &= (L^L[u_iu_{i+1}], \langle u_iu_{i+1}^{l_{i+1}} \rangle)(\alpha_{i+1}, \phi_{i+1}). \end{aligned}$$

Regarding case (ii), if there were no index i satisfying (18), then, from (14), we would have

$$|\phi_i| \geq |u_{i+1}^{l_{i+1}}| + |\phi_{i+1}| = l_{i+1} + |\phi_{i+1}|,$$

for all i, $0 \leq i < n$, with indexes interpreted modulus n. Adding the n inequalities above leads to the contradiction $0 \geq l_0 + \cdots + l_{n-1}$. Combining cases (i) and (ii), we conclude that there is always an index i satisfying (18), so that C is strictly absorbent.

For the converse statement, suppose that C is not absorbent according to L-attributes and L-labels. Then, there are proper L-attributes $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ such that either $L^L[u_iu_{i+1}](\alpha_{i+1}) \prec^L \alpha_i$ or $L^L[u_iu_{i+1}](\alpha_{i+1}) = \alpha_i$ for all i, with the inequality holding for at least one index. Any P-attributes $\phi_0, \phi_1, \ldots, \phi_{n-1}$ such that u_i is not in ϕ_{i+1} and $l_{i+1} + |\phi_{i+1}| < |\phi_i|$ whenever $L^L[u_iu_{i+1}](\alpha_{i+1}) = \alpha_i$ assert that C is not strictly absorbent. For example, if $L^L[u_{n-1}u_0](\alpha_0) \prec^L \alpha_{n-1}$, then we may take $\phi_{n-1} = \epsilon$ and $\phi_i = u_i^{l_{i+1}+\cdots+l_{n-1}+n-i-1}$, for $0 \leq i < n-1$, where ϵ is the empty string. ∎

Combining Theorems 1, 5, and 10, we obtain the condition for correctness of BGP.

Theorem 11: Whenever the routing policies implemented with LOCAL-PREF are decoupled from those implemented with AS-PATH, BGP is correct if and only if all cycles are absorbent according to the former routing policies.

C. Customers, Providers, Peers, and Siblings

Any two neighbor ASs of the Internet hold an economic relationship between them that, with some simplification, is classified in one of three types: customer-provider; peer-peer; and sibling-sibling. A customer pays to a provider to transit traffic with the rest of the Internet. Two peers exchange traffic between themselves and their respective customers without monetary compensations. Two siblings provide mutual transit without monetary compensations. These relationships materialize into routing policies implemented in BGP with recourse to parameter LOCAL-PREF. Gao and Rexford [20] identified typical routing policies governing customers, providers, and peers. According to these policies, routes learned from customers are preferred to routes learned from peers and the latter are preferred to routes learned from providers.[6] Routes learned from a customer are exported to all neighbor ASs, all routes learned from neighbor ASs are exported to a customer, and these are the only exportations allowed. Liao *et al.* [21] expand on these policies to accommodate siblings. All routes learned from neighbor ASs are exported to a sibling. A route that is exported to a sibling keeps the quality of having been learned from the outside from a customer, a peer, or a provider, as the case may be, but its preference is decreased.

We model these routing policies algebraically. L-attributes are of the form (x, n) standing for 'learned from a x through n siblings', where x is 'customer', 'peer', or 'provider', synthetically represented by the letters c, r, and p, respectively, and n is a nonnegative integer. For example, a route learned from a customer has L-attribute $(c, 0)$ and a route learned from a sibling which learned it from a customer has L-attribute $(c, 1)$. We have $(c, n) \prec^L (r, m) \prec^L (p, l)$, for all n, m, and l, and $(x, n) \prec^L (x, m)$, for all $x \in \{c, r, p\}$ and $n < m$. A customer link joins a provider to a customer; its L-label is denoted by $\mathcal{C}$. A peer link joins a peer to a peer; its L-label is denoted by $\mathcal{R}$. A provider link joins a customer to a provider; its L-label is denoted by $\mathcal{P}$. And a sibling link joins a sibling to a sibling; its L-label is denoted by $\mathcal{S}$. L-labels map L-attributes according to the following chart.

	(c, n)	(r, n)	(p, n)
$\mathcal{C}$	$(c, 0)$	$\bullet$	$\bullet$
$\mathcal{R}$	$(r, 0)$	$\bullet$	$\bullet$
$\mathcal{P}$	$(p, 0)$	$(p, 0)$	$(p, 0)$
$\mathcal{S}$	$(c, n+1)$	$(r, n+1)$	$(p, n+1)$

Each of the qualified lines characterizes one L-label. For example, $\mathcal{C}(c, n) = (c, 0)$ encodes the fact that a route learned through a sequence of siblings the first of which learned it from a customer is exported to a provider, losing track of the passage through siblings. The transformation $\mathcal{R}(p, n) = \bullet$ encodes the fact that a route learned from a provider is not exported to a peer no matter what the passage through siblings. The transformations $\mathcal{S}(x, n) = (x, n + 1)$, for $x \in \{c, r, p\}$, encode the fact that any route is exported to a sibling with a decrease in preference.

L-labels $\mathcal{R}$ and $\mathcal{S}$ are strictly absorbent, but neither L-label $\mathcal{C}$ nor L-label $\mathcal{P}$ is absorbent: for example, $(c, n) \succ^L (c, 0) = \mathcal{C}(c, n)$, for n a positive integer. On the other hand, each of the L-labels $\mathcal{C}$, $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{S}$ is isotone. Hence, we can always discern the L-absorbency of a cycle from the L-absorbency of its circuits, Theorems 7 and 8. The L-label of a circuit is the composition of the L-labels of its links. The next chart shows the composition of two L-labels, with $\mathcal{Z}$ denoting the L-label that maps every L-attribute to $\bullet$.

	$\mathcal{C}$	$\mathcal{R}$	$\mathcal{P}$	$\mathcal{S}$
$\mathcal{C}$	$\mathcal{C}$	$\mathcal{Z}$	$\mathcal{Z}$	$\mathcal{C}$
$\mathcal{R}$	$\mathcal{R}$	$\mathcal{Z}$	$\mathcal{Z}$	$\mathcal{R}$
$\mathcal{P}$	$\mathcal{PC}$	$\mathcal{PC}$	$\mathcal{P}$	$\mathcal{P}$
$\mathcal{S}$	$\mathcal{SC}$	$\mathcal{SR}$	$\mathcal{SP}$	$\mathcal{SS}$

For instance, the composition of L-label $\mathcal{C}$ with L-label $\mathcal{S}$ gives L-label $\mathcal{C}$, that is, $\mathcal{CS} = \mathcal{C}$. The compositions $\mathcal{PC}$ and $\mathcal{PR}$ are the same label, represented in the chart by $\mathcal{PC}$: we have, $\mathcal{PC}(c, n) = (p, 0)$ and $\mathcal{PC}(r, n) = \mathcal{PC}(p, n) = \bullet$, for all n.

We are now well positioned to characterize the strictly absorbent cycles of BGP configured with the routing policies described above.

- **All links of the cycle are either peer links or sibling links: strictly absorbent**. Each link is strictly absorbent according to L-attributes and L-labels. Thus, each cycle is strictly absorbent.
- **The cycle has at least two different types of links among customer, peer, and provider: strictly absorbent**. Suppose the cycle has at least one customer link. The L-label of the circuit with endpoint at the tail of a customer link is of the form $\mathcal{CX}_1 \cdots \mathcal{X}_k \cdots \mathcal{Y} \cdots$, where $\mathcal{X}_i \in \{\mathcal{C}, \mathcal{S}\}$ and $\mathcal{Y} \in \{\mathcal{R}, \mathcal{P}\}$. Consulting the chart of pairwise compositions, we readily obtain $\mathcal{CX}_1 \cdots \mathcal{X}_k \cdots \mathcal{Y} \cdots = \mathcal{CY} \cdots = \mathcal{Z}$. Therefore, the cycle is strictly absorbent according to L-attributes and L-labels and, thus, it is strictly absorbent. The same reasoning applies to a cycle without customer links, but with at least one peer link and one provider link.
- **The cycle has at least one customer link and all other links are either customer links or sibling links: *not* strictly absorbent.** The L-label of a circuit with endpoint at the tail of a customer link is of the form $\mathcal{CX}_1 \cdots \mathcal{X}_i \cdots$ where $\mathcal{X}_i \in \{\mathcal{C}, \mathcal{S}\}$. Consulting the chart of pairwise compositions, we deduce $\mathcal{CX}_1 \cdots \mathcal{X}_i \cdots = \mathcal{C}$. Since $\mathcal{C}$ is *not* absorbent, the cycle is not absorbent according to L-attributes and L-labels and, thus, it is not strictly absorbent.
- **The cycle has at least one provider link and all other links are either provider or sibling links: *not* strictly absorbent.** The same reasoning as used above can be applied here.

Having identified the cycles which are not strictly absorbent, we show in Figure 6 a small network containing one of those

[6] In [20], routes learned from peers do not have to be preferred to routes learned from providers. We make the extra assumption because it seems to be valid in practice and it simplifies the presentation.

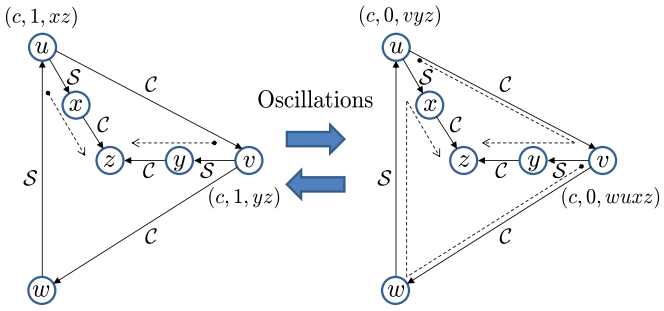

Fig. 6. Inter-domain routing policies involving customers, providers, and siblings. Destination is z. Attributes of elected routes at ASs u and v are indicated next to these nodes. Dashed paths mark the values of AS-PATH. BGP oscillates indefinitely between the routing state depicted on the left-hand side and the routing state depicted on the right-hand side.

cycles. The letters beside the links indicate their L-labels. AS w is a customer of AS v and AS v is a customer of AS u. ASs u and w are siblings. So are ASs u and x, and ASs v and y. AS z is a customer of both ASs x and y. For every link represented, there is another one in the opposite direction that we omit in order not to clutter the drawing. This network might have arisen from some initial arrangement where w was also a customer of u, but then the owners of u and w merged their business. As a consequence, u and w were configured to share all their elected routes, becoming siblings. When w was a customer of u, cycle $uvwu$ was strictly absorbent. Now that they are siblings, cycle $uvwu$ is not strictly absorbent.

The destination is node z alone. Suppose that, at some instant of time, u and v have just elected routes $(c, 1, xz)$ and $(c, 1, yz)$, learned from their siblings x and y, respectively, as shown on the left-hand side of Figure 6. These routes are exported counterclockwise around the cycle. Eventually, u and v elect routes $(c, 0, vyz)$ and $(c, 0, wuxz)$, respectively, learned from their neighbors around the cycle, as shown on the right-hand side of Figure 6. These routes, too, are exported counterclockwise around the cycle. At a later instant of time, u and v realize that the new routes they learn from around the cycle involve a loop containing them. When this happens, they revert to electing routes $(c, 1, xz)$ and $(c, 1, yz)$ learned externally to the cycle. A permanent routing state oscillation is established. The reason for this incorrectness can be traced to the routing policies for siblings, which perform "hot potato" routing between them in relation to customer routes, or peer routes, or provider routes learned from the outside. In Figure 6, u and v prefer to forward data-packets with destination in z to their clockwise customers v and w, respectively, rather than forward them more directly to z via siblings x and y, respectively. A permanent forwarding loop is potentially installed around cycle $uvwu$ except that AS-PATH prevents the formation of the forwarding loop, while converting it into route oscillations.

Preserving the Gao-Rexford routing policies for customer-provider and peer-peer relationships, it is possible to devise routing policies for sibling-sibling relationships that guarantee correctness of BGP. It is sufficient to have a sibling keep the preference of a route, rather than decrease it, as it is exported to a neighbor sibling, and let AS-PATH determine whether an AS forwards data-packets to customers, peers, or providers directly or through siblings. We model the new routing policies algebraically. There are only three proper L-attributes: 'learned from a customer', 'learned from a peer', 'learned from a provider', represented by the letters c, r, and p, respectively, and such that $c \prec^L r \prec^L p$. The L-label of a customer link, a peer link, a provider link, and a sibling link are still denoted by $\mathcal{C}$, $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{S}$, respectively, now given by the chart below.

	c	r	p
$\mathcal{C}$	c	•	•
$\mathcal{R}$	r	•	•
$\mathcal{P}$	p	p	p
$\mathcal{S}$	c	r	p

We note that $\mathcal{S}$ is just the identity label. All links are absorbent according to L-attributes and L-labels; all cycles are absorbent according to L-attributes and L-labels; all cycles are strictly absorbent according to BGP-attributes and BGP-labels, and BGP is correct.

With the previous routing policies, a unique stable state is reached in the network of Figure 6, whereby u elects route (c, xz) and v elects route (c, yz), corresponding to the values of AS-PATH shown on the left-hand side of the figure. In general, it can be shown that siblings sharing routes without a decrease or increase in preference neither introduce nor break routing anomalies, whatever the routing policies governing other pairs of ASs.

VII. Additional Related Work

We complement the citations made in the text with related work on the fundamentals of routing vector protocols. Two frameworks have been proposed for a unified study of these protocols: the Stable Paths Problem (SPP) [16], [38] and the Algebraic Theory of Routing (ATR) [19], [39] (which is the one we adopted here). An instance of the SPP is a network topology *and* one single destination in that topology. Routing policies are encoded by associating to each node an ordered set of permitted paths through which the destination can be reached. We refer the reader to the recent survey on the SPP [40]. By focusing on paths, which are concrete sub-parts of a network topology, the SPP provides a gentle introduction to conflicting routing policies and state oscillations in small networks.

An instance of the ATR is an ordered set of attributes together with labels that modify them. Such an instance reproduces the way routing policies are configured in practice, which is through parameters whose values (attributes) elect one route from among any set of routes (election operation) and rules for the importation and exportation of routes (labels of links). The ATR is more general than the SPP in that it can model routing systems that cannot be modeled with the SPP. For example, the ATR can be used to model routing vector protocols where permanent forwarding loops are a potential problem as is the case in the interconnection of routing instances through configuration of parameter AD and of RR [7]. The ATR is also more abstract than the SPP in that it provides for the description of, and reasoning

about, routing policies independently of concrete network topologies and destinations. In particular, the ATR allows us to talk about routing policies around cycles separately from the networks where they can be embedded, and it allows us to formulate routing policies that are the same across destinations, both of which features we exploited in this paper.

The ATR opens up the possibility of decomposing routing policies into simpler modules and infer on the correctness of routing vector protocols from the algebraic properties of the modules and the way they are combined [41]. Our model of BGP highlights this aspect. Routing policies realized with a combination of BGP's LOCAL-PREF and AS-PATH parameters do not have special properties that would lead to an easy identification of strictly absorbent cycles. The decomposition of routing policies into those implemented with LOCAL-PREF and those implemented with AS-PATH allowed us to equate the correctness of BGP to the absorbency of cycles with respect to the LOCAL-PREF routing policies alone. In turn, as example, this allowed a straightforward analysis of the role played by siblings in the correctness of inter-domain routing.

Many researchers have looked into the relation between routing policies and the behaviors they induce on routing vector protocols, both in the SPP and in the ATR. Several sufficient conditions for correctness of routing vector protocols have been published, but non-trivial necessary conditions have been much harder to find. A notable exception in the context of the SPP is given by Cittadini *et al.* [42]. The authors analyze the problem of *correctness under filtering*, previously posed by Feamster *et al.* [43].[7] This problem seeks conditions on the routing policies that guarantee correctness of a routing vector protocol for a given destination if nodes are allowed to change their routing policies by arbitrarily filtering routes. Cittadini et al. found that a routing vector protocol is correct under filtering if and only if the problem instance does not have a dispute reel, which is a certain type of circular dependency among permitted paths at various nodes. The problem addressed in the present paper is different. Routing policies are fixed and correctness is required across all destinations. Strict-cycle-absorbency is the sufficient and necessary condition for this type of correctness, being the first such condition framed in the ATR.

VIII. Conclusions

The operation of routing vector protocols is rather simple. It consists of iterations of electing routes and extending routes. This simplicity belies the difficulty in determining whether or not routing vector protocols behave correctly for arbitrary routing policies. Anomalous behaviors are diversified, but can only occur in networks with cycles. Therefore, it is natural to ask whether correctness of a routing vector protocol can be equated to how routing policies are configured at the nodes around the cycles of a network. We have shown that, when routing policies are not differentiated by destination, a routing vector protocol is correct if and only if the routing policies around every cycle in the network are strictly absorbent.

Instantiating the equivalence between correctness and strict-cycle-absorbency readily leads to results concerning concrete routing policies. We exemplified with next-hop routing policies and routing policies for sibling ASs of the Internet. Next-hop routing policies always pay a price in connectivity. Sibling ASs that exchange routes without altering their preference never introduce oscillations into BGP.

[7] In [42] and [43], the problem is called "safety under filtering."

Appendix
Absorbent Cycles With Isotone Links

We prove Theorem 8. Recall that cycle $C = u_0 u_1 \cdots u_{n-1}$ is absorbent if

$$\forall_{\alpha_0,\alpha_1,\ldots,\alpha_{n-1}} \quad \exists_{0\leq i<n}\ \alpha_i \prec L[u_i u_{i+1}](\alpha_{i+1}) \\ \vee\ \forall_{0\leq i<n} \alpha_i = L[u_i u_{i+1}](\alpha_{i+1}), \qquad (19)$$

where indexes are interpreted modulus n. We first show that if C is absorbent, then $\alpha \preceq L[u_i C u_i](\alpha)$ for every node u_i of C and every attribute α. Assign external attributes to all nodes of C as follows: $\alpha_i = \alpha$ and

$$\alpha_{i-k} = L[u_{i-k} C u_i](\alpha), \qquad (20)$$

for $1 \leq k < n$. Thus, $\alpha_{i-k} = L[u_{i-k} u_{i-k+1}](\alpha_{i-k+1})$. Satisfaction of Condition (19) imposes that $\alpha = \alpha_i \preceq L[u_i u_{i+1}](\alpha_{i+1})$. Thus, we write

$$\begin{aligned} \alpha &\preceq L[u_i u_{i+1}](\alpha_{i+1}) \\ &= L[u_i u_{i+1}](L[u_{i+1} C u_i](\alpha)) \quad \text{(from (20), } k = n-1) \\ &= L[u_i C u_i](\alpha). \end{aligned}$$

Note that isotonicity was not required in this part of the proof.

Next, we assume that all links are isotone and show that if $\alpha \preceq L[u_i C u_i](\alpha)$ for every node u_i of C and every attribute α, then C is absorbent. In fact, we prove the contrapositive statement. If C is *not* absorbent, then there is an index i and attributes α_{i-k}, for $1 \leq k < n$, such that

$$\alpha_{i-k} \succeq L[u_{i-k} u_{i-k+1}](\alpha_{i-k+1}) \qquad (21)$$

and

$$\alpha_i \succ L[u_i u_{i+1}](\alpha_{i+1}). \qquad (22)$$

We show by induction that $\alpha_{i-k} \succeq L[u_{i-k} C u_i](\alpha_i)$, for $1 \leq k < n$. The base case is Inequality (21) with $k = 1$. Using the isotonicity of link $u_{i-(k+1)} u_{i-k}$ on the induction hypothesis, we write

$$\begin{aligned} \alpha_{i-(k+1)} &\succeq L[u_{i-(k+1)} u_{i-k}](\alpha_{i-k}) && \text{(from (21))} \\ &\succeq L[u_{i-(k+1)} u_{i-k}](L[u_{i-k} C u_i](\alpha_i)) && \text{(isotonicity)} \\ &= L[u_{i-(k+1)} C u_i](\alpha_i), \end{aligned}$$

thus proving the induction step. Using the isotonicity of link $u_i u_{i+1}$, we write

$$\begin{aligned} \alpha_i &\succ L[u_i u_{i+1}](\alpha_{i+1}) && \text{(from (22))} \\ &\succeq L[u_i u_{i+1}](L[u_{i+1} C u_i](\alpha_i)) && \text{(isotonicity)} \\ &= L[u_i C u_i](\alpha_i). \end{aligned}$$

Therefore, node u_i and attribute $\alpha = \alpha_i$ are such that $\alpha \succ L[u_i C u_i](\alpha)$. Combining the two cases above, we conclude that cycle C is absorbent if and only if $\alpha \preceq L[u_i C u_i](\alpha)$ for all nodes u_i and attributes α.

ACKNOWLEDGMENTS

The author is grateful to J. Brázio, L. Vanbever, F. Le, and J. Rexford for their many comments on earlier drafts of this paper, and to the anonymous reviewers for remarks that lead to a better presentation of this work.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, *A Border Gateway Protocol 4 (BGP-4)*, document RFC 4271, Jan. 2006.
[2] R. White, D. McPherson, and S. Sangli, *Practical BGP*. Boston, MA, USA: Addison-Wesley, 2005.
[3] P. Lahiri, G. Chen, P. Lapukhov, E. Nkposong, D. Maltz, R. Toomey, and L. Yuan, "Routing design for large scale data centers: BGP is the better IGP," presented at the NANOG, Jun. 2012. [Online]. Available: https://www.nanog.org/meetings/nanog55/presentations/Monday/Lapukhov.pdf
[4] G. Malkin, *RIP Version 2*, document RFC 2453, Nov. 1998.
[5] A. Retana, R. White, and D. Slice, *EIGRP for IP: Basic Operation and Configuration*. Reading, MA, USA: Addison-Wesley, 2000.
[6] D. A. Maltz *et al.*, "Routing design in operational networks: A look from the inside," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 27–40.
[7] F. Le, G. G. Xie, and H. Zhang, "Understanding route redistribution," in *Proc. IEEE ICNP*, Oct. 2007, pp. 81–92.
[8] F. Le, G. G. Xie, D. Pei, J. Wang, and H. Zhang, "Shedding light on the glue logic of the Internet routing architecture," in *Proc. ACM SIGCOMM*, Aug. 2008, pp. 39–50.
[9] J. Feigenbaum, R. Sami, and S. Shenker, "Mechanism design for policy routing," in *Proc. 23rd Annu. ACM Symp. Principles Distrib. Comput.*, Jul. 2004, pp. 11–20.
[10] M. Schapira, Y. Zhu, and J. Rexford, "Putting BGP on the right path: A case for next-hop routing," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw.*, Oct. 2010, Art. no. 3.
[11] T. G. Griffin, "The stratified shortest-paths problem," in *Proc. 2nd Int. Conf. Commun. Syst. Netw.*, Jan. 2010, pp. 268–277.
[12] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proc. ACM SIGCOMM*, Oct. 1994, pp. 234–244.
[13] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *Proc. ACM SIGCOMM*, Aug. 2004, pp. 133–144.
[14] Y. Yang and J. Wang, "Design guidelines for routing metrics in multihop wireless networks," in *Proc. IEEE INFOCOM*, Apr. 2008, pp. 2288–2296.
[15] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Comput. Netw.*, vol. 32, no. 1, pp. 1–16, 2000.
[16] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, Apr. 2002.
[17] R. Engelberg, A. Fabrikant, M. Schapira, and D. Wajc, "Best-response dynamics out of sync: Complexity and characterization," in *Proc. 14th ACM Conf. Electron. Commerce*, Jun. 2013, pp. 379–396.
[18] D. P. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1991.
[19] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, pp. 1160–1173, Oct. 2005.
[20] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Netw.*, vol. 9, no. 6, pp. 681–692, Dec. 2001.
[21] Y. Liao, L. Gao, R. Guérin, and Z.-L. Zhang, "Safe interdomain routing under diverse commercial agreements," *IEEE/ACM Trans. Netw.*, vol. 18, no. 6, pp. 1829–1840, Dec. 2010.
[22] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 547–556.
[23] G. Huston. (2014). *Personal Site of Geoff Huston*. [Online]. Available: http://www.potaroo.net/
[24] X. Fan, J. Heidemann, and R. Govindan, "Evaluating anycast in the domain name system," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1681–1689.
[25] H. A. Alzoubi, S. Lee, M. Rabinovich, O. Spatscheck, and J. Van Der Merwe, "A practical architecture for an anycast CDN," *ACM Trans. Web*, vol. 5, no. 4, Oct. 2011, Art. no. 17.
[26] X. Zhao *et al.*, "An analysis of BGP multiple origin AS (MOAS) conflicts," in *Proc. 1st ACM SIGCOMM Workshop Internet Meas.*, Aug. 2001, pp. 31–35.
[27] L. Cittadini, M. Rimondini, S. Vissicchio, M. Corea, and G. Di Battista, "From theory to practice: Efficiently checking BGP configurations for guaranteed convergence," *IEEE Trans. Netw. Service Manage.*, vol. 8, no. 4, pp. 387–400, Dec. 2011.
[28] A. Wang *et al.*, "FSR: Formal analysis and implementation toolkit for safe interdomain routing," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1814–1827, Dec. 2012.
[29] M. Gondran and M. Minoux, *Graphes, Dioïdes et Semi-Anneaux: Nouveaux Modeles et Algorithmes*. Paris, France: Tec & Doc, 2001.
[30] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *Proc. ACM SIGCOMM*, Aug. 2005, pp. 1–12.
[31] B. Carré, *Graphs and Networks*. Oxford, U.K.: Clarendon, 1979.
[32] J. L. Sobrinho and T. Quelhas, "A theory for the connectivity discovered by routing protocols," *IEEE/ACM Trans. Netw.*, vol. 20, no. 3, pp. 677–689, Jun. 2012.
[33] T. Erlebach, A. Hall, A. Panconesi, and D. Vukadinović, "Cuts and disjoint paths in the valley-free path model," *Internet Math.*, vol. 3, no. 3, pp. 333–359, 2006.
[34] F. Le and G. G. Xie, "On guidelines for safe route redistributions," in *Proc. ACM SIGCOMM Workshop Internet Netw. Manage.*, 2007, pp. 274–279.
[35] F. Le and J. L. Sobrinho, "Interconnecting routing instances," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 540–553, Apr. 2014.
[36] E. Katz-Bassett *et al.*, "Machiavellian routing: Improving Internet availability with BGP poisoning," in *Proc. 10th ACM Workshop Hot Topics Netw.*, Nov. 2011, Art. no. 11.
[37] R. Gao, C. Dovrolis, and E. W. Zegura, "Interdomain ingress traffic engineering through optimized AS-path prepending," in *Proc. 4th IFIP-TC6 Int. Conf. Netw. Technol., Services, Protocols*, May 2005, pp. 647–658.
[38] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. ACM SIGCOMM*, Aug. 1999, pp. 277–288.
[39] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 727–735.
[40] L. Cittadini, G. Di Battista, and M. Rimondini, "On the stability of interdomain routing," *ACM Comput. Surv.*, vol. 44, no. 4, Sep. 2012, Art. no. 26.
[41] A. J. T. Gurney and T. G. Griffin, "Lexicographic products in metarouting," in *Proc. IEEE ICNP*, Oct. 2007, pp. 113–122.
[42] L. Cittadini, G. Di Battista, M. Rimondini, and S. Vissicchio, "Wheel + Ring = Reel: The impact of route filtering on the stability of policy routing," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1085–1096, Aug. 2011.
[43] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," *IEEE/ACM Trans. Netw.*, vol. 15, no. 6, pp. 1266–1279, Dec. 2007.

João Luís Sobrinho received the Licenciatura and Ph.D. degrees in electrical and computer engineering from the Instituto Superior Técnico, Universidade de Lisboa, Portugal, in 1990 and 1995, respectively. Before joining academia in 1997, he was with Bell Labs, Lucent Technologies, for two years. He is currently an Associate Professor with the Department of Electrical and Computer Engineering, Instituto Superior Técnico, and a Senior Researcher with the Instituto de Telecomunicações.

He won an Internet Society Applied Networking Research Prize 2015, the IEEE Communications Society William R. Bennett Prize 2006, and an IEEE PIMRC 1994 Best Ph.D. Student Paper Award.