Stabilizing BGP Through Distributed Elimination of Recurrent Routing Loops

João Luís Sobrinho, David Fialho, and Paulo Mateus
Instituto de Telecomunicações and Instituto Superior Técnico
joao.sobrinho@lx.it.pt, fialho.david@protonmail.com, and pmat@ist.utl.pt

***Abstract*—Despite years of research, the Internet still lacks a routing protocol with guaranteed termination. As is well-known, decentralization of routing decisions among the Autonomous Systems (ASes) that comprise the Internet may result in permanent oscillations of the state of its routing protocol—the Border Gateway Protocol (BGP). Some permanent oscillations are made from routing loops—the propagation of routing messages around the cycles of a network—that come back time and again. We discovered that the routing loop detection capability of BGP can be sharpened to predict which routing loops potentially recur and that the import policies can be adjusted to prevent the recurrence. The resulting protocol, named Self-Stable BGP (SS-BGP), is more stable than BGP. For the broad and common class of isotone routing policies, all permament oscillations are made from recurrent routing loops. For this class of routing policies, SS-BGP terminates. Our simulations with realistic Internet topologies and realistic variations of the Gao-Rexford (GR) inter-AS routing policies show that SS-BGP arrives at stable states at the expense of alterations in the import policies of only a handful of ASes.**

I. Introduction

The Border Gateway Protocol (BGP) [1] interconnects the tens of thousands of Autonomous Systems (AS) that constitute the Internet, providing reachability to hundreds of thousands of IP prefixes. BGP instantiates a separate computation process per destination. For a given destination, nodes import routes received from their neighbors, elect a best route among the imported routes, and export the elected route to their neighbors. Every route contains a cost and a path, corresponding, respectively, to the LOCAL-PREF and AS-PATH attributes of BGP.

Costs take primacy in route election, being transformed by import and export routing policies, set autonomously at each node. Autonomy has a price. The routing policies of different nodes may be incompatible, causing permanent oscillations of the state of BGP [2], [3]. Oscillations overload the control plane and wreak havoc in the data plane, attracting data-packets to forwarding loops. Paths, contained in routes, allow detection of routing loops[1], with BGP stopping the propagation of routing messages past those loops. One could expect that stopping the propagation of routing messages past routing loops would end oscillations, eventually. However, that is not the case, for two reasons. First, some permanent state oscillations are unrelated to routing loops. Second, some routing loops recur infinitely often.

[1]A routing loop refers to the propagation of a routing message around a cycle of a network and should not be confused with a forwarding loop, which refers to the expedition of data-packets around a cycle of a network.

In this paper, we present an extension to BGP, named Self-Stable BGP (SS-BGP), that prevents the recurrence of routing loops. SS-BGP adds to BGP the following test and action: if, and only if, a node learns from a neighbor a route that simultaneously offers the best cost to reach a destination and reveals a routing loop, then the node ignores future routes to the destination learned from that neighbor. SS-BGP uses standard BGP routing messages and can be deployed incrementally.

For a broad and common class of routing policies, characterized by the property known as isotonicity [4], [5], all permanent state oscillations are made from recurrent routing loops. For this class of routing policies, SS-BGP terminates, signifying that it reaches a stable state whatever the initial state and whatever the delays in the delivery of routing messages. Isotonicity means that if a node prefers the cost of one route over that of another route, then a neighbor node does not have the exact opposite preference between the costs of the two routes it learns from the former node. When routing policies are not isotone, SS-BGP still terminates in cases where BGP does not, although termination is not universally guaranteed.

The remainder of the paper is organized as follows. Section II introduces SS-BGP by means of two examples. Section III presents the routing model and describes SS-BGP. Section IV proves that isotonicity guarantees termination of SS-BGP, while Section V discusses the merits of isotonicity and its use in practice. Section VI evaluates the performances of BGP and SS-BGP. Section VII reviews related work and Section VIII concludes the paper.

II. Self-stable BGP in action

The Gao-Rexford (GR) model provides a baseline of understanding for inter-AS routing [6]. The model classifies business relationships between neighbor ASes into either provider-customer or peer-peer, and derives sensible routing policies in that context. Strict adherence to the GR routing policies stabilizes BGP. However, there is strong evidence that the GR routing policies are now sometimes violated, either intentionally [7], [8], [9], [10], to reflect complex goals of AS administrators, or inadvertently [11], [12], [8], as a result of misconfigurations. In addition, new types of business relationships have emerged [13]. In the next two sections, we introduce SS-BGP with two perturbations of the GR routing policies that make BGP prone to oscillations.

978-1-5090-6501-1/17/$31.00 @2017 IEEE

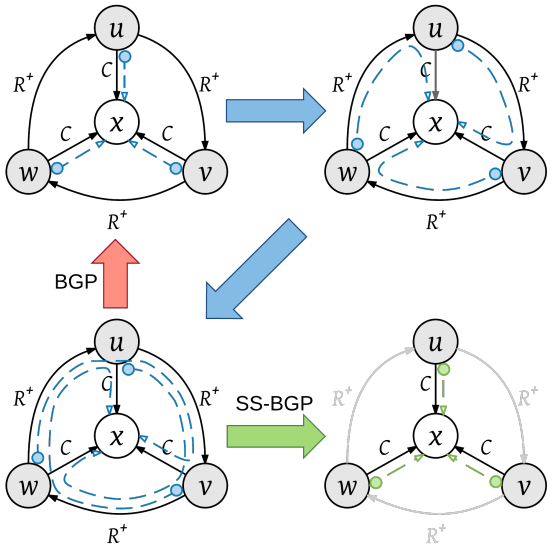

Fig. 1. BGP does not terminate, whereas SS-BGP terminates. Letters C and R^+ represent customer links and peer$^+$ links, respectively. The destination is x. Dashed arrows represent elected routes and dim links represent deactivated neighbors.

A. *Peer$^+$s*

The GR routing policies presuppose that neighbor ASes have either a provider-customer or a peer-peer business relationship. We call customer link to a link that joins a provider to a customer; provider link to a link that joins a customer to a provider; and peer link to a link that joins a peer to another peer. The GR routing policies distinguish three types of routes: customer routes, which are those learned from a customer; peer routes, which are those learned from a peer; and provider routes, which are those learned from a provider. A customer route is better than a peer route and the latter is better than a provider route. All routes are exported to a customer and customer routes are exported to all neighbors, these being the only exportations allowed.

In violation of the GR routing policies, some ASes sometimes prefer routes learned from a peer over routes learned from a customer to reach a destination [8], [9]. We introduce the concept of *peer$^+$* to model such cases. A peer$^+$ is a neighbor such that the routes learned from that neighbor are preferred to customer routes. A peer$^+$ link joins a node to a peer$^+$. A route learned from a peer$^+$ is a peer$^+$ route. We assume that peer$^+$ routes are exported to all neighbors, as customer routes are, because, after all, a node that elects a peer$^+$ route to the detriment of a customer route maintains the obligation to provide reachability to their common destination.

Routes are couplets of the form (α, P), where α is a cost and P is a path. Costs are one of r^+, c, r, and p, denoting, respectively, peer$^+$, customer, peer, and provider routes. In the network of Figure 1, letters C and R^+ stand for customer link and peer$^+$ link, respectively. Node x is a customer of u, v, and w. Node v is a peer$^+$ of u; w is a peer$^+$ of v; and u is a peer$^+$ of w. The destination is x.

For simplicity of presentation, we assume that it takes one unit of time for every routing message to travel across a link. At $T = 0$, each of u, v, and w elects route (c, x), learned directly from x, and exports this route counter-clockwise around the cycle (upper left sub-figure). At $T = 1$, u elects route (r^+, vx), learned from v, which is better than the direct route (c, x), because r^+ is better than c; w elects route (r^+, ux), learned from u; and v elects route (r^+, wx), learned from w (upper right). The peer$^+$s routes are exported further counter-clockwise around the cycle, so that, at $T = 2$, u elects route (r^+, vwx); v elects route (r^+, wux); and w elects route (r^+, uvx) (lower left). These routes, too, are exported counter-clockwise around the cycle. At $T = 3$, u learns route $(r^+, vwux)$ from v; v learns route $(r^+, wuvx)$ from w; and w learns route $(r^+, uvwx)$ from u. Each of these routes reveals a routing loop. They are all discarded, leading u, v, and w, to elect again the direct route (c, x) (upper left). The state of BGP goes back to that at $T = 0$, the routing loops recur indefinitely, and BGP oscillates forever.

However, u, v, and w can all detect at $T = 3$ that something is amiss, in that the neighbor from which they learn the best cost to reach x simultaneously reveals a routing loop. For instance, at $T = 3$, u learns route $(r^+, vwux)$ from v, whose cost r^+ is better than cost c of the direct route (c, x), but whose path $vwux$ reveals routing loop $uvwu$. SS-BGP adds to BGP the following test and action.

SS-BGP: If a route learned from a neighbor carries the best cost to reach a destination, but the path component of the route reveals a routing loop, then future routes to the destination learned from that neighbor are not imported.

We say that a node 'detects an incongruous cost' when the test of SS-BGP is positive and that a node 'deactivates' a neighbor when the action of SS-BGP is executed. Routes learned from a deactivated neighbor are stored locally to allow a fast routing response if and when the neighbor is reactivated.

With SS-BGP deployed in the network of Figure 1, at $T = 3$, each of u, v, and w detects an incongruous cost and each deactivates its clockwise neighbor (lower right). Therefore, at $T = 4$, u does not import route (r^+, vx), learned from v, maintaining its elected route (c, x); v does not import route (r^+, wx), learned from w, maintaining its elected route (c, x); and w does not import route (r^+, ux), learned from u, maintaining its elected route (c, x). A stable state has been reached, one that coincides with a stable state of BGP in a network where the links that join a node to a deactivated neighbor have been removed.

Deactivating a neighbor alters import policies and entails a reduction in the reliability of the network. For instance, in the network of Figure 1, if link ux fails after the stable state of SS-BGP is reached (lower right), then u is left without a route to x, because its only remaining neighbor, node v, is inactive. Our simulations, with realistic Internet topologies and realistic routing policies show that SS-BGP stabilizes with the deactivation of only very few neighbors (see Section VI). Nevertheless, it is important to have a strategy to reactivate neighbors that were previously deactivated. First, because the

network is dynamic and a previous deactivation might not repeat itself after changes in topology or routing policies. Second, because SS-BGP deactivates more neighbors than are needed for a stable state to be reached. For instance, in the network of Figure 1, SS-BGP deactivates the neighbors of u, v, and w around cycle $uvwu$. However, only one of u, v, and w would need to deactivate its neighbor around the cycle for termination of the protocol (see also the false positives of Section III-D).

The use of timers provides a solution to reactivation. Every inactive neighbor is associated with a reactivation time that is chosen when the neighbor is deactivated. Assuming a static topology and static routing policies, if all nodes around a cycle choose roughly the same reactivation time for their inactive neighbors, then recurrent routing loops may be reinstated around that cycle, to be eliminated again by SS-BGP. However, if one node chooses a longer reactivation time than all other nodes, then recurrent routing loops likely are not reinstated around the cycle. For example, in Figure 1 suppose that u and v reactivate v and w, respectively, before w reactivates u. After reactivation of v and w, the protocol reaches a stable state whereby u elects route (r^+, vwx); v elects route (r^+, wx); and w elects route (c, x). Node w learns route $(r^+, uvwx)$ from u, which is discarded, since it reveals a routing loop. Later, when w reactivates u, there is no candidate route at w to reach x via u. The elected routes of all nodes are untouched. Reactivation times can be chosen randomly with a mean that is on a coarser timescale than that of the propagation of routing messages. Mean reactivation times can be made adaptive, increasing with the number of deactivations of the neighbor and decreasing while the neighbor is active.

B. Siblings

Sibling-sibling relationships complement the business relationships of the GR model. A sibling link joins a sibling to another sibling. Siblings share all routes between them. In the routing policies presented by Liao et al. [13], siblings avoid sending data-packets through each other towards an outside destination: when a route is exported to a sibling, it keeps the information of how it was learned from outside a stretch of siblings, but the preference of the route decreases.

With siblings, routes are of the form (α, n, P), where (α, n) is a cost and P is a path. In cost (α, n), α is one of c, r, and p, subject to the same interpretation as in the previous section, and n is a natural number representing the propagation of a route across n consecutive siblings. Cost (α, n) is better than cost (β, m) if (i) α is better than β; or (ii) $\alpha = \beta$ and n is smaller than m. In the network of Figure 2, letters C and S stand for customer link and sibling link, respectively. Nodes x and y are both providers of z, and siblings of u and v, respectively. Node u is a provider of v, which is a provider of w. Nodes u and w are siblings. The destination is z.

We assume that it takes one unit of time both for a routing message to travel from v to u and for a routing message to travel from u to v via w. At $T = 0$, nodes u and v have just elected routes $(c, 1, xz)$ and $(c, 1, yz)$, learned from x and y, respectively, and exported these routes counter-clockwise around the cycle. At $T = 1$, u elects route $(c, 0, vyz)$, learned from v, which is better than route $(c, 1, xz)$; and v elects route $(c, 0, wuxz)$, learned from w, which is better than route $(c, 1, yz)$. The newly elected routes are exported counter-clockwise around the cycle. At $T = 2$, u learns route $(c, 0, vwuxz)$ from v, which reveals a routing loop and is discarded, electing instead route $(c, 1, xz)$. Similarly, v learns route $(c, 0, wuvyx)$ from w and discards this route, electing instead route $(c, 1, yz)$. With BGP, the state of the protocol goes back to that at $T = 0$, the routing loops recur indefinitely, and BGP oscillates forever. In contrast, SS-BGP terminates. At $T = 2$, u detects an incongruous cost—cost $(c, 0)$ of route $(c, 0, vwuxz)$ is better than cost $(c, 1)$ of route $(c, 1, xz)$, but path $vwuxz$ reveals routing loop $uvwu$—and deactivates v. Likewise, v detects an incongruous cost and deactivates w. A stable state is reached, whereby u elects route $(c, 1, xz)$; v elects route $(c, 1, yz)$; and w elects route $(c, 2, uxz)$.

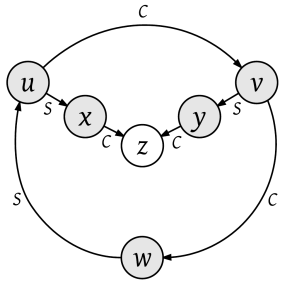

Fig. 2. BGP does not terminate, whereas SS-BGP terminates. Letters C and S represent a customer link and a sibling link, respectively, and the destination is z.

III. Self-stable BGP

Section III-A presents a model for the routing policies that are realizable with BGP, adapted from [14], and Section III-B relates termination of BGP to the combined routing policies of nodes around the cycles of a network. Section III-C presents the pseudo-code of canonical SS-BGP and Section III-D introduces a variant of SS-BGP, which more accurately predicts recurrent routing loops.

A. Routing policies

A BGP route is an association between a destination and a set of attributes. LOCAL-PREF and AS-PATH are, respectively, the first and second attributes considered in the election of a route from among a set of candidate routes. The LOCAL-PREF attribute of a route is a level of preference assigned locally to the route. The AS-PATH attribute of a route contains the sequence of nodes through which the destination has been successively advertised. BGP routing messages contain the AS-PATH attribute, but not the LOCAL-PREF attribute. They may optionally contain a COMMUNITY attribute. The LOCAL-PREF attribute of a route may depend on the AS-PATH attribute and on the COMMUNITY attribute, whenever present, in received BGP routing messages [1].

Rather than using LOCAL-PREF explicitly, we prefer to characterize BGP routes pertaining to a destination by couplets of the form (α, P), where α is a cost, and P is a path, as we did in the examples of Section II, because such representation is better suited to semantic interpretations of routing policies.[2] Costs are totally ordered by $\prec$. We write: $\alpha \preceq \beta$ if $\alpha \prec \beta$ or $\alpha = \beta$; $\alpha \succ \beta$ if $\beta \prec \alpha$; and $\alpha \succeq \beta$ if $\beta \preceq \alpha$. If $\alpha \prec \beta$, then we say that cost α is *better* than cost β and that cost β is *worse* than cost α. The election among costs is represented by binary operation $\sqcap$: $\alpha \sqcap \beta = \alpha$ if $\alpha \preceq \beta$, and $\alpha \sqcap \beta = \beta$, otherwise. We write $\sqcap\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$ for the best cost of the set of costs $\{\alpha_0, \alpha_1, \ldots, \alpha_{n-1}\}$. The symbol $\bullet$ denotes unreachability, being worse than any cost.

Paths are totally ordered by $\lhd$. Order $\lhd$ respects the lengths of paths, but is otherwise arbitrary. Given paths P and Q,

$$P \lhd Q \Rightarrow |P| \leq |Q|,$$

where $|P|$ denotes the number of links of P. The election among paths is represented by binary operation $\triangle$: $P \triangle Q = P$ if $P \lhd Q$ or $P = Q$, and $P \triangle Q = Q$, otherwise. We write $\triangle \{P_0, P_1, \ldots, P_{n-1}\}$ for the best path of the set of paths $\{P_0, P_1, \ldots, P_{n-1}\}$ with respect to $\lhd$.

Couplets are ordered lexicographically. Couplet (α, P) is *better* than couplet (β, Q), and couplet (β, Q) is *worse* than couplet (α, P), if

$$\alpha \prec \beta \vee (\alpha = \beta \wedge P \lhd Q).$$

Function $T[uv]$, called the *extender* of link uv, describes how the costs of routes are transformed from v to u. A cost β at v gives rise to cost $T[uv](\beta)$ at u. Function $T[uv]$ subsumes the export policies of v in relation to u and the import policies of u in relation to v. By modeling the transformation of costs with a function, we exclude the possibility of two routes with the same cost at v producing two routes with different costs at u. A route (β, Q) at v gives rise to learned route $(T[uv](\beta), vQ)$ at u, if $T[uv](\beta) \prec \bullet$. Learned route $(T[uv](\beta), vQ)$ becomes a candidate route for election at u, if $u \notin vQ$, that is, if a routing loop is not revealed. If either $T[uv](\beta) = \bullet$ or $u \in vQ$, then u cannot reach the destination via v, a fact which we represent by couplet $(\bullet, \bullet)$.

Isotonicity is an important property for guaranteed termination of SS-BGP (see Section IV). The routing policies of link uv are *isotone* if the relative preference between two costs at v is preserved between the corresponding costs at u:

$$\forall_{\alpha,\beta}\ \alpha \preceq \beta \Rightarrow T[uv](\alpha) \preceq T[uv](\beta).$$

In other words, the routing policies of link uv are isotone if $T[uv]$ is an increasing function. Equivalently, the routing policies of link uv are isotone if $T[uv]$ preserves binary operation $\sqcap$:

$$\forall_{\alpha,\beta}\ T[uv](\alpha \sqcap \beta) = T[uv](\alpha) \sqcap T[uv](\beta).$$

[2]For instance, the GR routing policies are positively stated in terms of customer, peer, and provider routes. However, different AS administrators may assign different values of LOCAL-PREF to any of these types of routes.

Both the GR routing policies with peer+, Section II-A, and the GR routing policies with siblings, Section II-B, are isotone. For example, consider the case of the GR routing policies with peer+s and let u be a provider of v. We have $T[uv](r^+) = T[uv](c) = c$ (peer+ and customer routes are exported to providers) and $T[uv](r) = T[uv](p) = \bullet$ (peer and provider routes are not exported to providers). Applying $T[uv]$ to each of the costs in the ordered sequence

$$r^+ \prec c \prec r \prec p,$$

yields the sequence

$$c = c \prec \bullet = \bullet,$$

which is ordered as well. The same is true if u is a peer, a peer+, or a customer of v. For the isotonicity of siblings, we refer to [14].

B. Termination and cost-circuits

The state of BGP is composed of the candidate and elected routes at the various nodes and the routing messages in transit across links. A state is *stable* if there are no routing messages in transit. BGP *terminates* if a stable state is eventually reached, whatever the initial state and whatever the delays of routing messages across links. A *cost-circuit* is a pair (α, uCu), where α is a cost and uCu is a circuit around cycle C starting and ending at node u of C. A route with cost α dispatched by u around C (in the direction opposite to that of the circuit) is learned back at u with cost $T[uCu](\alpha)$, where $T[uCu]$ is the composition of the extenders of all links of the circuit. Thus, function $T[uCu]$ models the transformation of the costs of routes around circuit uCu. Cost-circuit (α, uCu) is *absorbent* if the route learned from around the cycle has a worse cost than or the same cost as the dispatched route, $T[uCu](\alpha) \succeq \alpha$; it is *not absorbent* if it has a better cost, $T[uCu](\alpha) \prec \alpha$.

The work of Sobrinho [14] relates termination of BGP to a property of routing policies around the cycles of a network. From that work, the following two propositions relating termination of BGP to cost-circuits can be derived, and they provide insight into the design and termination of SS-BGP.

Proposition 1 *If there is a cost-circuit that is not absorbent, then BGP does not terminate for some destination, either in the network or in some sub-network obtained after a number of link failures.*

This proposition justifies chasing and eliminating non-absorbent cost-circuits. However, elimination of those cost-circuits may not suffice for termination of BGP.

Proposition 2 *If routing policies are isotone and all cost-circuits are absorbent, then BGP terminates for all destinations.*

This proposition implies that, in the case of isotone routing policies, the elimination of non-absorbent cost-circuits stabilizes BGP.

C. Pseudo-code

SS-BGP can be regarded as an extension to BGP that eliminates enough non-absorbent cost-circuits at run time to attain a stable state. As BGP, SS-BGP treats different destinations separately. Given a destination, each out-neighbor[3] of a node is either active or inactive. Initially, all out-neighbors of a node are active. The node stores candidate routes to reach the destination via each of its out-neighbors, but elects a route exclusively among the candidate routes of its active out-neighbors. Algorithm 1 presents canonical pseudo-code of SS-BGP for when node u receives routing message (β, vQ) from its active out-neighbor v. Variables $CostT_u[w]$ and $PathT_u[w]$ store, respectively, the candidate cost and the candidate path to reach the destination via out-neighbor w, and variables $Cost_u$ and $Path_u$, store, respectively, the elected cost and elected path to reach the destination. The special cases $CostT_u[u]$ and $PathT_u[u]$ contain the fixed cost and the fixed path of a route originated at u. A node u that originates a route pertaining to the destination, $(CostT_u[u], PathT_u[u]) \neq (\bullet, \bullet)$, is called a destination node. Anycast routing is anticipated, in which case there is more than one destination node.

The election of a cost in Line 2 runs over all active out-neighbors of u to the exception of v, and the election of a path in Line 3 runs over all active out-neighbors of u, to the exception of v, for which the candidate cost equals the elected cost. The only differences between SS-BGP and BGP are in Lines 5 and 6, which are absent in the latter protocol. The test in Line 5 is positive if and only if u detects an incongruous cost, that is, if and only if the cost learned from v is better than the best of all candidate costs, but the path learned from v reveals a routing loop. If u detects an incongruous cost, then it deactivates v in Line 6, meaning that future routes to the destination learned from v are not imported.

D. False positives and the enhanced SS-BGP

Due to race conditions, a node may detect an incongruous cost while the underlying cost-circuit is absorbent. Suppose that node u elects a route with cost α learned from outside cycle C. The route propagates around C, later being learned back at u as a route with cost $T[uCu](\alpha)$. While the route propagates around C, the best route at u learned from outside C changes to one with cost β. Node u may detect an incongruous cost, $T[uCu](\alpha) \prec \beta$, while cost-circuit (α, uCu) is absorbent, $T[uCu](\alpha) \succeq \alpha$. We call *false positive* to such a condition. The network of Figure 3 provides a concrete example. Letters C, R^+, and R stand for customer link, peer$^+$ link, and peer link, respectively. The destination is z. In the stable state, u elects route (c, yz). Now, suppose that link yz fails. News of the failure reach u first through neighbor y, so that u changes its elected route to (c, xyz), learned from x. This route propagates around cycle $uvwu$, from u to w to v, and back to u. Before the route arrives back at u, this node learns that x no longer has a route to z and, thus, elects route (r, z), learned directly from z. When route (c, xyz), propagating around cycle $uvwu$, is learned back at u as route $(c, vwuxyz)$, this node detects an incongruous cost—since c is better than r—and deactivates v, despite the absorbency of cost-circuit $(c, uvwu)$.

[3]In this section, we distinguish between out-neighbor and in-neighbor.

Algorithm 1 SS-BGP code for when node u receives routing message (β, vQ) from its active out-neighbor v.

```
    CostT_u[v] := T[uv](β); PathT_u[v] := vQ
    Cost_u := ⊓{CostT_u[w] | w ≠ v}
    Path_u :=△ {PathT_u[w] | Cost_u = CostT_u[w], w ≠ v}
    if CostT_u[v] = • or u ∈ PathT_u[v] then
5:      if CostT_u[v] ≺ Cost_u then          ▷ incongruous cost
            deactivate v
        CostT_u[v] := •; PathT_u[v] := •
    else                                     ▷ CostT_u[v] ≺ • and u ∉ PathT_u[v]
        Cost_u := Cost_u ⊓ CostT_u[v]
10:     if Cost_u = CostT_u[v] then
            Path := Path_u △ PathT_u[v]
    if Cost_u or Path_u have changed then
        if Cost_u ≺ • then
            for all w in-neighbor do
15:             send (Cost_u, uPath_u) to w
        else
            for all w in-neighbor do
                send (•, •) to w
```

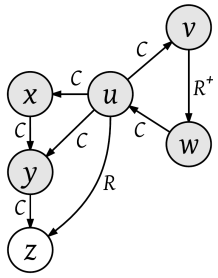

Fig. 3. When yz fails, u needlessly deactivates v with SS-BGP, whereas it will not deactivate v with ESS-BGP.

Enhanced SS-BGP (ESS-BGP) refines the test for deactivation used by SS-BGP for the purpose of reducing false positives to a bare minimum (see forthcoming Theorem 1). With ESS-BGP, a node only deactivates a neighbor if an incongruous cost is detected and the tail of the path learned from the neighbor, with the loop discounted, coincides with the elected path. In order to make this condition precise, we introduce the following notation. If P is a path and u is a node of P, then the tail of P at u, denoted by $u \circ P$, is the subpath of P that starts at the successor of u along P. For example, if $P = rstuvwxyz$, then $u \circ P = vwxyz$. With ESS-BGP, Line 5 of Algorithm 1 is substituted by

if $CostT_u[v] \prec Cost_u$ and $u \circ PathT_u[v] = Path_u$ **then**

Going back to the example of Figure 3, when u learns route $(c, vwuxyz)$ from around cycle $uvwu$, it compares tail xyz of path $vwuxyz$ ($xyz = u \circ vwuxyz$) with path z of the elected route. They are different and, hence, u does *not* deactivate v.

The routing policies of a network are *static* if they do not change with time.

Theorem 1 *With static routing policies, ESS-BGP is immune to false positives.*

Proof: Because routing policies are static, any two candidate routes at a node, (α, P) and (β, Q), that contain the same path, $Q = P$, must contain the same cost, $\beta = \alpha$.

Suppose that some node u elects route (α, P), which propagates around some cycle C before being learned back at u as route $(T[uCu](\alpha), CuP)$. Let (β, Q) be the best candidate route at u learned from outside C just before $(T[uCu](\alpha), CuP)$ is learned from around C. With ESS-BGP, u deactivates its neighbor around C if and only if $T[uCu](\alpha) \prec \beta$ and $Q = P$. Since routing policies are static, $Q = P$ implies $\beta = \alpha$. Therefore, condition $T[uCu](\alpha) \prec \beta$ writes as $T[uCu](\alpha) \prec \alpha$, implying that cost-circuit (α, uCu) is not absorbent: there is no false positive. ∎

Theorem 1 implies that, under static routing policies, if all cost-circuits are absorbent, then ESS-BGP has the same behavior as BGP. The stabilizing effect of ESS-BGP is felt only when routing policies make BGP prone to oscillations, as it should. Since the timescale at which routing policies change is coarser than the timescale at which routing messages propagate in the network, ESS-BGP ought not to suffer from false positives in practice. Contrary to the case of SS-BGP, we do not have proof that ESS-BGP terminates under isotone routing policies, but it did stabilize in all our simulations (see Section VI).

IV. Termination of SS-BGP and isotonicity

In Section IV-A, we present a proof that isotonicity implies termination of SS-BGP and in Section IV-B, we show through an example that, without isotonicity, SS-BGP may not terminate.

A. Proof of termination

We consider a fixed destination in a network. Routing messages are delivered across any given link in first-in-first-out order with arbitrary, but finite delay.

Lemma 1 *Given an arbitrary initial state, the set of all routes in all possible executions of SS-BGP is finite.*

Proof: An elected or candidate route (β, Q) at v, or a routing message (β, vQ) in the initial state can only give rise to $(\bullet, \bullet)$ or to routes of the form $(T[uPv](\beta), PvQ)$, where uPv is a path in the network and $T[uPv]$ denotes the composition of the extenders of all links of uPv. Since the number of paths in the network is finite, the set of all routes in all possible executions is finite. ∎

Lemma 2 *An update of elected route at a node to one with the same or a better cost gives rise, at an in-neighbor, to one of three conditions: (i) no change of elected route; (ii) an update of elected route to one with the same or a better cost; or (iii) the detection of an incongruous cost.*

Proof: Suppose that v updates its elected route from (β, Q) to (β', Q'), with $\beta' \preceq \beta$, sending routing message (β', vQ') to its in-neighbor u. Let: (i) (α^-, P^-) be the best route at u, to the exception of the candidate route learned from v, just before reception of routing message (β', vQ'); (ii) (α, P) be the elected route at u just before reception of routing message (β', vQ'); (iii) and (α', P') be the elected route at u after reception of routing message (β', vQ'). We want to show that either $\alpha' \preceq \alpha$ or $u \in vQ'$ and $T[uv](\beta') \prec \alpha^-$.

We divide the proof into four mutually exclusive cases, conditioning on the presence of u on paths vQ and vQ'.

- $u \notin vQ'$ **and** $u \notin vQ$**:** We have $\alpha' = T[uv](\beta') \sqcap \alpha^-$ and $\alpha = T[uv](\beta) \sqcap \alpha^-$. Because of isotonicity of link uv, $T[uv](\beta') \preceq T[uv](\beta)$. Thus, $\alpha' = T[uv](\beta') \sqcap \alpha^- \preceq T[uv](\beta) \sqcap \alpha^- = \alpha$.
- $u \notin vQ'$ **and** $u \in vQ$**:** We have $\alpha' = T[uv](\beta') \sqcap \alpha^- \preceq \alpha^- = \alpha$.
- $u \in vQ'$ **and** $u \notin vQ$**:** We have $\alpha' = \alpha^-$ and $\alpha = T[uv](\beta) \sqcap \alpha^-$. If $T[uv](\beta) \prec \alpha^-$, then, because of isotonicity of link uv, $T[uv](\beta') \preceq T[uv](\beta) \prec \alpha^-$, meaning that u detects an incongruous cost. Otherwise, if $T[uv](\beta) \succeq \alpha^-$, then $\alpha' = \alpha^- = \alpha$.
- $u \in vQ'$ **and** $u \in vQ$**:** We have $\alpha' = \alpha^- = \alpha$. (It may be that u also detects an incongruous cost in this case.)

∎

Lemma 3 *An execution of SS-BGP where no node ever detects an incongruous cost either terminates or there is an instant of time such that from hence every node updates its elected route to one with the same or a better cost.*

Proof: The proof is by contradiction. Suppose that we are given an infinite execution where no node ever detects an incongruous cost and that there is at least one node u that updates its elected route from some (α, P) to some (α', P'), with $\alpha' \succ \alpha$, infinitely often. Since, from Lemma 1, the set of all routes in the execution is finite, there is a pair (u, P), with u a node and P a path, such that u updates its elected route from some (α, P) to some (α', P'), with $\alpha' \succ \alpha$, infinitely often. Let (u, P) be such a pair with minimum P according to order $\lhd$. It cannot be the case that P is originated at u, since the cost of an elected route at u is never greater than the cost of an originated route. Hence, there is a neighbor v of u such that $P = vQ$ and v updates its elected route from some (β, Q) to some (β', Q') infinitely often, giving rise to the updates at u. Since (u, P) was chosen to have minimum P such that u updates its elected route from some (α, P) to some (α', P'), with $\alpha' \succ \alpha$, infinitely often, the elected route at v is updated from some (β, Q) to some (β', Q'), with $\beta' \succ \beta$, only finitely many times. Therefore, there is an update of elected route at v from some (β, Q) to some (β', Q'), with $\beta' \preceq \beta$, which gives rise, when routing message (β', vQ') arrives at u, to an update of elected route from (α, P) to (α', P'), with $\alpha' \succ \alpha$. Since

u never detects an incongruous cost, we have a contradiction from Lemma 2. ■

Lemma 4 *An execution of SS-BGP where no node ever detects an incongruous cost and every node updates its elected route to one with the same cost terminates.*

Proof: We present a function F from the state of the protocol to a well-founded set, which decreases with the reception of every routing message that preserves the cost of elected route at the receiving node. From Lemma 1, we deduce the finiteness of the set of all paths appearing in routes of all executions where every node updates its elected route to one with the same cost. For such a set of paths, let the rank of path P, denoted by $r(P)$, be the ordinal number of P according to order $\lhd$, with the empty path having rank 0. If $P \lhd Q$, then $r(P) < r(Q)$. Let $n-1$ be the rank of a path of maximum rank.

The range of F is the set of n-tuples of non-negative integers ordered lexicographically. The i-th coordinate of F, denoted by F_i, counts: (i) the number of elected routes (α, P) such that $r(P) = i$; plus (ii) the number of routing messages (α, uP) in transit such that $r(P) = i$. Suppose that u receives routing message (β', vQ') from v. Let (α, P) and (α, P') be the elected routes at u just before and after routing message (β', vQ') is received, respectively. Further, let $r(Q') = i$, $r(P) = j$, and $r(P') = k$. We distinguish three mutually exclusive cases.

$P' = P$: F_i decreases by one, so that F decreases.

$P' = vQ'$: F_i decreases by one, F_j decreases by one, F_k increases by one plus the number of in-neighbors of u. Since $P' = vQ'$, we have $P' \lhd Q'$. Thus, $i < k$, so that F decreases.

$P' \neq P$ and $P' \neq vQ'$: F_i decreases by one, F_j decreases by one, F_k increases by one plus the number of in-neighbors of u. It must be that case that route (α, P) was learned from v. Therefore, $P = vQ$. Since the cost of the elected route at u is the same just before and after the routing message is received, we must have $P \lhd P'$: otherwise, u would have elected route (α, P') instead of route (α, P). Thus, $j < k$, so that F decreases.

■

Theorem 2 *An execution of SS-BGP where no node ever detects an incongruous cost terminates.*

Proof: From Lemma 3, either the execution terminates or there is an instant of time t such that from hence every node updates its elected route to one with the same or a better cost. From Lemma 1, we deduce that the set of all costs appearing in routes of all executions is finite. Therefore, the cost of elected route at every node cannot improve indefinitely. Either the execution terminates or there is an instant of time t', $t' > t$, such that from hence every node updates its elected route to one with the same cost. From Lemma 4, we conclude that the execution terminates. ■

Theorem 3 *SS-BGP terminates.*

Proof: The proof is by contradiction. Suppose that we are given an infinite execution. The number of detections of incongruous costs is finite, because every detection deactivates an out-neighbor and the number of out-neighbors is finite. But then, from a certain time onwards, no node ever detects an incongruous cost, which contradicts Theorem 2. ■

B. When isotonicity does not hold

We go back to the model of Section II-A, but instead of assuming that peer$^+$ routes are exported to all neighbors, we now assume that these routes are exported *only* to customers, as is the case with standard peer routes. As a consequence, the routing policies of a peer$^+$ link uv are not isotone: we have $r^+ \prec c$, but $T[uv](r^+) = \bullet \succ r^+ = T[uv](c)$. Consider the network of Figure 1 with the new assumption regarding peer$^+$ routes. At $T = 1$, all of u, v, and w elect the peer$^+$ route learned from their clockwise neighbor. They do not export these elected routes around the cycle—rather, they withdraw the routes that they had previously advertised. Hence, at $T = 2$, all of u, v, and w elect route (c, x), as they did at $T = 0$. BGP oscillates forever without a trace of routing loops. Because the oscillatory behavior is not made from routing loops, SS-BGP and ESS-BGP behave as BGP, oscillating forever.

However, isotonicity of the routing policies of all links is not necessary for termination of SS-BGP or ESS-BGP. In Figure 1, if, for instance, v and w export peer$^+$ routes to all neighbors, but u exports them only to customers, then u will deactivate v at $T = 3$, a condition which stabilizes the protocol.

V. On isotonicity

Isotonicity brings benefits to inter-AS routing. First, as this paper shows, isotonicity guarantees termination of SS-BGP. Second, isotonicity guarantees visibility of the relevant routing information to every node. Conversely, without isotonicity, BGP may hide relevant routing information from some nodes. In extreme cases, a node may fail to elect a route to reach a destination, despite the existence of a path from the node to the destination allowed by the routing policies. Consider the network of Figure 1 without links wu and ux, and with peer$^+$ routes exported only to customers (see Section IV-B). Nodes w and v elect, respectively, routes (c, x) and (r^+, wx) to reach x. Because peer$^+$ routes are not exported across a peer$^+$ link, u does not elect a route to reach x. However, there is a path from u to x, path uvx, that is allowed by the routing policies.

Apart from the routing policies examined in this paper, little is known about the prevalence of isotonicity in inter-AS routing practices. Gill et al. [9] surveyed approximately 100 AS administrators (out of a universe of tens of thousands) about their routing policies. The survey does not inquire explicitly about isotonicity. But, it inquires about next-hop routing policies, asking AS administrators whether the cost of a route depends exclusively on the AS from which the route was learned. Gill et al. estimate that at least 68% of the ASes

use next-hop routing policies. We observe that with next-hop routing policies, isotonicity can be ensured solely through local regulation of export policies, without coordination between nodes. Suppose that node u uses next-hop routing policies in regard to routes learned from its neighbor v. The routing policies of link uv are isotone if and only if the routes that v exports to u are all better than the routes it does not export to u [15].

VI. Evaluation

We ran simulations of BGP, SS-BGP, and ESS-BGP over realistic topologies of the Internet with realistic routing policies. Section VI-A describes the simulator and the input networks. Section VI-B compares the stability properties of BGP with those of SS-BGP and ESS-BGP, and Section VI-C discusses partial deployment of SS-BGP.

A. Simulator and input networks

We developed our own discrete-event simulator.[4] In the simulator, routing messages traveling across a link are subjected to a random delay drawn from a uniform distribution in the interval from 0.01 to 1 seconds, and delivered in a first-in-first-out order. The announcements and withdrawals of routes at each AS are paced by the Minimum Route Advertisement Interval (MRAI) [1], for which we took the reference value of 5 seconds.

The input networks to the simulator start off with AS-level topologies of the Internet made available by CAIDA [16]. These topologies list pairs of ASes annotated with either a provider-customer or a peer-peer relationship. CAIDA's AS-level topologies are updated monthly. We report results for the topology from July 1st, 2016. We first removed the few ASes from the AS-level topology that could not reach other ASes by a customer, a peer, or a provider route. Out of $54,733$ ASes, 786 were removed.

Tier-ones (14 of them) are ASes without providers, whereas stubs ($45,914$ of them) are ASes without customers. They stand, respectively, at the top and bottom of the provider-customer hierarchy. Stubs produce and consume data-packets, but they do not transit them: they do not export to neighbor ASes routes learned from other neighbor ASes. Therefore, stubs never contribute to permanent oscillations of the state of BGP. In order to save on simulation time, stubs were removed from the initial topology, while the origination of their routes was delegated to their providers. We ended up with a digraph containing $8,033$ ASs, $21,854$ customer links, that same number of provider links, and $134,182$ peer links.

The GR routing policies serve as benchmark. Assuming, as we do, that ties among customer (peer, provider) routes are broken by the lengths of the paths carried in those routes, BGP terminates under these routing policies. We perturb the GR routing policies in two ways, each of which makes BGP prone to oscillations.

- Substitution of some peer links by peer$^+$ links, with peer$^+$ routes exported to all neighbors (see Section II-A). Peer$^+$ links are chosen randomly from the set of all peer links according to a uniform distribution. We present results for three percentages of peer$^+$ links: 1%, 5%, and 10%.
- Substitution of some links by sibling links (see Section II-B). We resorted to CAIDA's Inferred AS to Organizations Mapping Dataset [17] to identify pairs of ASes that belong to the same organization. The two links between each of these pairs were replaced by sibling links. We ended up with $4,038$ sibling links, which amount to 3% of the peer links.

[4] https://github.com/davidfialho14/bgp-simulator.

B. Stability results

The basic experiment is a simulation run, in which a destination AS originates a route that is propagated network-wide according to the rules of the routing protocol subject to the routing policies in effect. The termination time of an AS in a simulation run is the time it takes for its elected route to stabilize. For any given destination, we executed 100 simulation runs with different seeds for generating random delays across links. The termination time of the routing protocol for that destination is the average of the termination times of all ASes over all 100 simulation runs. BGP may fail to terminate in the presence of peer$^+$s or siblings. Given a destination, if at least one AS in at least one simulation run of BGP has a termination time greater than 300 times the termination time of BGP with the GR routing policies, then we declare non-termination for that destination.

Non-termination of BGP. The mere presence of peer$^+$s and siblings does not automatically create non-absorbent cost-circuits. Moreover, we recall from Section III-B that the presence non-absorbent cost-circuits does not imply non-termination of BGP for all destinations—it only implies non-termination for at least one destination. We first assess the impact of peer$^+$s and siblings on the stability of BGP. Table I shows the percentage of destinations from among 200 destinations chosen randomly with a uniform distribution for which BGP does not terminate, subject to the GR routing policies with peer$^+$s and the GR routing policies with siblings. Each percentage of peer$^+$s represents an average over five samples. The following conclusions can be drawn.

- Peer$^+$s are much likelier to induce oscillations of the state of BGP than siblings. The percentage of destinations for which BGP does not terminate is 62.8%, 46.7%, and 36.6%, respectively, for 1%, 5%, and 10% of peer$^+$ links, whereas the percentage of destinations for which BGP does not terminate is only 2%, for 3% of sibling links.
- Interestingly, the stability of BGP increases with the percentage of peer$^+$s. Non-absorbent cost-circuits come in different lengths, measured in terms of the number of links they contain. Considered in isolation, the likelihood of reaching a stable state is higher in shorter non-absorbent cost-circuits than in longer ones. As the percentage of peer$^+$s increases so does the presence of short non-absorbent cost-circuits. ASes belonging to these loops are likely to see their elected routes stabilize,

despite simultaneous membership of those ASes in long non-absorbent cost-circuits.

TABLE I
PERCENTAGE OF DESTINATIONS FOR WHICH BGP DOES NOT TERMINATE.

Policies	Non-termination
Siblings (3%)	2.0%
Peer$^+$ (1%)	61.8%
Peer$^+$ (5%)	45.8%
Peer$^+$ (10%)	35.6%

Detection of incongruous costs. Table II shows the number of detections of incongruous costs that it takes SS-BGP and ESS-BGP to terminate, averaged over the 200 destinations. The difference in detections between SS-BGP and ESS-BGP is justified by false positives, which occur with the former detection method and are absent in the latter. The interesting conclusion to be taken from these results is that only very few detections are needed for stabilization. For instance, at 5% of peer$^+$s, ESS-BGP terminates after 10.3 detections, corresponding to 0.006% of all links in the input network. The number of detections equals the number of deactivations of neighbor ASes and, thus, equals the number of import policies that are altered. Hence, the results show that SS-BGP and ESS-BGP stabilize BGP at the expense of affecting the import routing policies of only very few ASes.

TABLE II
NUMBER OF INCONGRUOUS COSTS DETECTED.

	Detections	
Policies	**SS-BGP**	**ESS-BGP**
Siblings (3%)	1.2	1.0
Peer$^+$ (1%)	5.7	4.3
Peer$^+$ (5%)	23.8	10.3
Peer$^+$ (10%)	64.4	17.8

Termination times. Table III compares the termination time of BGP, for those destinations for which the protocol terminates, to the termination time of SS-BGP. The termination time of ESS-BGP is very similar to that of SS-BGP and is omitted. We conclude that even for destinations for which BGP terminates, SS-BGP terminates faster. For instance, at 5% of peer$^+$s, BGP takes 37.0 seconds to terminate, while SS-BGP takes 25.6 seconds to terminate, a reduction of 30.8%.

TABLE III
TERMINATION TIME OF BGP, FOR WHEN IT TERMINATES, AND OF SS-BGP.

Termination (s.)			
Policies	**BGP**	**SS-BGP**	**Reduction**
GR	7.3	7.3	0%
Siblings (3%)	13.9	13.7	1.4%
Peer$^+$ (1%)	29.5	21.7	25.6%
Peer$^+$ (5%)	37.0	25.6	30.8%
Peer$^+$ (10%)	37.4	26.8	28.3%

C. Partial deployment

Termination of SS-BGP for all possible destinations in a given network does not require deployment of SS-BGP at every node. Termination is ensured if all nodes that belong to at least one non-absorbent cost-circuit deploy SS-BGP. We investigate this condition for the case of the GR routing policies with peer$^+$s, which are likelier than siblings to produce oscillations. In this case, it can be verified from the definition that a node belongs to a non-absorbent cost-circuit *if and only if* it is at the tail of a peer$^+$ link and that link belongs to a cycle all links of which are peer$^+$ links or customer links.

Table IV, under the heading 'Loop', presents the percentage of ASes that belong to a non-absorbent cost-circuit in the input network (from where stubs were removed). We conclude that the deployment of SS-BGP on selected 7.8% of the ASes is sufficient for termination at 1% of peer$^+$ links. The percentage of ASes that need to deploy SS-BGP increases with the percentage of peer$^+$ links, reaching 28.0% at 10% of peer$^+$ links.

TABLE IV
PERCENTAGE OF ASES THAT BELONG TO A NON-ABSORBENT COST-CIRCUIT AND THAT ARE AT THE TAIL OF A PEER$^+$ LINK.

	Loop	Tail
Peer$^+$ (1%)	7.8%	8.5%
Peer$^+$ (5%)	21.4%	21.5%
Peer$^+$ (10%)	28.0%	28.2%

An AS cannot tell from its routing configuration whether or not it belongs to a non-absorbent cost-circuit. However, it can tell from its routing configuration whether or not it has a link to a peer$^+$, which condition is necessary for membership in a non-absorbent cost-circuit: if all ASes with a link to a peer$^+$ deploy SS-BGP, then termination is guaranteed. This conclusion prompts a simple guideline for AS administrators, stating that they can autonomously violate the GR routing policies by preferring some peers over some customers as long as SS-BGP is deployed in their networks. Table IV, under the heading 'Tail', presents the percentage of ASes at the tail of at least one peer$^+$ link, showing that these percentages are close to those of the ASes that belong to non-absorbent cost-circuits.

VII. RELATED WORK

The problem of BGP state oscillations caused by incompatible routing policies is not new. However, none of the solutions proposed to date is satisfactory. We survey related work divided into three main groups.

Modifications to BGP. Griffin and Wilfong [18], Ahronovitz et al. [19], Ee et al. [20], and Agarwal et al. [21], all propose solutions to stabilize BGP that work with routes that were elected in the past and no longer are. In [18], an elected route is stored and advertised along with its history, which records the sequence of past routes that justifies the elected route. Histories with repeated routes uncover incompatible routing policies and serve to identify routes that should be filtered to arrive at a stable state. In [19], each node keeps a list of routes that were elected in the recent past. When an elected route is repeated in the list, the node launches a token which, on arriving back, confirms an incompatibility and identifies a route to be filtered. In [20], each node likewise

keeps a list of routes that were elected in the recent past. Routes carry a global precedence value which is the first parameter consulted in the election of a route—the smaller the better. The global precedence value is incremented each time a node elects a route which is worse than one of the old routes in its list. In [21], detection of conflicting routing policies is based on [18], but rather than filtering routes to avoid future oscillations, multiple routes are sent to neighbor nodes. Albeit in different degrees, all the proposals above involve taxing modifications to BGP and require network-wide deployment to earn their benefits. In contrast, SS-BGP is an extension to BGP, and it promotes stability even during partial deployment.

Policy restrictions that ensure termination of BGP. The class of inflationary routing policies guarantees termination of BGP [5], [22], [23]. The routing policies of link uv are inflationary if a route at v gives rise to a route at u with the same or a worse cost. However, inflation is an artificial property insofar as inter-AS routing is concerned, because it implies a comparison between two costs held at different nodes. Inflation does not hold for the GR routing policies with peer$^+$, with siblings, or if customers or providers are divided into distinct classes of preference. In contrast, isotonicity does not have the artificiality of inflation, being satisfied by baseline inter-AS routing policies.

Policy guidelines with cycles to avoid. Several works present guidelines for the configuration of BGP accompanied by an identification of cycles that should be avoided in any given network [6], [13]. For example, the GR routing policies exclude cycles where every node is a customer (a provider) of the next node around the cycle. Guidelines provide broad insights into routing. However, they smooth over much diversity found in practice. In addition, they do not come with a mechanism to avoid the identified cycles of incompatible routing policies during an execution of BGP. Isotonicity allows for a wide range of routing policies, with SS-BGP resolving policy incompatibilities at run time.

VIII. Conclusions

SS-BGP is an extension to BGP that prevents recurrent routing loops, thereby attaining better stability properties than BGP. For the important class of isotone routing policies, SS-BGP terminates.

We conducted simulations of BGP and of SS-BGP on Internet topologies with variants of the GR routing policies. We verified that, contrary to BGP, SS-BGP reaches a stable state in all cases, with alterations in the import routing policies of only very few nodes. As a matter of fact, SS-BGP needs to be deployed only in the ASes that do not comply with the GR routing policies in order for its stabilizing effects to have network-wide impact.

Acknowledgments

We thank José Brázio, Franck Le, and Jennifer Rexford for commenting on earlier drafts of this work, and we thank our shepherd, Marco Chiesa, for a stimulating interaction, positively reflected in the final paper. We acknowledge the support of Fundação para a Ciência e Tecnologia under grant UID/EEA/50008/2013.

References

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

[2] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, no. 1, pp. 1–16, 2000.

[3] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 232–243, April 2002.

[4] M. G. Gouda and M. Schneider, "Maximizable routing metrics," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 663–675, August 2003.

[5] J. L. Sobrinho, "An algebraic theory of dynamic network routing," *IEEE/ACM Transactions on Networking*, vol. 13, no. 5, pp. 1160–1173, October 2005.

[6] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 681–692, December 2001.

[7] V. Giotsas, M. J. Luckie, B. Huffaker, and kc claffy, "Inferring complex AS relationships," in *Proc. ACM Internet Measurement Conference*, November 2014, pp. 23–30.

[8] R. Mazloum, M. Buob, J. Augé, B. Baynat, D. Rossi, and T. Friedman, "Violation of interdomain routing assumptions," in *Proc. Passive and Active Measurement Conference*, March 2014, pp. 173–182.

[9] P. Gill, M. Schapira, and S. Goldberg, "A survey of interdomain routing policies," *ACM SIGCOMM Computer Communications Review*, vol. 44, no. 1, pp. 28–34, January 2014.

[10] R. Anwar, H. Niaz, D. R. Choffnes, Í. S. Cunha, P. Gill, and E. Katz-Bassett, "Investigating interdomain routing policies in the wild," in *Proc. ACM Internet Measurement Conference*, October 2015, pp. 71–77.

[11] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration," in *Proc. ACM SIGCOMM 2002*, August 2002, pp. 3–16.

[12] S. Deshpande, M. Thottan, and B. Sikdar, "An online scheme for the isolation of BGP misconfiguration errors," *IEEE Tranactions on Network and Service Management*, vol. 5, no. 2, pp. 78–90, 2008.

[13] Y. Liao, L. Gao, R. Guérin, and Z.-L. Zhang, "Safe interdomain routing under diverse commercial agreements," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1829–1840, December 2010.

[14] J. L. Sobrinho, "Correctness of routing vector protocols as a property of network cycles," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 150–163, February 2017.

[15] J. Feigenbaum, V. Ramachandran, and M. Schapira, "Incentive-compatible interdomain routing," in *Proc. ACM Conference on Electronic Commerce*, June 2006, pp. 130–139.

[16] "AS relationships," http://www.caida.org/data/as-relationships, July 2016, CAIDA.

[17] "Mapping autonomous systems to organizations," http://www.caida.org/research/topology/as2org/, July 2016, CAIDA.

[18] T. G. Griffin and G. Wilfong, "A safe path vector protocol," in *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, March 2000, pp. 490–499.

[19] E. Ahronovitz, J. König, and C. Saad, "A distributed method for dynamic resolution of BGP oscillations," in *Proc. International Parallel and Distributed Processing Symposium*, April 2006.

[20] C. T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker, "Resolving inter-domain policy disputes," in *Proc. ACM SIGCOMM*, August 2007, pp. 157–168.

[21] R. Agarwal, V. Jalaparti, M. Caesar, and P. B. Godfrey, "Guaranteeing BGP stability with a few extra paths," in *Proc. International Conference on Distributed Computing Systems*, June 2010, pp. 221–230.

[22] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of autonomy for the expressiveness of policy routing," *IEEE/ACM Transactions on Networking*, vol. 15, no. 6, pp. 1266–1279, December 2007.

[23] A. Wang, L. Jia, W. Zhou, Y. Ren, B. T. Loo, J. Rexford, V. Nigam, A. Scedrov, and C. Talcott, "FSR: Formal analysis and implementation toolkit for safe interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1814–1827, December 2012.