Interdomain Routing Without Instabilities

David Fialho

***Abstract*—The Internet is comprised of a large set of Autonomous Systems (ASs). Every AS participates in a single routing protocol, called Border Gateway Protocol (BGP), allowing them to learn paths to communicate with each other. BGP allows AS administrators to set a wide range of routing policies autonomously. However, it does not guarantee stability.**

This work investigates the relationship between routing policies and routing loops and the effect they have on BGP's ability to reach a stable state. In particular, this work focuses the stabilizing BGP for isotone routing policies. An addon to BGP is proposed, called Self-Stable BGP (SS-BGP), which guarantees termination with isotone routing policies. SS-BGP adds to BGP the ability (i) to distinguish between temporary routing loops and persistent routing loops, and (ii) to eliminate the latter. Gradually eliminating persistent routing loops ensures termination in the case isotone routing policies, and improves the stability of BGP in general.

The performance and effectiveness of SS-BGP are evaluated through simulation. For that purpose, a new discrete event simulator was developed able to simulate the dynamic behavior of BGP and SS-BGP in large-scale topologies, like the Internet topology, with a wide range of routing policies. A large number of simulations were conducted for both BGP and SS-BGP in realistic topologies of the Internet with realistic perturbations of the baseline interdomain routing policies. Results show that SS-BGP terminates in all cases, in contrast to BGP, with only a small percentage of ASs having to change their import policies.

I. Introduction

The Internet is comprised by a large collection of ASs. Each AS establishes commercial relationships with a few other ASs so as to attain global connectivity across the entire Internet. The process through which ASs discover paths to reach each destination is called routing. In the Internet, all ASs participate in a single interdomain routing protocol called BGP [1]. In BGP, nodes exchange routes. At an high-level BGP routes are composed of a cost, which is materialized principally by the LOCAL-PREF attribute of BGP; and a path, which is materialized by the AS-PATH attribute of BGP. Costs are determined by the routing policies set at each node and they impose a ranking among routes. Paths allow BGP to detect routing loops and stop the propagation of routing messages past those loops: a node discards a route the advertised path of which contains the identifier of the node.

BGP is a very flexible protocol that allows AS administrators to autonomously set a wide range of routing policies. Consequently, nodes can set conflicting routing policies that cause the state of the protocol to oscillate permanently [2][3]. Permanent oscillations have a negative impact on the Internet's control plane and data plane, since they attract data messages to forwarding loops, causing these to be delayed or even lost while traveling around those loops.

This work had as its starting point an embryonic idea to devise an extension to BGP able to detect routing loops that may cause oscillations and eliminate these loops from the network in order to improve the stability of BGP. After some research work, it was determined that there are two types of routing loops: temporary routing loops and persistent routing loops. Temporary routing loops occur a finite number of times over an execution of BGP. On the other hand, persistent routing loops may recur indefinitely in such an execution. This conclusion justifies chasing and eliminating the latter. An addon to BGP, called SS-BGP, is proposed in this thesis work, which devises a refinement to the routing loop detection method included in BGP, able to distinguish between temporary and persistent routing loops, and that eliminates them. The basic idea is simple: if an AS learns a route from a neighbor, but the AS-PATH of which identifies a routing loop, then that AS discards all future routes learned from that neighbor. SS-BGP uses standard BGP routing messages and, therefore, can be deployed incrementally.

For the common and broad class of routing policies characterized by the isotonicity property [5][6], all permanent state oscillations are associated with persistent routing loops. SS-BGP is guaranteed to terminate for these routing policies, that is, BGP reaches a stable state whatever the initial state and whatever the delays in the delivery of routing messages. When routing policies are not isotone, SS-BGP still brings benefits to the protocol's stability, with SS-BGP being able to reach a stable state when BGP is not.

The dynamic behavior of BGP and SS-BGP were evaluated by simulation. The simulation of these protocols in large-scale networks requires low-level details to be abstracted so that they do not consume unnecessary resources. In the thesis work, a new discrete-event simulator was developed from scratch, which implements an high-level version of BGP and SS-BGP, maintaining all the important aspects required for the purposes of this work. This approach allowed a large number of simulations to be performed with realistic topologies of the Internet with limited time and resources.

II. Elimination of Oscillations

This Section introduces SS-BGP through three examples. The first example, Section II-A, relates to shortest-path routing and illustrates the distributed detection and elimination of oscillations. The next two examples, Sections II-B and II-C, relate to interdomain routing and they highlight the importance of isotonicity in ensuring termination, and the possibility of termination even in partial deployment or when not all routing policies are isotone.

A. Shortest-path routing with negative costs

This Section starts with an example of shortest path routing because this is the most familiar form of routing, thus serving

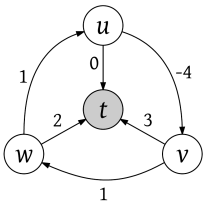

Fig. 1: Nodes u and v detect a routing loop and discard the routes learned from their clockwise neighbor. The state of BGP oscillates forever while SS-BGP terminates. Integers represent additive costs and the destination is announced by t.

to illustrate how the problem of permanent state oscillations can be mitigated resorting strictly to the information available at each node. BGP routes are couplets of the form (l, P), where l is an integer number representing a cost and P is a path. For any destination, every node stores a candidate route learned from each one of its neighbors and elects the best among those routes. A route is better than another if (i) its cost is better, that is, smaller, or (ii) the costs are the same and the length of its path is shorter. The route that a node learns from a neighbor is obtained by (i) adding the cost of the link to the neighbor to the cost of the route elected at the neighbor, and by (ii) prefixing the neighbor to the path of the elected route. A learned route that identifies a routing loop is immediately discarded.

With respect to shortest path routing, it is well established that as long as the costs of links are non-negative, BGP always terminates [4]. However, the use of links with negative costs allows network operators to force traffic to follow some paths over others without having to change the costs of most links in the network. For instance, the operator of the network presented in Figure 1 wants to force traffic to flow through link uv instead of link ut because link ut is congested. Therefore, it sets the cost of uv to -4, forcing u to prefer path uvt to reach t over path ut.

Assume t announces a new destination at $T = 0$. Thus, at $T = 1$, nodes u, v, and w elect, respectively, routes $(0, t)$, $(3, t)$, and export their respective elected routes counter-clockwise around the cycle. At $T = 2$, u elects route $(-1, vt)$ learned from v; v learns route $(3, wt)$, but still elects route $(3, t)$ since the latter corresponds to a shorter path; and w elects route $(1, ut)$ learned from u. The newly elected routes by u and w are exported counter-clockwise around the cycle. At $T = 3$, v elects route $(2, wut)$ learned from w; and w elects route $(0, uvt)$ learned from u. They both export their respective newly elected route counter-clockwise around the cycle. Hence, at $T = 4$, u learns route $(-2, vwut)$ from v, but path $vwut$ identifies a loop. Therefore, although u prefers this route over its alternative route via t, it must discard the new route and elect route $(0, t)$ instead; similar to u, node v also discards route $(1, wuvt)$ and elects the worse route $(3, t)$; The routes elected by u and v are, once again, exported counter-clockwise around the cycle. As a consequence, at $T = 5$, the protocol returns to the same state it was at $T = 2$, with nodes u, v, and w electing, respectively, routes $(-1, vt)$, $(3, t)$, and $(1, ut)$. From now on, the state of BGP will repeatedly evolve through the same sequence of states observed from $T = 2$ to $T = 5$, without any of the nodes ever sticking to a single route.

However, an argument can be made that, at $T = 4$, both u and v detect that something is amiss, in that, they both discard their best route to the destination because these routes identify a loop. With knowledge of the global network topology, it can be identified that the cycle $uvwu$ has cost -2 ($-2 = -4 + 1 + 1$). This fact justifies routes being dispatched around the cycle and arriving back at the original node with a lower cost than the one with which they were originally exported. Based on this observations, 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 the destination, but the path component of the route indicates a routing loop, then that neighbor is deactivated.

Deactivating a neighbor corresponds discarding all routes learned from that neighbor. Saying that a node u deactivated a link uv is the same as saying that u deactivated v.

SS-BGP evolves similarly to BGP in the network of Figure 1 up until $T = 4$. At that point, u and v detect a conflict and they deactivate their respective neighbors v and w, discarding all future routes coming from these nodes. As a result, at $T = 5$, in contrast to BGP, u does not import route $(-1, vt)$ learned from v, keeping its election of route $(0, t)$. Similarly, at $T = 6$, v also does not import route $(-1, wut)$ learned from w and keeps electing route $(3, t)$. As a consequence, the protocol stabilizes with u electing route $(0, t)$, v electing route $(3, t)$, and w electing route $(1, ut)$ to reach the destination.

Notice links uv and wv were both deactivated. However, deactivating just one of them would eliminate the routing loop and, thereby, ensure the protocol would reach a stable state. SS-BGP performs a distributed detection and elimination of oscillations and, therefore, it is susceptible to race conditions as verified in the example of Figure 1. This example also suggests that not all nodes are required to use SS-BGP to ensure termination: if just u or v deploy SS-BGP, then the protocol still terminates. However, if only w deploys SS-BGP, then the protocol can still oscillate. Partial deployment is discussed further in Section IV-E.

B. Interdomain routing with siblings

The problem of permanent state oscillations is now presented in the context of interdomain routing. The Gao-Rexford (GR) routing policies provide a baseline of understanding for how AS administrators set their routing policies [10]. These policies consider two types of commercial relationships between neighboring ASs: provider-customer and peer-peer. A link that joins a provider to a customer is called a customer link. Similarly, a provider link joins a customer to a provider and a peer link joins a peer to another peer. Routes are of three types: 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

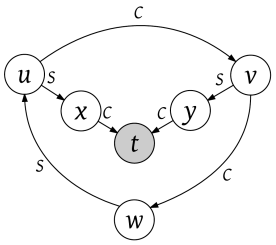

Fig. 2: In contrast to BGP, SS-BGP terminates with the GR routing policies with siblings. S and C represent sibling and customer links, respectively, and the destination is announced by t.

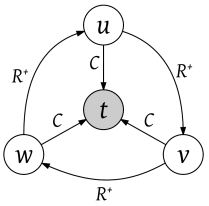

Fig. 3: SS-BGP does not terminate if peer+ routes are exported only to customer. If peer+ routes are exported to all neighbors, then SS-BGP terminates. R^+ and C represent peer+ and customer links, respectively, and the destination is announced by t.

customer and customer routes are exported to all neighbors, these being the only exportations allowed.

Liao *et al.* [11] proposes the additional sibling-sibling relationship to be established between ASs belonging to the same organization. Siblings are neighbors that share all routes between them. A sibling link joins a sibling to another sibling. 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.

Given a destination, BGP routes are of the form (α, n, P), where (α, n) is the cost and P the path. In cost (α, n), α is one of c, r, and p, representing the type of route learned from outside the stretch of siblings, respectively, customer, peer, and provider, and n is a natural number representing the number of consecutive sibling hops traversed by the route. Cost (α, n) is better than another 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, links are labeled with C or S, corresponding to a customer and sibling link, respectively. Nodes x and y are providers of t and they are, respectively, siblings of u and v. Furthermore, around the cycle $uvwu$, v is a customer of u; w is a customer of v; and u is a sibling of w. Node t announces the destination. At $T = 1$, both x and y elect a route $(c, 0, t)$ and export this route to their respective siblings u and v. At $T = 2$, u elects the route $(c, 1, xt)$ learned from x; and v elects the route $(c, 1, yt)$ learned from y. Both nodes export its elected route counter-clockwise around the cycle. At $T = 3$, u learns the customer route $(c, 0, vyt)$ from v and elects this route over the customer route with one sibling hop learned from x; and w elects route $(c, 2, uxt)$ learned from x. The routes elected by u and w are then exported counter-clockwise around the cycle. At $T = 4$, v elects a customer route with no sibling hops $(c, 0, wuxt)$ learned from w, which is better than the customer route with one sibling hop learned from y; and w elects the route $(c, 1, uvyt)$ learned from u. Once again, nodes v and w export their elected routes to their respective counter-clockwise neighbors. Then, at $T = 5$, nodes u and v learn routes $(c, 0, vwuxt)$ and $(c, 0, wuvyt)$, respectively. Both nodes prefer its new customer route with no sibling hops over its customer route with one sibling hop learned from outside the cycle. However, the new routes identify a loop and, therefore, must be discarded. As a consequence, nodes u and v elect, respectively, routes $(c, 1, xt)$ and $(c, 1, yt)$. Hence, the protocol has reverted back to the state it was on at $T = 2$, implying permanent state oscillations.

However, the checking condition included in SS-BGP is verified, once again, by u and v at $T = 5$: they learn a route which simultaneously is its best route and identifies a routing loop. Which means, that if they both use SS-BGP, then, at $T = 5$, u deactivates its neighbor v, and v deactivates w. Consequently, in the next unit of time, u discards route $(c, 0, vt)$ learned from its deactivated neighbor v, keeping its election of route $(c, 1, xt)$ learned from x. Therefore, in contrast to BGP, SS-BGP terminates with u electing route $(c, 1, xt)$, v electing route $(c, 1, yt)$, and w electing route $(c, 2, uxt)$ to reach the destination announced by t.

Serves this example to illustrate that the methods to detect and eliminate oscillations included in SS-BGP apply both to shortest path routing and interdomain routing.

C. Interdomain routing with peer+s

There is compelling evidence that the baseline GR routing policies are consistently violated by AS administrators for various reasons [12]. For instance, some ASs sometimes prefer peer routes over customers routes perhaps the link to the customer is congested or deemed unreliable. A peer+ is a peer the routes learned from which are preferred to customer routes and such routes are called peer+ routes. A link joining a peer/peer+ to a peer+ is called a peer+ link.

For a given destination, BGP 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. Two different export rules can be defined for peer+ routes: either peer+ routes are exported strictly to customers or they are exported to all neighbors.

In the network of Figure 3, links are labeled with C or R^+, corresponding to a customer and peer+ link, respectively. Node v is a peer+ of u; w is a peer+ of v; and u is a peer+ of w. Node t announces the destination at $T = 0$. At $T = 1$, each provider of t elects a route (c, t) and exports it counter-clockwise around the cycle $uvwu$. At $T = 2$, each of u, v, and w elects a peer+ route learned from their clockwise neighbor. Assuming peer+ are exported only to customers, then in they all inform their respective counter-clockwise neighbors that they no longer hold a route to reach the destination (although, they do). Consequently, at $T = 3$, nodes u, v, and w revert back to elect route (c, t) they each previously learned from t. Which means, the global state of the protocol also reverted

back to the state it was on at $T = 1$, implying a permanent state oscillations.

Assume that t announces a new destination again, but this time assume peer$^+$ routes are exported to all neighbors. As before, at $T = 2$, each of, u, v, and w, elects a peer$^+$ route learned from its clockwise neighbor. But now, they export their elected peer$^+$ route to their counter-clockwise neighbors. Consequently, at $T = 3$, u elects route (r^+, vwt) learned from v; v elects route (r^+, wut) learned from w; and w elects route (r^+, uvt) learned from v. Once again, the new routes elected routes by u, v, and w are exported counter-clockwise around the cycle. At $T = 4$, u learns route $(r^+, vwut)$ from v; v learns route $(r^+, wuvt)$ from w; and w learns route $(r^+, vwut)$ from u. Each of the, by inspecting the path of its new learned route, realizes that the route identifies a loop. With BGP, these routes are immediately discarded leaving the protocol to oscillate forever. However, with SS-BGP, each of u, v, and w detects that the cost r^+ of the route that denounces a loop is better than the cost c of its direct route to t and, consequently, deactivates its clockwise neighbor. Thus, at $T = 4$, SS-BGP reaches a stable state every node electing route (c, t).

The reason why SS-BGP terminated in the network of Figures 1 and 2, and in the network of Figure 3 when peer$^+$ were exported to all neighbors, has to do with the isotonicity property which is verified in all of these cases. Isotonicity [5] means that the relative preference between any two costs at a node is not inverted between the costs learned from them at a neighbor node. In [13], it is shown that isotonicity ensures termination of SS-BGP.

The example of Figure 3 can also be used to show that, in some cases, SS-BGP still terminates even when not all nodes apply isotone routing policies: if only one of the nodes does export peer$^+$ routes to all neighbors, but the other two do so, then SS-BGP still terminates. For instance, consider u is the only one which does not export peer$^+$ routes to peer$^+$s. In that case, u still detects a routing loop and deactivates v at $T = 4$, which is sufficient to ensure SS-BGP reaches a stable state.

III. BGP Routing Model

A route associates a set of attributes to a destination. A destination is not a node, but an address that can be announced by any node in the network. BGP supports a large set of attributes to characterize routes. The first two attributes in BGP's election process, and therefore the most relevant ones, are the LOCAL-PREF and the AS-PATH. The former defines the degree of preference locally assigned to the route by a node. This attribute is local to each node and is not included in the exchanged routing messages. The AS-PATH contains the sequence of nodes traversed by the route, from the origin node to the current node holding the route. The AS-PATH is specially relevant in BGP, because it allows nodes to detect routing loops, avoiding the election of routes that traverse the same node more than once.

Rather than using LOCAL-PREF explicitly, the routing model presented here characterizes BGP routes as couplets of the form (α, P), where α is a cost, and P is a path. The finite set Σ contains all possible costs that can be assigned to a route. The operation $\prec$ establishes an order among the costs in Σ. For two costs $\alpha, \beta \in \Sigma$, if $\alpha \prec \beta$, then cost α is *better* than cost β, and cost β is *worse* than cost α. Writing $\alpha \preceq \beta$, means that either $\alpha \prec \beta$, or $\alpha = \beta$. Writing $\alpha \succ \beta$ is equivalent to $\beta \prec \alpha$. Cost $\bullet$ represents unreachability: a node has no knowledge of a path to the destination; thus cost $\bullet$ is than any cost, $\forall_{\alpha \in \Sigma}\ \alpha \prec \bullet$.

The election among costs is modeled by the binary operation $\sqcap$. With two costs α, β, $\alpha \sqcap \beta = \alpha$ if $\alpha \preceq \beta$, and $\alpha \sqcap \beta = \beta$ if $\alpha \succeq \beta$. For a set of costs, the elected cost is represented by $\sqcap\{\alpha_1, ..., \alpha_n\}$.

Paths are totally ordered by $\ll$. Order $\ll$ respects first and foremost the path lengths, such that, given two paths P and Q,

$$P \ll Q \implies |P| \leq |Q|, \tag{1}$$

where $|P|$ denotes the number of links of P. Paths with the same length are ordered according to an arbitrary ordering function.

A similar notation as the one used for costs, is also used for routes. Routes are ordered by $\lhd$. For two routes r and s, route r is said to be: *better* than route s, if $r \lhd s$; and, *worse* than route s, if $r \rhd s$. Routes are totally ordered based on its cost and path. A route (α, P) is *better* than route (β, Q), $(\alpha, P) \lhd (\beta, Q)$, if the following condition is verified.

$$\alpha \prec \beta, \text{ or } \alpha = \beta \text{ and } P \ll Q \tag{2}$$

As with costs, the election among routes is modeled by a binary operation $\triangle$. With two routes r, s, $r \triangle s = r$ if $r \trianglelefteq s$, and $r \triangle s = s$ if $r \trianglerighteq s$. Election among a set of routes is represented by $\triangle\{r_1, ..., r_n\}$.

The routing policies between neighboring nodes are modeled by functions. Function $T[uv]$ describes the transformation of costs over the link uv. The function $T[uv]$ is called the *extender* of link uv, and models the export policies of v in relation to u and the import policies of u in relation to v. A route with cost α exported by v through a link uv is learned at u with cost $T[uv](\alpha)$. By modeling the transformation of costs with a function, the possibility of two routes with the same cost at v producing two routes with different costs at u is *excluded*.

IV. Self-Stable BGP

This Section formalizes the conclusions taken from the previous Section and presents SS-BGP. Section IV-A relates termination with routing loops. Then, Section IV-B presents the pseudo-code for the canonical version of SS-BGP and compares it with that of BGP's (Section **??**), highlighting the small differences between the two. The following Sections IV-C and IV-D, suggest, respectively, better alternatives for the detection and elimination methods of SS-BGP. Finally, Section IV-E discusses SS-BGP in partial deployment.

A. Termination and routing loops

The state of BGP is composed of the candidate and elected routes at the various nodes and the routing messages in transit across the 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 the links.

The examples of Section II suggest that, in some cases, permanent oscillations are related to routing loops. A *routing loop* is a static object which refers to the propagation of a route around a network cycle. It is characterized by a triplet of the form (C, u, α), where C is a cycle, u is a node of C, and α is a cost. A route with cost α exported by u around C is learned back at u with cost $T[uCu](\alpha)$, where uCu is the circuit around C starting and ending at u and $T[uCu]$ is the composition of the extenders of all links of the circuit (see Section III). If a route learned by a node u from around a cycle C has a better cost than the cost α of the dispatched by u, then the corresponding routing loop is *persistent*. In symbols, a routing loop (C, u, α) is *persistent* if the following condition is verified.

$$T[uCu](\alpha) \prec \alpha \tag{3}$$

In contrast, a routing loop is *temporary* if the route learned from around the cycle has a worse cost than or the cost as the dispatched route, $T[uCu](\alpha) \succeq \alpha$.

The definition of persistent routing loop resembles the detection condition included in SS-BGP. Thus, SS-BGP extends the routing loop detection method already included in BGP, giving it the ability to distinguish between temporary and persistent routing loops. In addition, it also eliminates latter.

The work of Sobrinho [4] relates termination of BGP in a network to a property of routing policies around the cycles of that network. From that work, the following two propositions can be derived.

Proposition 1 *If there is a persistent routing loop, 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 persistent routing loops. However, it does not imply that eliminating all persistent routing loops ensure termination.

Proposition 2 *If routing policies are isotone and BGP does not terminate for some destination, then there is a persistent routing loop.*

This proposition implies that, in the case of isotone routing policies, the progressive elimination of persistent routing loops eventually terminates BGP. It was proved in [13] that, in the case of isotone routing policies, if SS-BGP does not terminate, then there is at least one node which detects and eliminates a persistent routing loop. This together with Proposition 2 leads to the conclusion that termination of SS-BGP is guaranteed with isotone routing policies.

B. Pseudo-code

Algorithm 1 presents the pseudo-code of SS-BGP for a fixed destination, when u receives a route (α, vP) from an in-neighbor v. Variable β stores the cost of the route learned at u from route (α, vP). An in-neighbor is active if routes learned from that neighbor are imported. Otherwise, the in-neighbor is inactive. The set $\overline{N}$ stores the in-neighbors that are inactive. Initially, $\overline{N}$ is empty, meaning all in-neighbors are active.

Variable $RouteT[w]$ stores the candidate route to reach the destination via in-neighbor w, and variable $Route$ stores the route elected by u to reach the destination. The special case of $RouteT[u]$ stores the route announced at u. The functions Cost(r) and Path(r) return, respectively, the cost and path components of a route r.

The test in Line 1 checks if v is inactive. If the test is positive, then u does not import the route received from v, setting $\bullet$ as the cost of the route learned from v in Line 2. The election of a route at Line 5 runs over all active in-neighbors of u with the exception of v. The only differences between SS-BGP and BGP are in Lines 1-4 and 7-8, which are absent in the latter protocol.

The test in Line 7 is positive if $\beta \prec \text{cost}(Route)$ and $u \in Q$, that is, if the path learned from v indicates a loop, while the cost learned from v is better than the best of all candidate costs. If the test in Line 7 is positive, then u deactivates v in Line 8, by adding v to set $\overline{N}$, on the presumption that a persistent routing loop has been found.

Algorithm 1 SS-BGP code for when node u receives a routing message (α, vP) from its in-neighbor v.

```
    if v ∈ N̄ then                                   ▷ is v inactive?
        β := •
    else
        β := T[uv](α)
 5: Route :=△ {RouteT[w] | w ≠ v}
    if u ∈ vP or β = • then
        if β ≺ Cost(Route) then                     ▷ is routing loop persistent?
            add v to N̄                              ▷ deactivate v
        RouteT[v] := (•, •)
10: else                                            ▷ if u ∉ vP and β ≺ •
        RouteT[v] := (β, vP)
        Route := Route △ RouteT[v]
    if Route has changed then
        for all w in-neighbor do
15:         send (Cost(Route), uPath(Route)) to w
```

C. Improved oscillation detection method

SS-BGP detects persistent routing loops in a distributive manner. Therefore, it is susceptible to race conditions. The network in Figure 4 is used to illustrate an example of a race condition which causes a node to signal a persistent routing when, in fact the loop is only temporary. Assuming t announces a new destination, BGP will always reach the same stable state, where y elects route (c, t) via t; x elects route (c, yt) via y; u elects route (c, yt) via y; w elects route (c, uyt) via u; and v elects route $(c, wuyt)$ via w. Now, consider that, once a stable state is reached, link yt fails. As consequence, y is left without a candidate route to the destination announced

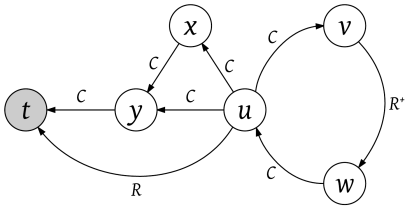

Fig. 4: When yt fails, u incorrectly signals a persistent routing loop.R and C represent peer and customer links, respectively, and the destination is announced by t.

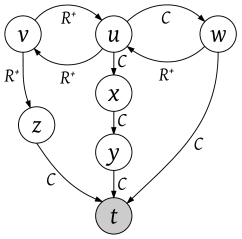

Fig. 5: BGP terminates, but if only u deploys SS-BGP, then the protocol oscillates. Peer$^+$ routes are exported to all neighbors and t announces the destination.

by t. Thus, y sends an withdrawal to u and x. Once u receives the withdrawal from y, it is forced to elect another route from its current set of candidate routes. Notice that u still believes, at this point, that the customer route (c, xyt) learned from x is valid. Therefore, u elects this route and dispatches it around the cycle $uvwu$. Meanwhile, u is informed by x that route (c, xyt) is not longer valid. Thus, u elects the only candidate route it has left, route (r, t) learned from t. Afterwards, the route dispatched by u around the cycle arrives back at u with a cost of c. Then, u compares this cost with the cost r of its alternative route and determines the routing loop around $uvwu$ is persistent. Therefore, u deactivates its neighbor v. However, as stated before, any routing loop involving $uvwu$ is temporary.

At the time a node detects a routing loop, it can not be sure if its currently elected route is the same route which originated the routing loop. However, if this were true, then the tail of the looping path, that is the path without the loop, would have to match the path of the route currently elected. Therefore, the detection condition in SS-BGP can be improved, requiring the tail of the path learned from the neighbor, with the loop discounted, to match the path of the route currently elected. In Algorithm 1, we can write vP in the form CuQ, where uCu is a circuit starting and ending at u, and uP is a path starting at u and ending at the node announcing the destination. The test condition in Line 7 is then be narrowed to:

if $\beta \prec$ Cost($Route$) **and** $P =$ Path($Route$) **then**
  add v to $\overline{N}$

The version of SS-BGP with the improved detection condition is called ISS-BGP. Back to the example in Figure 4, at $T = 4$, when u learns route $(c, vwuxyt)$. With ISS-BGP, u identifies that the tail xyt of path $vwuxyt$ does not correspond to path t of its currently elected route. Hence, u determines the routing loop is only temporary and does not deactivate v, in contrast to what happens with SS-BGP.

D. Improved oscillation elimination method

Some of the examples presented in Chapter **??** highlight that sometimes deactivating a neighbor may be an extreme measure which may have an impact on connectivity. Take the example in Figure 3. After the announcement of a destination by t, SS-BGP terminates at cost of each node deactivating its clockwise neighbor around $uvwu$. If link ut fails, then u is left with no candidate route to reach t. Although, there is a peer$^+$ route via v.

As an alternative to deactivating a neighbor, a node can inflate the costs of future routes imported from that neighbor with the goal of transforming a persistent routing loop into a temporary one. The pseudo-code for BSS-BGP adds to Algorithm 1, variable $Bound[w]$, which stores the bound cost for neighbor w. When the test condition in Line 7 is positive, the cost of the elected route is assigned to $Bound[v]$, replacing Line 8 with $Bound[v] :=$ Cost($Route$). This action affects future routes learned from v when Line 4 is replaced with,

if $T[uv](\beta) \prec Bound[v]$ **then**
  $\beta := Bound[v]$
else
  $\beta := T[uv](\alpha)$

forcing the cost β learned from v to always be worse or the same as the bound cost for v, $Bound[v]$.

Back to the example of Figure 3, when at $T = 4$, nodes u, v, and w detect a persistent routing loop, they change their import policies such that all routes learned from their clockwise neighbors have cost c rather than cost r^+, effectively treating their clockwise neighbors as if they were customers. As a result, at $T = 5$, each node u, v, and w learns from its clockwise neighbor the customer routes (c, vt), (c, wt), and (c, ut), respectively, but elects its direct customer route to t because this route has a shorter path. Thus, the protocol terminates. Now, if link ut fails, then u elects a route to reach t with cost c, learned from v.

E. Partial deployment

An important feature of SS-BGP is the fact that it uses standard BGP routing messages and, therefore, it can be deployed together with BGP. The example of Figure 1 suggests that not all nodes are required to use SS-BGP to ensure termination: if just u or v deploy SS-BGP, then the protocol still terminates. However, if only w deploys SS-BGP, then the protocol can still oscillate. From this follows that the nodes required to deploy SS-BGP must be handpicked.

Notice that if only w deploys SS-BGP, then the protocol behaves exactly as standard BGP. Thus, indicating that partially deploying SS-BGP can only improve or maintain the stability properties of BGP. However, this is not true. Consider the network of Figure 5, where node t announces the destination.

With standard BGP, the protocol reaches a stable state with v electing route (r^+, t) direct to t, u electing route (r^+, vt) learned from v, and w electing route (r^+, uvt) learned from

u. However, assuming that only u deploys SS-BGP, then v may learn a route for the destination announced by t through the sequence $t \to y \to x \to u \to v$, before v learns the route incoming directly from t. In that scenario, v exports a route $(r^+, uxyt)$ to u. As a result, u deactivates link vu. With link vu deactivated, the routing the cycle uwu may prevent the protocol from reaching a stable state without u being able to detect a persistent routing loop. The protocol oscillates because both u and w prefer routes learned from around the cycle to routes learned externally to the cycle: the peer$^+$ routes w learns from u are better than the customer routes that w learns directly from t; and the customer routes u learns from w have the same cost the ones u learns from x, however they are shorter.

Serves the example from Figure 5 to illustrate that it is possible to find a network and a destination for which BGP terminates, but SS-BGP deployed at one single, selected node does not terminate. However, in that case, there is some destination of that network for which BGP does not terminate. For instance, in Figure 5 if x is the node announcing the destination, then BGP oscillates, due to the existence of cycle uwu and the fact that u does not learn any candidate route from v to reach the destinations of x.

With peer$^+$s, it can be verified from the definition of persistent routing loop in Section IV-A that a node belongs to such a loop if, an 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. The intuition for this conclusion follows from the two following observations. First, only a customer route exported through a peer$^+$ link gives rise to a route with a better cost at the corresponding neighbor, which justifies requiring the node to be at the tail of a peer$^+$ link. And second, only customer and peer$^+$ routes are exported through a peer$^+$ or customer link, which suggests all links of the cycle must be either customer or peer$^+$ links. In the network of Figure 5, both u and w belong to a persistent routing loop around the cycle uwu. However, only w is at the tail of a peer$^+$ link in that cycle. Therefore, w is the only node able to detect and eliminate the persistent routing loop, which explains why the protocol did not terminate when only u was deploying SS-BGP.

However, a node cannot tell based on its local information whether or not it belongs to a persistent routing loop. But, it can tell from its local information whether or not it is at the tail of a peer$^+$ link (it is at the tail of a peer$^+$ link if it has a peer$^+$ neighbor), which condition is necessary to belong to a persistent routing loop. Therefore, if every node with at least one peer$^+$ link deploys SS-BGP, then the protocol terminates. From this conclusion, follows a simple guideline for AS administrators.

Guideline 1 *If an AS violates the GR routing policies by preferring some peers over customers, then that AS must deploy SS-BGP.*

The previous guideline can be widen as follows.

Guideline 2 *If an AS violates the GR routing policies in any way, then that AS must deploy SS-BGP.*

The latter does *not* apply to every isotone routing policy. However, it is valid not only with peer$^+$s, but also for siblings. Regarding the GR routing policies with siblings, a node which does not violate the baseline GR routing policies is a node which is *not* at the tail of a sibling link. That being the case, that node can only learn and export routes with costs $(c, 0)$, $(r, 0)$, and $(p, 0)$. Thus, cost component which represents the number of sibling hops can be discarded. Which means, the only component of the cost that matters is the type of route (c, r, or p). Since every extender in the siblings model (Section III) either keeps or inflates this component, then every route dispatched by that node always arrives back at the node with a worse cost than or the same cost as the dispatched route. Therefore, only a node at the tail of a sibling link is required to deploy SS-BGP to ensure termination with these routing policies.

V. Evaluation

The dynamic behavior of distributed routing protocols is too complex to evaluate analytically. Thus, it is common for research work to resort to simulators to evaluate such protocols with different network topologies. Here, the same approach was taken. A new simulator was developed to fulfill the requirements of the experiments conducted in this work. It runs standard BGP, SS-BGP, and ISS-BGP and it is capable of running each protocol in large-scale networks using very few computing resources. Resorting to this simulator, extensive simulations were conducted of BGP, SS-BGP, and ISS-BGP in realistic topologies of the Internet with the perturbations to the baseline GR routing policies described in Section **??**.

Section VI describes the implementation of the simulator with generality. It highlights some of its most important features and explains what are its inputs and outputs. Then, Section VII describes the input networks and the simulator settings that were used in each experiment. Section VIII compares the stability properties of BGP with those of SS-BGP and ISS-BGP. Section IX presents the results of applying the partial deployment guidelines presented in Section IV-E in the evaluated topologies. Finally, Section **??** presents some of the challenges faced during the realization of the simulations and presents the methods applied to overcome them.

VI. Simulator

A new discrete-event simulator was developed to fulfill the requirements of the research work conducted here. The simulator supports routing with BGP, SS-BGP, and ISS-BGP.

The input of the simulator is a formatted file, including the network topology to simulate and a set of routing configurations applied by each node in the network.

When designing a scalable simulator of BGP like protocols, it is important to determine the appropriate level of detail to take into account regarding the protocol and the network. To be able to simulate such protocols in large scale network topologies (such as the current topology of the Internet), the simulator presented here abstracts or ignores some parts which are not relevant to the subject of study of this work and focuses on the aspects that truly affect the dynamic behavior of the

protocol. In real networks, routing messages are carried over various wired media and network devices. These characteristics of the network induce various communication delays to every routing message. To evaluate the dynamic behavior of the routing protocol, the exact factors inducing message delays are not of importance, but rather the timing in which each route is received and processed at each node. In the simulator, routes are subjected to a random delay drawn from an uniform distribution within a given interval. The maximum and minimum delay values are a configurable parameters of the simulator.

The MRAI timer is an enhancement mechanism included in BGP to limit the rate with which routes are exported to neighboring nodes. When a node exports a route for a given destination to a neighbor, it starts an instance of this timer. From that point on, it is not allowed to export another route concerning this destination until the associated timer has expired after MRAI seconds. While waiting for an MRAI timer to expire, a node may learn various routes and perform the election process many times. When the timer expires, it exports the best route elected at that time. The use of the MRAI timer reduces the number of routes exchanged during an execution of the protocol and, thereby, impacts the dynamic behavior of the protocol. the simulator implements the MRAI timer. Every node contains a MRAI timer and its value can be set individually for each node.

In its current form, the simulator reports results to a file using the standard Comma Separated Value (CSV) format. For each simulation run, the following data is collected: (i) the simulation finish time, (ii) the average of the termination times of each node, (iii) the total number of routes exchanged during the simulation run, (iv) the number of nodes which detected at least one persistent routing loop, (v) the total number of detections, and (vi) within these, the number of false positive detections (see Section IV-C). The information corresponding to each recorded detection is also stored, such as, the detected loop, the detecting node, the costs of the looping and elected routes, and a flag indicating if the detection was a false positive or not.

VII. Input data and parameters

For the experiments conducted in this work the route delay values were limited to the interval between 0.01 and 1.0 seconds. No known documentation is available regarding the times for these delays. But, the previous values were considered plausible in [14].

In general, the optimal value for the MRAI depends on large number of factors and is thereby very hard to estimate. Recent research work [15] proposes a value between 3 and 5 seconds for the MRAI. In the experiments conducted in this work, the MRAI value is set to 5 seconds for all nodes.

The input networks of the simulator are based on AS-level topologies of the Internet inferred by CAIDA using the method described in [16] and published at [17]. These topologies are updated monthly. The results reported in this work refer to the topology from July 1st of 2016.

The topology from CAIDA lists pairs of ASs labeled as either a customer-provider or a peer-peer relationship following the GR routing policies. Before using this topology some operations were performed to fix some inaccuracies found in the date [18]. The method described in [**?**] was used to eliminate all customer-provider cycles and to ensure the topology is policy connected, that is, there is a valid path from every AS to every other. The topology started with $54,733$ ASs and $234,907$, after applying the fixes the resulting topology included $53,947$ ASs (98% of them) and $226,087$ relationships (96% of them).

ASs without customers do not export to neighbors routes they learn from other neighbors. Therefore, they never contribute to permanent state oscillations. These ASs are called stubs. In the previous AS network most ASs (86% of them) are stubs. To save on simulation time, stub ASs were removed from the initial network, reducing the number of nodes to $8,033$ (14% of them) and the number of links to $177,890$ (39% of them), $21,854$ of them were customer links, the same number of them were provider links, and $134,182$ of them were peer links.

BGP always terminates under the GR routing policies, but not with siblings or with peer$^+$s. Therefore, the GR routing policies were perturbed to accommodate them. Each perturbation was made separately, that is, peer$^+$s and siblings were included in different network configurations.

Peer$^+$s were added to the initial topology by replacing some peer links with peer$^+$ links. The peer$^+$ links were chosen randomly from the set of all peer links according to a uniform distribution. Results are presented for five percentages of peer$^+$ links: 0.25%, 0.5%, 1%, 5%, and 10% (relative to the total number of peer links). Five samples with different sets of peer$^+$ links were generated for each percentage of peer$^+$ links.

To accommodate for siblings in the initial network, some links were replaced by sibling links (see Section III). Siblings are usually ASs belonging to the same organization. Thus, any link between ASs that belong to the same organization was replaced by a sibling link. The data on CAIDAs Inferred AS to Organizations Mapping Dataset [17] was used to identify the organizations to which each AS belongs. The final network configuration included $4,038$ sibling links corresponding to 3% of all peer links.

VIII. Stability results

This section reports simulation results for the topologies described in the previous section. The basic experiment emulates the announcement of a new destination, corresponding to a simulation run, in which an AS announces a route to a destination that is propagated network-wide according to the rules of the routing protocol and the routing policies in effect. For any given destination AS, 100 simulation runs were executed with different seeds for generating random delays across links. The termination time of the routing protocol for a single destination is the average of termination times over all 100 simulation runs.

The termination time of BGP with the GR routing policies was used to determine the threshold parameter (see Section VI): the threshold value was set to a time 300 times greater than the termination time of BGP with the GR routing

policies. For a given destination, if at least one simulation run (out of 100) reaches the threshold, than the destination is declared as non-terminated. The same collection of 200 destinations was used for all destinations. These destination were randomly selected from the all collection of ASs with an uniform distribution.

Non-Termination of BGP

BGP may fail to terminate in the presence of peer$^+$s and siblings. However, the mere presence of peer$^+$s and siblings does not automatically imply the formation of persistent routing loops. Moreover, Section IV-A concludes that the existence of persistent routing loops does not immediately imply non-termination of BGP for all destinations. It only implies non-termination for at least one destination, which may even need to be announced by multiple nodes for the state of BGP to oscillate. Table I shows the percentages of destinations (among 200 destinations) for which BGP did not terminate subject to the GR routing policies with peer$^+$s and 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 16.0% at 0.25% of peer$^+$s links and reaches 61.8% at 1% of peer$^+$s links, whereas the percentage of destinations for which BGP does not terminate is only 2% for 3% of sibling links.

For higher percentages of peer$^+$s, the stability of BGP increases with the percentage of peer$^+$ links. Although, the percentage of destinations for which BGP does not terminate reaches 61.8% at 1% of peer$^+$s links, this percentage decreases to 35.6% at 10% of peer$^+$s links. Persistent routing loops come in various lengths, measured in terms of the number of links they contain. Considering each persistent routing loop in isolation, the likelihood of reaching a stable state in a shorter persistent routing loop is higher than in a longer one. Furthermore, the stabilization of a short persistent routing loop containing nodes that are simultaneously involved in longer persistent routing loops may also help stabilize those longer loops. As the percentage of peer$^+$ links increases so does the number of short persistent routing loops. Therefore, increasing the percentage of peer$^+$ links also increases the stability of BGP.

This also explains why the stability of BGP decreases abruptly with only a small percentage of peer$^+$s. Notice that, approximately, one quarter of the destinations did not terminate at only 0.5% of peer$^+$ links. With a lower percentage of peer$^+$ links, it is likelier persistent routing loops are longer. Therefore, the protocol becomes very unstable with the introduction of only a few peer$^+$ links.

Detections

Table II, under the heading 'Detections', presents the number of detections of persistent routing loops that it takes SS-BGP and ISS-BGP to terminate. The interesting conclusion to draw from these results is that both protocols require very few detections to ensure stabilization. For instance, at 10% of peer$^+$s, SS-BGP terminates after 64.4 detections, corresponding to 0.038% of all links.

Policies	**Non-terminated**
Siblings (3%)	2.0%
Peer$^+$ (0.25%)	16.0%
Peer$^+$ (0.5%)	25.3%
Peer$^+$ (1%)	61.8%
Peer$^+$ (5%)	45.8%
Peer$^+$ (10%)	35.6%

TABLE I: Percentage of destinations for which BGP does not terminate, for the GR routing policies with siblings and with peer$^+$s.

	Detections		False
Policies	**SS-BGP**	**ISS-BGP**	Positives
Siblings (3%)	1.2	1.0	0.14%
Peer$^+$ (0.25%)	1.1	1.0	0.05%
Peer$^+$ (0.5%)	2.7	2.4	0.17%
Peer$^+$ (1%)	5.7	4.3	11.13%
Peer$^+$ (5%)	23.8	10.3	58.24%
Peer$^+$ (10%)	64.4	17.8	75.61%

TABLE II: The first two columns show the number of detections, respectively, with SS-BGP and ISS-BGP for the GR routing policies with siblings and with peer$^+$s. The last column shows the number of false positive detections with SS-BGP.

Under the heading 'False Positives', Table II presents the percentage of detections that correspond to false positives with SS-BGP. The results show that the difference in detections between SS-BGP and ISS-BGP is justified by the number of false positives, which occur in the former method and are absent in the latter.

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 (which terminated for all destinations). The termination time of ISS-BGP is very similar to that of SS-BGP and, for that reason, it is not presented here. The times presented under the heading 'Termination' correspond to an average over the termination times of the considered destinations. The results show that even for destinations where 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, which corresponds to a 30.8% reduction.

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%

TABLE III: First two columns show that termination times of BGP, for when it terminates, and of SS-BGP, for the GR routing policies with siblings and with peer$^+$s. The last column shows the reduction in the termination times from BGP to SS-BGP.

IX. Partial deployment

SS-BGP terminates for all possible destinations in a given network without requiring every node to deploy SS-BGP. Termination is assured if all nodes that belong to a persistent routing loop deploy SS-BGP. Section IV-E shows that in the case of the GR routing policies with peer$^+$s, a node belongs to a persistent routing loop 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. This Section investigates the percentage of ASs verifying this condition in the .

Table IV, under the heading 'Loop' presents an average, over five samples, of the percentage of ASs that belong to a persistent routing loop for each percentage of peer$^+$ links in the input networks (stubs are excluded). The results show that if 7.8% of selected ASs deploy SS-BGP, then termination is guaranteed at 1% of peer$^+$ links. As expected, the percentage of ASs required to deploy SS-BGP increases with the percentage of peer$^+$ links, reaching 28.0% at 10% of peer$^+$ links.

	Loop	**Tail**
Peer$^+$ (1%)	7.8%	8.5%
Peer$^+$ (5%)	21.4%	21.5%
Peer$^+$ (10%)	28.0%	28.2%
Siblings	-	30.8%

TABLE IV: First three entries show the percentages of ASs that belong to a persistent routing loop and that are at the tail of a peer$^+$ link, for the GR routing policies with peer$^+$s. The last entry shows the percentage of ASs at the tail of a sibling link for the GR routing policies with siblings.

An AS cannot tell from its local configuration whether or not it belongs to a persistent routing loop. Section IV-E indicates that a node must deploy SS-BGP if it is at the tail of peer$^+$ link or a sibling link. The first three entries of Table IV, under the heading 'Tail', present the percentage of ASs at the tail of at least one peer$^+$ link. Comparing the with the ones corresponding to the nodes belonging to a persistent routing loop, shows that this local condition provides a very close approximation to the actual number of nodes required to use SS-BGP, with the biggest difference being at 1% of peer$^+$ links, where this condition shows a 0.7% more nodes are required to deploy SS-BGP. The last entry of Table IV, under the heading 'Tail', shows the percentage of ASs at the tail of at least one sibling link for the siblings network, showing that the deployment of SS-BGP on selected 30.8% ASs is sufficient to ensure termination.

X. Related Work

The problem of BGP state oscillations is not new. Several proposals have been presented to sightly modify BGP's operation to guarantee the protocol stabilizes. Up until now, the proposed solution all rely on some kind of route history which limits the scalability of the protocol and comprises the privacy of routing policies, which are two of the most valued aspects by AS administrators.

Griffin and Wilfong [19] were among the first to present such a solution to which they called Safe Path Vector Protocol (SPVP), which adds to the set of route attributes an history of routes: each route is exported carrying the sequence of routes which justifies the election of that route. Histories with repeated routes reveal the existence of conflicting routing policies and serve to identify routes that should be avoided: a route which is repeated in the history is no longer imported.

Ahronovitz et al. [20] proposed an improvement over the SPVP, where instead of histories being sent between ASs, each AS keeps a record of the routes elected in the recent past and uses this local history to identify a possible conflict. The instant an AS detects a repeated route in its history it sends a token to its neighbors. If that token is propagated around a cycle and arrives back at that same AS, then there is a policy conflict and the route included more than once in the history is not imported in the future.

Ee et al. [21] proposes a different approach to the previous two which also requires an history and changing the routing messages. Routes carry an extra value called global precedence value, which is the first attribute evaluated in the route election process. An higher global precedence value denotes a less preferable route. An AS determines that it may be involved in a policy conflict when it elects a route which is less preferred than its previous elected route. Before exporting the new route, that AS stores the previously elected route in a list of past elected routes and, then, it exports the new elected route with the global precedence value incremented. consequently, in future iterations of the protocol, more preferred and also more unstable routes will have higher global precedence values causing ASs to elect less preferred, but more stable routes.

All these three solutions involve major modifications to BGP, both at the message level and at the local level. They all require ASs to include extra information in the routing messages which compromises the privacy of their routing policies. Furthermore, modifying the messages exchanged between ASs implies that they are not compatible with BGP and, therefore, require network-wide deployment for their benefits to be earned. In addition, none of them scales as well as BGP since they all require some kind of route history to be recorded. In contrast, SS-BGP improves the scalability properties of BGP without requiring any history of previous seen routes and keeps BGP routing messages untouched. Consequently, it preserves the privacy of routing policies and it can be deployed incrementally with BGP without affecting its scalability.

XI. Conclusions

In this thesis work an embryonic idea was developed and refined, producing a complete solution in the form of an addon to BGP called SS-BGP, which detects and eliminates permanent state oscillations associated with routing loops. SS-BGP is guaranteed to terminate for the important class of routing policies characterized by isotonicity. Isotonicity is already present in baseline interdomain routing policies and it can be enforced in a distributive way while retaining privacy.

A new BGP and SS-BGP simulator was developed. The simulator focuses on the fundamental aspects of these protocols, which allows simulations to be conducted efficiently in large-scale networks, such as the Internet. Extensive simulations

were conducted of BGP and SS-BGP in realistic topologies of the Internet with realistic perturbations to the baseline GR routing policies. The results show that BGP did not terminate for a considerable number of the simulated destinations. In contrast, SS-BGP terminated in all cases with only a small percentage of nodes having to change their import routing policies.

It was also found that only a fraction of nodes are required to deploy SS-BGP for its stabilizing effects to be felt network-wide. However, the set of nodes required to deploy SS-BGP must be handpicked. For each alternative routing policy, a guideline was formulated for AS administrators to determine whether or not they should deploy SS-BGP based strictly on their local routing configurations. For the topologies evaluated in this work, results shows that less than a third of the nodes are required to deploy SS-BGP.

REFERENCES

[1] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (bgp-4)," tech. rep., 2005.

[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 (ToN)*, vol. 10, no. 2, pp. 232–243, 2002.

[4] J. L. Sobrinho, "Correctness of routing vector protocols as a property of network cycles," *IEEE/ACM Transactions on Networking*, 2016.

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

[6] J. L. Sobrinho, "Algebra and algorithms for qos path computation and hop-by-hop routing in the internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 10, no. 4, pp. 541–550, 2002.

[7] X. A. Dimitropoulos and G. F. Riley, "Efficient large-scale bgp simulations," *Computer Networks*, vol. 50, no. 12, pp. 2013–2027, 2006.

[8] J. H. Cowie, D. M. Nicol, and A. T. Ogielski, "Modeling the global internet," *Computing in Science & Engineering*, vol. 1, no. 1, pp. 42–50, 1999.

[9] "SS-BGP simulator." https://github.com/davidfialho14/bgp-simulator, July 2017. CAIDA.

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

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

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

[13] J. Sobrinho, D. Fialho, and P. Seia, "Stabilizing bgp through distributed elimination of persistent routing loops,"

[14] T. G. Griffin and B. J. Premore, "An experimental analysis of bgp convergence time," pp. 53–61, 2001.

[15] E. Alabdulkreem, H. S. Al-Raweshidy, and M. F. Abbod, "Mrai optimization for bgp convergence time reduction without increasing the number of advertisement messages," *Procedia Computer Science*, vol. 62, pp. 419–426, 2015.

[16] X. Dimitropoulos, D. Krioukov, M. Fomenkov, B. Huffaker, Y. Hyun, G. Riley, *et al.*, "AS relationships: Inference and validation," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 29–40, 2007.

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

[18] M. Roughan, W. Willinger, O. Maennel, D. Perouli, and R. Bush, "10 lessons from 10 years of measuring and modeling the internet's autonomous systems," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1810–1821, 2011.

[19] T. G. Griffin and G. Wilfong, "A safe path vector protocol," in *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 2, pp. 490–499, IEEE, 2000.

[20] E. Ahronovitz, J.-C. Konig, and C. Saad, "A distributed method for dynamic resolution of bgp oscillations," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 10–pp, IEEE, 2006.

[21] C. T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker, "Resolving inter-domain policy disputes," in *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 157–168, ACM, 2007.