Yale University
Department of Computer Science

Robustness of Path-Vector Protocols without Independent Route Ranking

Aaron D. Jaggard   Vijay Ramachandran

YALEU/DCS/TR-1314
April 2005

This work was partially supported by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR).

Robustness of Path-Vector Protocols without Independent Route Ranking

Aaron D. Jaggard* Vijay Ramachandran†

Abstract

Recent work has presented theoretical frameworks that rigorously model the behavior of path-vector protocols [6, 7, 13], which are primarily used for inter-domain routing on the Internet. We expand the scope of these to include protocols with route-selection procedures that cannot be captured by a per-node linear order on paths; in particular, our generalized model captures the use of commonly deployed route attributes such as MED, which is used to fine-tune routing between networks that share more than one interconnection. Using the model, we give the best-known sufficient condition guaranteeing robust convergence of path-vector protocols in the generalized case and discuss its applications to protocol design.

1 Introduction

Most routes on the Internet transit several independently administered network domains, called autonomous systems (ASes). Although routing within an AS is well understood, routing between ASes is difficult because of the diverse networks and larger distances involved. Inter-domain routing on the Internet is accomplished today using the Border Gateway Protocol (BGP) [12], a path-vector protocol. Routes are established hop-by-hop through the network as information about destinations is shared between routers; at each step, this depends on routers' locally configured

---

This work was partially supported by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR).

*Department of Mathematics, Tulane University, New Orleans, LA, USA. E-mail: adj@math.tulane.edu. Partially supported by National Science Foundation (NSF) Grant DMS–0239996 and by ONR Grants N00014–99–1–0150 and N00014–01–1–0795.

†Department of Computer Science, Yale University, New Haven, CT, USA. E-mail: vijayr@cs.yale.edu. Partially supported by a 2001–2004 DoD National Defense Science and Engineering Graduate (NDSEG) Fellowship, by ONR Grant N00014–01–1–0795, and by NSF Grant ITR–0219018.

routing policies, which can be quite expressive. Therefore, convergence to a stable set of consistent routes throughout the network is dependent on a composition of decisions involving many different, autonomously provided inputs. Previous work [14] has demonstrated that the interaction of these local policies can produce global anomalies in BGP, *e.g.*, nondeterministic routing and protocol divergence. To acheive greater network stability, a better understanding of the interaction of routing policies is necessary; furthermore, this must be done in a rigorous manner so that network operators can rely on provable guarantees about protocol behavior, even in worst-case scenarios.

This paper continues a line of work that explores the theoretical foundations of inter-domain routing and routing-policy interaction. Several existing models of path-vector protocols [5–7, 13] that are used to derive general sufficient conditions for robust convergence ignore the complexities of sharing inter-domain routes within an AS; in particular, the model of the Internet assumes that every AS can be represented by one node in a graph with a single routing policy and a single link to each neighboring node. (In reality, an AS is often made up of several routers that maintain BGP sessions to share inter-domain routes; these sessions often connect links to different neighboring ASes and provide multiple inter-connections between the same ASes.) On the other hand, work that included these complexities [1, 8] has been unable to prove general guidelines for convergence in the same way that these models have. Our paper bridges this gap by presenting a generalized model to capture the static semantics of policy interactions for both inter-domain and intra-domain BGP sessions. This model is used to derive a sufficient condition analagous to those for the simplified case.

In the remainder of this section, we review existing theoretical frameworks that inspired the methodology of this work and previous attempts to analyze anomalies related to multiple AS inter-connections, discussing a specific example with BGP. In Section 2, we introduce our model, defining relevant terms (both old and new). We then derive the generalized convergence constraint in Section 3 and discuss various applications in Section 4.

1.1 Related Work on Modeling BGP

Path-vector protocols establish connectivity by sharing putative routes to reachable destinations on a hop-by-hop basis between routers: potential routes are collected; a best route is chosen; and that choice is advertised to neighboring routers that repeat the process. Local policies influence each step of this process, because the *attributes* of a route—information in a path's data record—is changed on import and export based on policy configuration, and this affects the choice of the best routes and which routes are shared. Gao and Rexford [5] showed that constraints on policies and a simple assumption about the business-relationship structure of the Internet can guarantee robustness, *i.e.*, predictable convergence to a routing solution, even in the presence of link and node failures. The benefit of this result is that the constraints are local, *i.e.*, network stability could be achieved with little global coordination between policies, which is normally impossible given the autonomy of ASes.

Griffin, Shepherd, and Wilfong [7] presented the Stable Paths Problem (SPP) as the underlying theoretical problem being solved by BGP. SPP captured the static semantics of the interaction of routing policies on paths to a single destination as a preference ordering at each node on putative routes. They were able to give a sufficient global condition for robustness, but showed that checking individual policies exactly for the existence of a stable routing solution is *NP*-hard. The combination of these results with the Gao-Rexford conditions produced an elegant analysis of the behavior of a modified, safe version of BGP that assumed the required Internet business structure but allowed back-up routing [4].

These results were incorporated into two theoretical frameworks [6, 13] that model the behavior and design of path-vector protocols more generally; both of these give concrete, rigorous analysis of convergence conditions. They show that an underlying consistency between the preference ordering of routes at nodes and ordering routes by path length represents a sufficient condition for robust convergence equivalent to that of the original SPP work; they also explain how to achieve or enforce that ordering in various ways.

1.2 MED-Induced Oscillations

Unfortunately, the above branch of work only applies to a specific type of path-vector protocol—those in which the best-route selection procedure can be modeled by mapping paths under consideration to a rank, or weight, in some totally ordered set and choosing the path of minimum (or maximum) rank. This property is called *independent route ranking* (IRR) because the rank of a path can be determined from the attributes of that path's data structure, which, in turn, can be used to compare it to any other path for best-route selection. However, BGP's full route-selection procedure cannot be modeled in this way. In particular, use of the multi-exit discriminator (MED) attribute, which is common when two ASes share multiple inter-connections and want to perform cold-potato routing,[1] violates IRR.

[1]Normally, ASes use hot-potato routing, in which traffic destined for another AS is routed to the nearest egress point—or border router—with a connection to the next-hop AS on the path. In the case of cold-potato routing, if an AS has two or more connections to the next-hop AS, some factor other than shortest distance is used to choose the egress point. The MED attribute is a common way to force cold-potato routing, and its use will be discussed in more detail later in this paper.

MED-induced oscillations are a well-known problem of BGP [2, 3, 10], and it has been conjectured that the violation of independent route ranking is the major reason. These oscillations are especially difficult to analyze and debug on a real network because they are a product of not only BGP policy settings—involving attributes set in separately configured, independent ASes—but also internal distance settings within an AS (determined by an interior gateway routing protocol, or IGP).

There has been some theoretical work on the consequences of using the MED attribute, but the results have been incomplete. Basu *et al.* [1] and Musunuri and Cobb [11] proved that including in advertisements routes not chosen as best prevents MED-induced oscillations, but this change to BGP would increase the size of routing tables and the number or size of update messages. Griffin and Wilfong [8] enumerated canonical examples of MED-induced oscillations and described them using a narrow extension to their SPP model, but did not propose broad configuration suggestions for using the MED attribute nor a robustness constraint analogous to that given for the original SPP model. Other suggestions to solve the MED-oscillation problem affect the use of route reflectors and configuration of iBGP sessions within an AS [15] or require changing the interpretation of attributes [10].

This paper provides a complete, rigorous model that not only permits analysis of the MED attribute but also includes more general route-selection procedures that may violate the independent-route-ranking property. We fully extend the SPP framework to cover these instances of the inter-domain routing problem and derive a constraint for policy configuration that guarantees robust convergence; as it is more general, it applies to instances of the limited, original SPP model as well. The generalized SPP can be included in the existing path-vector design frameworks [6, 13] to broaden their application and give a much cleaner measure for protocol expressiveness.

2 A Generalized Framework for Inter-Domain Routing

We begin this section by reviewing the dynamics of inter-domain routing protocols. We then define route-selection functions and independent route ranking (IRR), explaining the difference between our more general definitions and the more specific definitions used in previous theoretical work. We then present the Generalized Stable Paths Problem (GSPP) and the Generalized Path-Vector Policy System (GPVPS) as the underlying theoretical problem being solved by routing protocols and the framework for routing-protocol design, respectively, both of which incorporate the generalized version of route selection. In doing so, we provide an example GSPP demonstrating a MED-induced oscillation.

2.1 Overview of Inter-Domain Routing

Internet traffic is *forwarded* from source to destination by routers along paths that traverse inter-domain and intra-domain links. Routers perform a basic forwarding operation, in which the destination IP address of a packet of traffic is matched to an entry in a forwarding table, and the packet is sent to the corresponding *next hop*—or neighboring router—listed in the entry. The job of routing protocols is to fill this forwarding table to form consistent, loopless paths for traffic to follow.

Intra-domain routing is well understood and is often based on simultaneous best-path calculations using some Interior Gateway Protocol (IGP)—at the intra-domain level, "best" is often defined as shortest. Inter-domain routing, however, is more complicated because the autonomy of domains and the scale of the Internet prevents both information about network topology to be distributed for such calculations and coordination or consistency among definitions of "best." Therefore, routes are computed on a hop-by-hop basis and decisions are influenced by local policy configurations.

Knowledge about destinations is learned through *advertisements* from neighboring routers; once a path to another AS is established, an AS will share that *reachability information* with its neighbors so that they gain knowledge of the destination as well. Assuming that destinations are first *originated* by the protocol-speaking router responsible for that destination, paths are thus established by repeating the following three-step process:

1. Information about established routes through neighboring routers is collected, called *importing* routes. The route data stored in the local routing table depends on the route information in the update message and the *import policy*; the policy may *filter* routes entirely, *i.e.*, remove them from consideration.

2. For each destination, the protocol's best-route selection procedure is used to choose best routes from the local routing table. Best routes are then used to populate the forwarding table for these destinations.

3. Best routes are advertised to neighboring routers, called *exporting*. Update-message information about these routes is influenced by *export policy*, which may also filter routes.

The routers with inter-AS connections exchanging this information are *border routers*; however, an AS consists of non-border routers that must learn how to reach external destinations as well. The inter-domain protocol is thus also used to share external destinations with internal routers. As a result, path-vector protocols accomplish two inter-domain routing tasks:

1. establishing connectivity and sharing reachability information across inter-domain links; and
2. distributing knowledge of inter-domain routes to non-border routers.

It is important to note that much of the previous theoretical work studying the convergence of BGP and other path-vector protocols, *e.g.*, [5–7, 13], modeled the Internet as a graph in which each vertex represents one AS, thereby considering only inter-AS connections and ignoring anomalous behavior related to task (2). However, such anomalies have indeed been identified [2, 3, 10], and this paper generalizes the previous theoretical work to address these anomalies.

We write paths in the direction of forwarding traffic; *e.g.*, $P = v_0 v_1 \cdots v_n$ is a path from node v_0 to destination v_n. Node v_1 is the next hop on P. At the inter-domain level, most nodes v_i will represent ASes, not individual routers. However, because of task (2), it will be important to write a portion of the path from the source router to the border router such that nodes represent internal routers; *e.g.*, we may write $P = ABC(3)(6)(12)(7)$ for a path from the source AS starting at router A through internal router B to border router C, then onto ASes 3, 6, and 12 before reaching the destination AS 7. We assume that each of transit and destination ASes can appropriately route traffic; thus inter-domain route-signaling messages do not contain intra-domain routing information for these other ASes. In general, when a router is establishing forwarding paths to a destination, we can view the Internet graph from that router's perspective as one in which all other ASes are represented by one node, neighboring ASes connect to the border routers of this router's AS, and other nodes represent the intra-domain routers and connections. (It is interesting to note that inter-domain routing protocols are themselves built on top of IP; therefore, it is expected that intra-domain IP forwarding can take place based on computations of the IGP.)

2.2 Route-Selection Functions and Independent Route Ranking

Step 2 in the above-described three-step process of choosing best routes from a routing table can be modeled by the following type of function.

Definition 2.1. A *route-selection function* σ_v maps a set of paths R to a set $S \subseteq R$ that is a set of "best" routes at node v; we write $\sigma_v(R) = S$. When we restrict the selection to a particular destination, we will write $\sigma_v^d(R) = S^d$ such that all paths S^d have destination d.

In most cases, including BGP, $|\sigma^d(R)| \leq 1$ for a set of paths R and some destination d (*i.e.*, for each destination, at most one best path is chosen). Furthermore, we assume that choosing some permitted path is preferred to choosing no path, although some paths are filtered by local policy such that they are never considered as part of the selection process. Assuming that these filtered paths are not stored in the routing table R, then for all $R^d \subset R$ to a particular destination d, $R^d \neq \emptyset$ implies $\sigma^d(R^d) \neq \emptyset$. The process of collecting and storing routes, including what data structures are used for this purpose, and how it interacts with the selection procedure depend on the protocol implementation.

Independent route ranking (IRR) means that the preference of a path relative to other paths depends only on that path alone (and any information in that path's routing-table entry) and not knowledge of other paths.

Definition 2.2. A selection function σ obeys *Independent Route Ranking* iff, for all sets of routes R_1 and R_2 and destinations d, the following two conditions hold:

1. $\sigma^d(R_1) = S$ implies $\sigma^d(R_1 \cup R_2) \cap (R_1 \setminus S) = \emptyset$; and
2. $\sigma^d(R_1) = S$ and $\sigma^d(R_1 \cup R_2) \cap S \neq \emptyset$ implies $\sigma^d(R_1 \cup R_2) \supseteq S$.

We call violations of the first condition *type-1 IRR violations* and those of the second condition *type-2 IRR violations*. In the case of single-route selection functions, the above definition of IRR is equivalent to the following: if path P_1 is chosen over all paths in P as best, then additional knowledge of a route $P_2 \notin P$ does not then permit another route $P_3 \neq P_1$ in P to be chosen as best; only P_1 or P_2 may be chosen relative to $P \cup \{P_2\}$. (Condition 2 is not relevant for single-valued selection functions.)

Previous theoretical work [6, 7, 13] on path-vector protocols modeled only selection functions that independently assign a *rank* to each route and choose the path of minimal (or maximal) rank. Selection functions written in this way are called *linear selection functions*; at each node, the preference order on unfiltered (permitted) paths is consistent with a linear order. Because the protocol-convergence conditions described in [6, 7, 13] depended on this notion of rank, they do not apply to the more general setting involving arbitrary selection functions. We note the relationship between linear selection functions and IRR below.

Definition 2.3. A selection function σ is a *linear selection function* iff there exists a map $\omega : \mathcal{P} \to \mathcal{U}$ from permitted paths $\mathcal{P}$ to a totally ordered set $\mathcal{U}$ such that

$$\forall\, R \subset \mathcal{P},\;\; \sigma(R) = \left\{P \mid \forall\, P' \in R,\; \omega(P) \leq \omega(P')\right\}.$$

Proposition 2.4. *A selection function has no IRR violations iff it can be written as a linear selection function.*

Proof. First assume that σ is a linear selection function; then there exists some ranking function ω such that $\sigma(R) = \{P \mid \omega(P) \leq \omega(P')\, \forall\, P' \in R\}$. If there were a type-1 IRR violation, then there exist R_1, R_2 such that $\sigma(R_1) = S_1$ and $\sigma(R_1 \cup R_2) = S_2$ such that $S_2 \subset (R_1 \setminus S_1)$. But then for all $P \in S_2$, it must be that $\omega(P) \le \omega(P')$ for all $P' \in (R_1 \cup R_2)$, but this implies that $\omega(P) \le \omega(P')$ for all $P' \in R_1$, thus $P \in S_1$, which contradicts $S_2 \subset (R_1 \setminus S_1)$. If there were a type-2 IRR violation, then there exist R_1, R_2 such that $\sigma(R_1) = S_1$ and $\sigma(R_1 \cup R_2) = S_2$ such that $S_2 \cap S_1 \neq S_1$ and $S_2 \cap S_1 \neq \emptyset$. Let $P \in S_1 \setminus S_2$; then $\omega(P) \le \omega(P')$ for all $P' \in R_1$ but there exists $P'' \in (R_1 \cup R_2)$ such that $\omega(P) > \omega(P'')$. Thus $P'' \in R_2$, but by transitivity, if $\omega(P) \le \omega(P')$ for $P' \in R_1$, then $\omega(P'') < \omega(P')$; this means that no route $P' \in R_1$ could be chosen by $\sigma(R_1 \cup R_2)$, which contradicts $S_2 \cap S_1 \neq \emptyset$. Therefore, any linear selection function has no IRR violations.

Now assume that we have an IRR selection function σ. For all $R \subseteq \mathcal{P}$, if $\sigma(R) = S$, then assign path ranks such that $\omega(P) \le \omega(P')$ for $P \in S$, $P' \in R$. Suppose this ordering of path ranks is not consistent with a linear order; then there exist two permitted paths P_1, P_2 such that more than one of $\omega(P_1) < \omega(P_2)$, $\omega(P_1) > \omega(P_2)$, and $\omega(P_1) = \omega(P_2)$ are true. Suppose that $\sigma(\{P_1, P_2\}) = \{P_1\}$ so that $\omega(P_1) < \omega(P_2)$, but $\omega(P_2) \le \omega(P_1)$: then there exists $R \subset \mathcal{P}$ such that $\sigma(R) \cap \{P_1, P_2\} = \{P_2\}$ and $R \supset \{P_1, P_2\}$. But then let $R_1 = \{P_1, P_2\}$ and $R_2 = R \setminus \{P_1, P_2\}$; then σ has a type-1 IRR violation with R_1 and R_2, which contradicts our IRR assumption. (By symmetry, there is also a contradiction if $\omega(P_1) > \omega(P_2)$ but $\omega(P_2) \not< \omega(P_1)$.) Suppose that $\sigma(\{P_1, P_2\}) = \{P_1, P_2\}$ so that $\omega(P_1) = \omega(P_2)$ but $\omega(P_2) \neq \omega(P_1)$: then there exists $R \subset \mathcal{P}$ such that $\sigma(R) \not\supset \{P_1, P_2\}$ and $R \supset \{P_1, P_2\}$. But then let $R_1 = \{P_1, P_2\}$ and $R_2 = R \setminus \{P_1, P_2\}$; then σ has a type-2 IRR violation with R_1 and R_2, again contradicting our IRR assumption. Therefore, any IRR selection function can be written as a linear selection function. □

It has been conjectured that IRR violations are a major cause of protocol oscillations [1, 8]. Below we prove that given one IRR violation at a node, we can construct a simple network on which the protocol diverges, but in which the other nodes' selection functions obey IRR and could satisfy previously established convergence conditions.

Theorem 2.5. *Suppose σ_v is an IRR-violating (nonlinear) selection function. Then there exists an oscillating network instance containing node v in which all other nodes have IRR (linear) selection functions.*

Proof. Assume single-valued selection; the argument below generalizes. If σ violates IRR, then there exists a set Y of paths to d containing at least vP_1, vP_2 such that $\sigma_v^d(Y) = vP_1$ and for some set of paths $Z \neq \emptyset$ such that $Z \cap Y = \emptyset$, $\sigma_v^d(Z \cup Y) = vP_2$. Let the next hops on $vP_1, vP_2, \ldots \in Y$ be $v_1, v_2, \ldots$, respectively; assume that these paths are fixed, *i.e.*, $\forall\, R \ni P_1,\ \sigma_{v_1}^d(R) = P_1$ and analogously for the other $\sigma^d_{v_i}$. Let the next hops on $vZ_1, vZ_2, \ldots \in Z$ be $z_1, z_2, \ldots$, respectively. For each z_i, let $\sigma^d_{z_i}(R \cup \{Z_i\})$ be Z_i if $R \not\ni z_i v P_2$ and $z_i v P_2$ if $R \ni z_i v P_2$; but, assume that Z_i are fixed, *i.e.*, these paths are always broadcast.

This instance diverges. Assume that the links between all routers use first-in-first-out (FIFO) communication, though the network may be asynchronous. Consider a snapshot in time in which v has chosen vP_2 as best; this only happens if it also learns of all paths in Z, which means that all z_i have selected Z_i as their best paths. After this, v will broadcast vP_2 to its neighbors, eventually reaching the z_i. These nodes will thus switch to $z_i v P_2$, withdrawing Z_i. When the withdrawal reaches v, it will switch to choosing vP_1 as best, withdrawing vP_2. This withdrawal will eventually reach the z_i (after the first broadcast of vP_2) causing these nodes to switch back to Z_i; the broadcast of these choices back to v returns us to the original state when all the z_i switch, thus producing an oscillation.

If we examine the network when v has chosen vP_1 as best, there are two possibilities: (1) All z_i have chosen Z_i as best but the broadcasts of these choices have not yet reached v; or (2) some z_i (possibly all) have not chosen Z_i as best. In case (1), the broadcasts of Z_i will eventually reach v causing a choice of vP_2 as best; this leads to the starting state above. In case (2), if Z_i is not best at some z_i, $z_i v P_2$ must be available because Z_i is fixed; thus, v must have chosen vP_2 at some previous time. In this case, choose that snapshot of time as a starting point, and it is clear that the above oscillation will occur. $\square$

2.3 Generalized Stable Paths Problem

The Stable Paths Problem (SPP) [7] was suggested as the theoretical problem underlying inter-domain routing, but limits nodes' route-selection functions to linear selection functions. We now present the generalized version first discussed in [8] to accommodate modeling attributes in BGP that are inconsistent with independent route ranking.

Definition 2.6. An instance of the *Generalized Stable Paths Problem (GSPP)* is a network $G = (V, E)$ and a set of permitted paths $\mathcal{P}$ in G to a fixed destination node $v_0 \in V$. (The set $\mathcal{P}$ of permitted paths can be partitioned into sets $\mathcal{P}_v$, $v \in V$, which are the permitted paths at node v, *i.e.*, starting at v and ending at v_0.) All nodes $v \neq v_0$ have a route-selection function $\sigma^{v_0}_v : 2^{\mathcal{P}_v} \to \mathcal{P}_v$. A *path assignment* $\pi : V \to \mathcal{P}$ is a solution to GSPP iff $\pi(v_0) = (v_0)$ and for every $v \neq v_0 \in V$, $\pi(v) = \sigma^{v_0}_v(\{vP \in \mathcal{P} \mid P = \pi(u) \text{ and } \{u, v\} \in E\})$.

Remark 2.7. GSPP is *NP*-complete. This is because GSPP is in *NP*—given a solution, it is easy to check whether it is stable—and because SPP, an *NP*-complete problem [7], trivially reduces to GSPP by writing its path preferences as (linear) selection functions. Also note that this version of the problem assumes single-valued selection functions.

Example 2.8. Figure 1 shows an example GSPP given in [8, 10]. This instance models the route-selection procedure of BGP running on a network in which the Multi-Exit Discriminator (MED) attribute is used. The network is shown from the perspective of AS 3. We briefly discuss the BGP route-selection procedure and then discuss this specific GSPP.

When a route is imported (or learned) from neighbors, it is given a local-preference value that is entered into the routing table to indicate how "good" the route is; the MED attribute, on the other hand, is set by the *exporting* (or advertising) AS to indicate *its* preference among multiple inter-AS connections. The path-selection procedure for BGP is as follows:

1. Routes with the largest local preference are chosen as best.

2. In the case of a tie, routes with the shortest AS-path length are chosen.

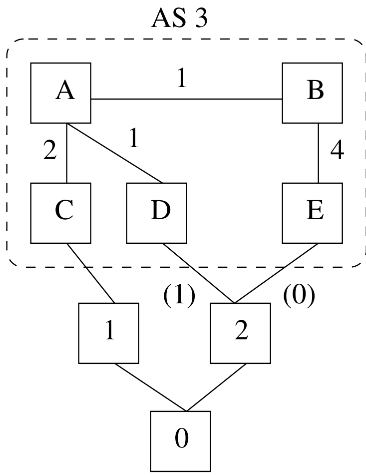
3. In the case of a tie, if there are multiple paths to the same AS, choose the path with the lowest MED value. MED values are only compared among paths to the same AS.

4. If there remains a tie because there are paths to different ASes, choose the path with the shortest IGP distance to its egress point.

Therefore, the importing AS has ultimate authority by setting local-preference values, but these are often set to the same value for all routes through given AS, even across different inter-AS links. In practice, this allows a neighboring AS to influence the decision between the inter-AS links using the MED attribute.

One typical example of MED usage is *cold-potato routing*. Assuming MEDs are not used (*i.e.*, ignoring step 3), the route-selection procedure above (via step 4) breaks ties based on closest egress point (minimal IGP distance). This is known as *hot-potato routing*. Depending on the destination prefix, a neighboring AS may specify alternate preferences for ingress points using the MED attribute, *e.g.*, to avoid using expensive intra-domain links. Consider two inter-AS connections: one in San Francisco, one in New York. A small customer network may have high costs sending traffic across its internal links. When advertising destinations to its Internet provider, the customer can attach appropriate MED values to the destinations so that the provider chooses the egress point in California or New York closest to the destination. If the provider instead used basic hot-potato routing, the closest egress point in the provider network would be chosen, possibly causing the customer to handle transcontinental traffic.

In Figure 1, IGP distances are listed as numbers next to links; MED values are listed next to inter-AS connections in parentheses. Let the fixed destination be AS 0, and assume that all paths have the same local-preference value assigned at AS 3. The selection functions for the internal routers A and B are also shown.



Selection functions for routers A and B:

$$\sigma_A^0(AC10, AD20) = AD20$$
$$\sigma_A^0(AD20, ABE20) = ABE20$$
$$\sigma_A^0(AC10, ABE20) = AC10$$
$$\sigma_A^0(AC10, AD20, ABE20) = AC10$$

$$\sigma_B^0(BAD20, BE20) = BE20$$
$$\sigma_B^0(BAC10, BE20) = BAC10$$

Figure 1: The GSPP MED-EVIL.

This instance, called MED-EVIL, was first given in [8] as an example of a MED-induced oscillation. It is important to note that both selection functions have IRR violations because of the MED values set by AS 2; thus, the paths cannot be ranked and this configuration cannot be represented as a standard SPP.

We now briefly describe why this GSPP has no solution. First assume that A and B have not advertised routes to each other; then they will choose $AD20$ and $BE20$, respectively, because of minimal IGP distances. If these nodes share these choices, B will still choose $BE20$ because, even though $BAD20$ has a shorter IGP path length, its MED value is higher than $BE20$ and both paths lead to AS 2. Router A, upon learning of $ABE20$, will no longer consider $AD20$ because of its higher MED value and will choose $AC10$ instead (because of its IGP path length is shorter than $ABE20$, the other viable option). When A's new choice is broadcast to B, router B will choose $BAC10$ because of its shorter IGP distance (over $BE20$), withdrawing $BE20$. However, this withdrawal removes the path through AS 2 with lower MED value, causing A to choose $AD20$ again, withdrawing $AC10$. Thus, we have an oscillation similar to that in the proof above.

2.4 Generalized Path-Vector Policy Systems

The Path-Vector Policy System (PVPS) framework is a model of all the components of a path-vector protocol [6]. It used the standard SPP as a semantic domain for protocol expressiveness and included path rank as a basic component of the route-selection procedure. In this section, we expand the framework to include arbitrary selection functions, and we incorporate GSPP as a new measure of protocol expressiveness. The expanded framework can then be used to design and analyze non-IRR path-vector protocols.

Definition 2.9. A *generalized path-vector policy system (GPVPS)* is a triple comprising: PV, the *path-vector system* that models the underlying message-passing system for route advertisements and signaling; PL, a *policy language* used to configure local-policy inputs; and K, a *global constraint* on network instances assumed to be true for executions of the protocol modeled by PV.

The PV is described by several components, including:

$\mathcal{R}$**:** the set of allowable *path descriptors*, the data structure used to store and share routes;

L^{in}**,** L^{out}**:** constraints on *import and export policies*, which are transformations on path descriptors used to change the attributes of paths stored in routing tables;

O**:** a constraint on *originated* path descriptors (for destinations advertised for the first time);

L^{σ}**:** constraints on nodes' route-selection functions; and

t^{in}**,** t^{out}**:** the *policy-application functions*, which indicate (1) how the protocol implements the application of nodes' policies to path descriptors and (2) any built-in transformations performed on import or export.

Remark 2.10. By fixing some constraints, a GPVPS can be restricted to model a linear selection function and a standard PVPS. In particular, assume there is some totally ordered set $\mathcal{U}$ and some map $\omega : \mathcal{R} \to \mathcal{U}$, and let

$$\mathrm{L}^{\sigma}(\sigma) \Rightarrow \forall\, d,\ \sigma^{d}(R) = \min\{P \mid \omega(P) \le \omega(P')\ \forall P' \in R\}.$$

Definition 2.11. An *instance* of a GPVPS is a pair $I = (G,\ F)$ of a network $G = (V,\ E)$ and a *configuration function* F that maps each $v \in V$ to a tuple comprising:

m^{in}, m^{out}: import and export policies; *e.g.*, $m^{in}(u) : 2^{\mathcal{R}} \to 2^{\mathcal{R}}$ is a function f on sets of path descriptors describing the transformation performed when v imports descriptors from u such that $\mathrm{L}^{in}(f)$ holds (analogously for m^{out} and L^{out});

m^{orig}: a set of descriptors originated by v such that $\mathrm{O}(m^{orig})$ holds; and

σ_v: the route-selection function to be used on routing tables for node v such that $\mathrm{L}^{\sigma}(\sigma_v)$ holds.

If R is a set of path descriptors at some node u, then those path descriptors imported from u by a neighboring node v can be represented as the output of the *arc-policy function* for the signaling edge $(u,\ v)$. This function combines the effect of export- and import-policy application and is defined as

$$f_{(u,v)}(R) = t^{in}\left(v,\ u,\ m_v^{in}(u),\ t^{out}\left(u,\ v,\ m_u^{out}(v),\ R\right)\right).$$

A *path assignment* $\rho : V \to 2^{\mathcal{R}}$ is a *solution* to the instance I iff

$$\rho(v) = \sigma_v\left(m_v^{orig} \cup \left(\bigcup_{\{u,v\}\in E} f_{(u,v)}(\rho(u))\right)\right).$$

Remark 2.12. We assume, just as in the original work on PVPSes [6], that policy functions are *separable*, *i.e.*, if p is a policy function and R is a set of path descriptors, then $p(R) = \{p(r) \mid r \in R\}$. Separability is preserved by policy application, thus arc-policy functions are separable:

$$\forall\,(u,v) \in E,\ f_{(u,v)}(R) = \left\{f_{(u,v)}(r) \mid r \in R\right\}.$$

A solution therefore must consist of consistent, best paths to each destination. We make the following definition to ease the notation when converting paths to path descriptors and *vice versa*.

Definition 2.13. Any *realizable path* $P = vv_1v_2\cdots v_nd$, which is a path that is contained in m_v^{orig} or is an extension of a path realizable at the next hop (v_1), has a corresponding path descriptor

$$r(P) = f_{(v_1,v)} \circ f_{(v_2,v_1)} \circ \cdots \circ f_{(d,v_n)}(r^*),$$

where $r^* \in \mathcal{R}$ is the path descriptor for the originated destination d. If P is not realizable or is filtered, $r(P) = \emptyset$.

Conversely, given a path descriptor r, let $\mathcal{P}(r)$ be the path described by r.

Because the GPVPS framework is general enough to model various types of inputs and constraints on those inputs, the implementation of intended protocol behavior is left to the protocol designer; several methods may achieve the same types of permitted routing configurations. For example, the general notion of "routing-policy expressiveness" might be captured by either allowing broad choices of selection functions for individual routers, or by defining a simple route-selection procedure and limiting path-descriptor transformations via import- and export-policy constraints.

Example 2.14. To model BGP, let the set of path descriptors be the set of data records used in BGP update messages and routing tables (including attributes such as local preference and MED). Set the policy-application functions to hide (or zero-out) all private attributes on eBGP export and extend the AS-path entry, checking for and filtering loops. Then set the import and export policy constraints to limit attribute changes as described in the BGP specification [12], *e.g.*, local preference is an integer value in some range. Set the origination constraint to check for the proper form of path descriptors for local paths. Finally, set the selection function constraint so that all routers use the BGP selection procedure described earlier.

The original framework used SPP as a semantic domain to measure the *expressiveness* of a PVPS, *i.e.*, the types of routing configurations that could be expressed using a PVPS given its constraints. For example, a single-destination network instance of a shortest-paths PVPS must be consistent with an SPP in which each node's ordering on paths matches the path length. Because we allow arbitrary selection functions for GPVPSes, we define a new measure of expressiveness incorporating GSPP.

Definition 2.15. Suppose that I_r is a *restriction* of a GPVPS instance I in which the only path descriptor originated is r for some single destination d. Define the GSPP $\mathcal{S}(I_r)$ to have the set of permitted paths at each node be the realizable paths at that node, *i.e.*, $\mathcal{P} = \bigcup_{v \in V} \{P = v \cdots d \mid r(P) \neq \emptyset\}$. (These are the unfiltered paths.) Let the selection functions be the same as in the GPVPS instance. Then $\mathcal{S}(I_r)$ represents the *routing configuration* for that destination.

The *expressive power* of a GPVPS PV is the set of all allowable routing configurations (all allowable instances), *i.e.*,

$$\mathcal{M}(PV) = \{\mathcal{S}(I_r) \mid I_r \text{ is a restriction of a legal instance } I \text{ of } PV\}.$$

Remark 2.16. We note that any solution π for the GSPP $\mathcal{S}(I_r)$ corresponds to a solution ρ for the restricted GPVPS instance I_r and *vice versa*. The proofs of these facts are mostly algebraic manipulation and exactly mirror the analogous proofs in [6] for the original SPP and PVPS.

GSPP can also be used as a better measure of expressiveness for the original PVPS; all GSPPs in this case will have linear selection functions because the original PVPSes do not allow IRR violations.

The original framework defined expressiveness in terms of equivalence classes of SPPs because only the relative preference ordering of paths at each node, not the actual integer ranking function used to order paths, is important in capturing a routing configuration. Therefore, any SPP S belongs to an equivalence class of SPPs $\mathcal{E}(S)$ in which all SPPs may have different integer rank functions but have the same ordering of permitted paths at each node. We note that there is an injection from SPP equivalence classes to GSPPs; there is a canonical GSPP whose selection function orders paths in the same way as each of the SPPs in the equivalence class.

Another way to see this is from the definition of linear selection functions (Definition 2.3). Given a linear selection function, there are any number of rank maps ω that preserve the choices of the selection function. Every one of these rank assignments corresponds to a possible SPP instance (because nodes have functions assigning ranks to paths), but these SPPs all share the same relative preference between paths and thus belong to the same equivalence class. However, there is only one (canonical) GSPP for a set of selection functions assigned to nodes.

2.5 GSPP and GPVPS Convergence Properties

We are not only interested in whether policies interact such that there is a stable path assignment, *i.e.*, whether or not a GSPP has a solution, but also in how path-vector protocols, following the three-step hop-by-hop process described above, can reach that assignment. In the next section we will provide a broad sufficient condition that guarantees robust protocol convergence to a unique solution; the condition can be used as a global constraint for GPVPSes. To derive this condition, we must investigate protocol behavior in addition to the existence of solutions. The following structure, a graph constructed from a GSPP instance, allows us to do this.

To simplify our discussion, we can assume that routing to different destinations are computed completely independently; therefore, we can always discuss protocol convergence with respect to one destination. This allows us to use restricted GPVPS instances and GSPPs to describe protocol convergence in general. If a particular convergence property holds for all GSPP instances in the expressiveness of a GPVPS, we can then say the GPVPS itself has that convergence property.

Definition 2.17. The *evaluation digraph* of a GSPP instance S is a directed graph $\mathcal{T}(S) = (V_{\mathcal{T}}, E_{\mathcal{T}})$ in which the nodes represent *protocol selection states*, and the edges represent transitions between states. A selection state is a path assignment $\pi \in \left(\prod_{v \in V} \mathcal{P}_v\right)$; if $\alpha \in V_{\mathcal{T}}$, then we write the path assignment corresponding to this node as π_α. The *start state* is the node corresponding to the empty path assignment, in which $\pi(v_0) = (v_0)$ and, for $v \neq v_0$, $\pi(v) = \epsilon$, the empty path.

The directed edge (α, β) is present in $E_\mathcal{T}$ iff for all $v \neq v_0 \in V$

$$\pi_\beta(v) = \sigma_v^{v_0} \left(\bigcup_{\{u,v\} \in E} \{v\pi_\alpha(u)\} \right);$$

i.e., given that nodes select the paths π_α and then broadcast these selections to their neighbors through asynchronous FIFO links, nodes might next select the paths π_β. Note that there may already be path data in the links that has been delayed in transit, so that $\pi_\alpha(v) = P$ and $\pi_\beta(v) = P'$ but, for a neighbor u, $\pi_\alpha(u) = Q$ and $\pi_\beta(u) = uP$. (Therefore, states may not be consistent; these states are not acceptable as solutions.)

We can follow the execution of a path-vector protocol on a GSPP instance by its *trace*, which corresponds to a directed path in the evaluation digraph beginning at the start state. Traces end at *sink states*, *i.e.*, nodes whose only outgoing edges are loop edges. Because the evaluation digraph is finite, if all traces are acyclic (ignoring loop edges), then all protocol runs will converge. Conversely, it is clear that if the network can dynamically oscillate during route selection then there is a cycle in the corresponding evaluation digraph; each of the paths among which a node oscillates will appear in at least one of the states in the corresponding cycle.

Proposition 2.18. *A path assignment corresponds to a sink state iff it is a solution.*

Proof. A solution is a stable set of consistent routes. Suppose π_α is a solution; then by Definition 2.6, for all $v \neq v_0 \in V$, $\sigma_v^{v_0}\left(\bigcup_{\{u,v\}\in E}\{v\pi_\alpha(u)\}\right) = \pi_\alpha(v)$. By Definition 2.17, this is equivalent to α having no outgoing edges in the evaluation digraph other than loop edges, meaning that α is a sink state. $\square$

Therefore, we can define protocol-convergence properties in terms of the structure of the corresponding evaluation digraph. The following combinations of the existence of solutions and the ability of protocols to reach those solutions are of interest to us.

Definition 2.19. The following are convergence properties for GSPP and GPVPS instances.

Solvability A GSPP is *solvable* if there exists at least one path assignment that is a solution; *i.e.*, the evaluation digraph of the GSPP has at least one sink state.

Unique Solvability (Predictability) A routing configuration is *uniquely solvable* if there exists exactly one GSPP path assignment that is a solution; *i.e.*, the evaluation digraph contains exactly one sink state.

Safety A routing configuration is *safe* if a path-vector protocol is able to converge to a solution; *i.e.*, all traces in the GSPP's evaluation digraph are acyclic. The existence of a solution does not determine safety.

Robustness A routing configuration is *robust* if it and all sub-instances (resulting from node or link failures) are uniquely solvable and safe; *i.e.*, all traces in the GSPP evaluation digraph are acyclic and end at the same sink state.

Remark 2.20. Note that the definition of robustness, while requiring all sub-instances to be predictable and safe, requires all traces only in the original GSPP's evaluation digraph to be acyclic and end at the same sink. This is because sub-instances have evaluation digraphs that are subgraphs of the original instance's evaluation digraph (with some paths no longer possible because of failures); the property of acyclicity holds on subgraphs.

We are interested in robust path-vector protocols because these avoid nondeterminism and divergence, which are problems that are difficult for network operators to understand and debug when they occur at the inter-domain level.

3 Main Results: Conditions for Protocol Convergence

Given a set of routing-policy inputs, we can study the corresponding GSPP's evaluation digraph to see how they affect path-vector-protocol execution. However, an evaluation digraph is both large and complex; it is impractical to construct it as doing so requires simulating all possible update sequences in the GSPP instance. Griffin, Shepherd and Wilfong [7] showed that a smaller structure, called a *dispute wheel*, can be constructed from an SPP instance that is not robust. Unfortunately, the original definition of the structure is not compatible with nonlinear selection functions.

We begin this section by introducing a new version of dispute wheels and prove that it does adequately capture oscillations in generalized SPPs. From that discussion, we are then able to describe oscillations in terms of an underlying order on permitted paths described by local-policy configurations. This notion of *partially ordered SPPs* first appeared in [6]; however, because our generalized version of the problem does not have a notion of rank, we must nontrivially change the components of this order to correctly describe the robustness condition.

3.1 Generalized Dispute Wheels

Definition 3.1. A *generalized dispute wheel* (see Figure 2) contains *active nodes* $v_0, \ldots, v_k$ (with all subscripts interpreted modulo $k+1$) such that v_i has a *spoke path* Q_i to the destination d and v_i and v_{i+1} are connected by a *rim segment* R_{i+1} such that either:

1. $\exists\, S \supseteq \{Q_i,\ R_{i+1}Q_{i+1}\}$ such that $\sigma^d_{v_i}(S) = R_{i+1}Q_{i+1}$; or
2. $\exists\, S \not\ni R_{i+1}Q_{i+1}$ such that
   (a) $\sigma^d_{v_i}\left(S \cup \{Q_i\}\right) \neq Q_i$ and
   (b) $\sigma^d_{v_i}\left(S \cup \{Q_i,\ R_{i+1}Q_{i+1}\}\right) = Q_i$; or
3. $\exists\, S \not\ni R_{i+1}Q_{i+1}$ such that
   (a) $\sigma^d_{v_i}\left(S \cup \{Q_i\}\right) = Q_i$ and
   (b) $\sigma^d_{v_i}\left(S \cup \{Q_i,\ R_{i+1}Q_{i+1}\}\right) \notin \{Q_i,\ R_{i+1}Q_{i+1}\}$.

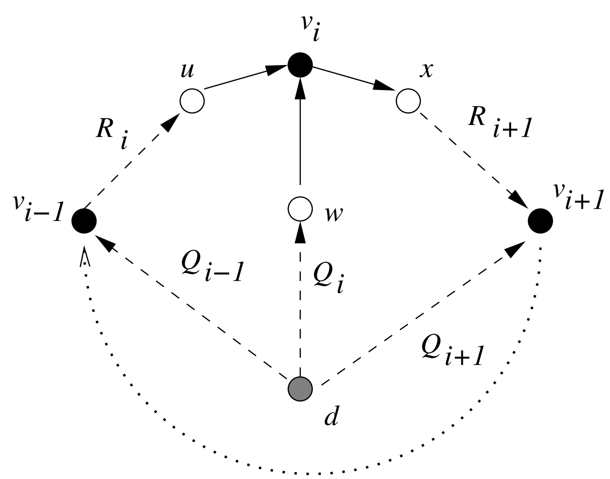


Figure 2: Dispute wheel.

Remark 3.2. Note that of the three relationships between active nodes in a generalized dispute wheel, only condition (1) can occur for a linear selection function; conditions (2) and (3) imply the existence of an IRR violation. Condition (1) is analogous to the condition on rim segments found in the original definition of a dispute wheel for standard SPPs.

The dispute wheel is a graph constructed from a GSPP instance, using the same vertices and edges in the instance's network graph. However, nodes may appear more than once in a dispute wheel, *e.g.*, in more than one spoke path.

Theorem 3.3. *If the evaluation digraph of a GSPP instance contains a cyclic trace,* i.e.*, if a GSPP instance is not safe, then it contains a generalized dispute wheel.*

Proof. Let C be a cycle in the evaluation digraph of the instance, v_0 a node which does not select the same route throughout C, and Q_0 one of the paths that v_0 selects in C. Without loss of generality, we may assume that u is the last (and thus only) node on Q_0 that does not select the same route throughout C. Viewing Q_0 as one of the spokes of a generalized dispute wheel, we now attempt to construct another such spoke and a rim segment joining it to the spoke Q_0.

Let v_0P_1 be the next path that v_0 selects in C, and let x_1 be the first node on P_1. If x_1 oscillates its path selection in C, then let v_1 be the last cycling node on P_1, let $Q_1 = v_1 \cdots d$ be the next spoke, and $R_1 = v_0x_1 \cdots v_1$ be the rim segment connecting these two spokes. (Both Q_1 and R_1 are subpaths of P_1.) Because x_1 oscillates in C, it must broadcast and withdraw P_1 during the oscillation, and one of these actions causes the selection-state transition; thus the rim segment satisfies condition (1) in Definition 3.1.

If x_1 does not oscillate in C, let v_0P_2 be the path that v_0 selects in C after v_0P_1 and x_2 the first node on P_2. If x_2 cycles in C, we may proceed as above, otherwise we consider the path v_0P_3 that v_0 selects in C after v_0P_2, *etc.* Eventually, we either construct another spoke connected to Q_0 by a new rim segment or we progress through all of C and return to the path assignment in which v_0 selects v_0P_1. If the latter happens, then v_0 cycles through a sequence of paths in C, and each of these paths is learned from a neighbor who does not cycle in C. All of these paths are thus known to v_0 at all times, therefore all of the changes in path assignment to v_0 must be the result of IRR violations. (This is because a change in path assignment requires that v_0 know of different routes before and after the change. If the change selects a route that was already known but not chosen, by Definition 2.2, the selection function for v_0 has a type-1 IRR violation.)

In this case, assume that v_0's selection of Q_0 is the result of $\sigma^d_{v_0}(S) = Q_0$ and v_0's choice of v_0P_1 is the result of $\sigma^d_{v_0}(S_1) = v_0P_1$, with $Q_0, v_0P_1 \in (S \cap S_1)$. Because $S \,\Delta\, S_1 \neq \emptyset$, there is some route P_2 such that either learning or withdrawing v_0P_2 causes the transition from S to S_1 and Q_0 to v_0P_1. Let x be the first node on P_2 and v_1 be the last oscillating node on P_2. (There is such a node because P_2 is broadcast and withdrawn in the oscillation; otherwise we would not have this oscillation.) Then we can let $Q_1 = v_1 \cdots d$ be the next spoke, and $R_1 = v_0x \cdots v_1$ be the rim segment joining them such that either condition (2)—if P_2 is learned—or condition (3)—if P_2 is withdrawn—is satisfied.

Because the oscillation cycle is finite, we can repeat this process until we reach a selection state or path assignment that we have already visited. At this point, a subset of the spoke and rim segments will form a generalized dispute wheel. ☐

Corollary 3.4. *If an instance of GSPP is not solvable, then it contains a generalized dispute wheel.*

Proof. An unsolvable GSPP has no sink state in its evaluation digraph; therefore all traces must contain cycles, and any of these cyclic traces produces a generalized dispute wheel by Theorem 3.3. ☐

Proposition 3.5. *If an instance of GSPP has multiple solutions, then it contains a generalized dispute wheel.*

Proof. We follow an analogous proof method in [7]. Suppose π_1, π_2 are two solutions; we can view these as trees in the network, rooted at the destination v_0: $T_i = \bigcup_{v \in V} \pi_i(v)$. Then let $H = (V,\ E(T_1) \cap E(T_2))$ be the graph induced by the intersection of the trees and let T be the component of H including v_0. Note that $V - V(T)$ is nonempty—otherwise $T_1 = T_2$.

In the following process, assume that all nodes u_i are assigned paths in both solutions. Choose an edge $\{u_1, v_1\} \in T_1$ where $u_1 \notin V(T)$ and $v_1 \in V(T)$. Then $\pi_1(u_1) = u_1 Q_1$, where Q_1 is the path in T from v_1 to d; $\pi_1(v_1) = \pi_2(v_1) = Q_1$ so that Q_1 is in both solutions because T is the intersection of both solutions. There is some other path $P_1 = \pi_2(u_1)$ in T_2; this path is of the form $R_2 Q_2$ where $R_2 = u_1 \cdots u_2$ contained in $T_2 \setminus H$ and $Q_2 = v_2 \cdots d$ contained in T. Note that $\pi_2(u_2) = u_2 Q_2$, so we can repeat this process by examining the path $\pi_1(u_2)$. Continuing, we can alternate between both solutions until we repeat a node u_i.

The paths R_i, Q_i form a generalized dispute wheel. This is because for each i, there must exist some $S \subset \mathcal{P}_{u_i}$ such that $\sigma_{u_i}^{v_0}(S \cup \{R_{i+1}Q_{i+1}, u_i Q_i\}) = R_{i+1}Q_{i+1}$ because for either $i = 1$ or $i = 2$, $\pi_i(u_i) = R_{i+1}Q_{i+1}$ given the construction above. (If not, π_i is not a stable solution, because the path $u_i Q_i$ must be available given that Q_i is in the intersection of both solutions.) This satisfies condition (1) in Definition 3.1. ☐

The contrapositive of the above three assertions forms a sufficient condition on GSPP instances that guarantees robust protocol convergence; we summarize this as the following.

Proposition 3.6. *If a GSPP instance has no generalized dispute wheel, it is robust.*

3.2 Partially Ordered GSPPs and Generalized Dispute Digraphs

The three types of conditions described in Definition 3.1 that connect dispute-wheel spokes by rim segments can be used to define relations between permitted paths in a GSPP. These relations can, in turn, be used to define a graph structure on the paths in a GSPP that makes the relationship between paths based on policy interactions easy to visualize. The underlying compatibility of local-policy configurations can then be described as the existence of a consistent partial order on permitted paths using these relations. By rephrasing the sufficient condition from Proposition 3.6 using these terms, we can better understand how individual policy interactions (corresponding to the following path relations) constitute a global routing anomaly.

Definition 3.7. Define the following four relations on permitted paths in a GSPP instance; assume that v_0 is the fixed destination node and that $u, v \in V$ are other network nodes.

Subpath $P_1 \ominus P_2$ iff

$$P_1 = v \cdots v_0, \quad P_2 = u \cdots v_0, \quad \text{and} \quad uP_1 = P_2$$

Linear Selection $P_1 \oslash P_2$ iff

$$P_1 = v \cdots v_0, \quad P_2 = u \cdots v_0, \quad \text{and} \quad \exists\, S : \sigma_u^{v_0}\left(\{uP_1,\ P_2\} \cup S\right) = uP_1$$

Nonlinear Selection (first type) $P_1 \odot_1 P_2$ iff $P_1 = v \cdots v_0$, $P_2 = u \cdots v_0$, and

$$\exists S \not\ni uP_1 : \sigma_u^{v_0}\left(\{P_2\} \cup S\right) \neq P_2 \quad \text{and} \quad \sigma_u^{v_0}\left(\{uP_1,\ P_2\} \cup S\right) = P_2$$

Nonlinear Selection (second type) $P_1 \odot_2 P_2$ iff $P_1 = v \cdots v_0$, $P_2 = u \cdots v_0$, and

$$\exists S \not\ni uP_1 : \sigma_u^{v_0}(S) = P_2 \quad \text{and} \quad \sigma_u^{v_0}\left(\{uP_1\} \cup S\right) \notin \{uP_1,\ P_2\}$$

We now define the following graph on the set of permitted paths using the above relations.

Definition 3.8. Given a GSPP instance S, its *generalized dispute digraph* is the directed graph $\mathcal{D}(S) = (V_{\mathcal{D}}, E_{\mathcal{D}})$. The nodes $V_{\mathcal{D}} = \mathcal{P}$ are the permitted paths in the network. The directed edge $(P_1,\ P_2)$ is present in $E_{\mathcal{D}}$ iff one of $P_1 \ominus P_2$, $P_1 \oslash P_2$, $P_1 \odot_1 P_2$, or $P_1 \odot_2 P_2$ holds.

Note that the dispute digraph is smaller than the evaluation digraph as each node is labeled with a single network route rather than a set of network routes; it is also easy to build given the definition of each node's selection function.

Because the relations correspond to transitions in the evaluation digraph and connections between dispute-wheel spokes, we can prove the following.

Theorem 3.9. *A GSPP instance has a generalized dispute wheel iff it has a cycle in its generalized dispute digraph.*

Proof. First assume that the instance has a generalized dispute wheel. Its rim gives a cycle in the generalized dispute digraph as follows, because the pair of paths from adjacent rim nodes to the destination each belong to one of the four relations in Definition 3.7. Begin with any active node v_i on the rim; let r_1 be the next node on the rim segment R_i. From the construction of the dispute wheel, $r_1Q_i = r_1v_i \cdots d$ is an extension of Q_i, so $Q_i \ominus rQ_i$; this relation holds for further extensions along the rim, such that $(r_i \cdots r_1Q_i) \ominus (r_{i+1}r_i \cdots r_1Q_i)$. Let R_i^* be the rim segment up to, but not including, v_{i-1}; using these relations, we see there is a path from Q_i to $R_i^*Q_i$ in the dispute digraph for each active node v_i in the dispute wheel. Call these paths D_i. Then, for every R_iQ_i and Q_{i-1}, one of the three conditions in Definition 3.1 holds. In the case of condition (1), $\exists\, S : \sigma^d_{v_{i-1}}(S \cup \{R_iQ_i,\ Q_{i-1}\}) = R_iQ_i$; thus $R_i^*Q_i \oslash Q_{i-1}$, corresponding to the edge $(R_i^*Q_i,\ Q_{i-1})$ connecting D_i and D_{i-1}. In the case of condition (2), learning R_iQ_i at v_{i-1} forces another route to be selected over Q_{i-1}; thus $R_i^*Q_i \odot_2 Q_{i-1}$, also corresponding to the edge $(R_i^*Q_i,\ Q_{i-1})$ connecting D_i and D_{i-1}. Finally, in the case of condition (3), withdrawing some route at v_{i-1} forces Q_{i-1} to be chosen; thus $R_i^*Q_i \odot_1 Q_{i-1}$, corresponding to the same edge connecting D_i and D_{i-1}. Therefore the dispute-digraph edges corresponding to pairwise relations between paths starting at adjacent rim nodes form a cycle.

In the other direction, assume that we have a cycle in the dispute digraph. Consider any edge $(P_1,\ P_2)$ and examine the relation between P_1 and P_2. If $P_1 \ominus P_2$, then let the first node of P_1 be a rim node and connect it to the first node of P_2 as an adjacent rim node (counterclockwise, referencing Figure 2.). If $P_1 \oslash P_2$, $P_1 \odot_1 P_2$, or $P_1 \odot_2 P_2$, then let P_2 be a spoke Q_i and connect the first node of P_2 to the first node of P_1 on the rim segment R_{i+1}; the subpath of P_1 from the first node to the last oscillating node will be the rim segment R_{i+1} and the remainder of P_1 will be the next spoke Q_{i+1}. The resulting structure will obey one of the three conditions in Definition 3.1 for rim segments connecting spokes and will have subpaths along individual rim segments (moving clockwise); therefore, this structure is the dispute wheel corresponding to the dispute-digraph cycle. □

This immediately leads to the following corollary, which provides an equivalent sufficient condition to Proposition 3.6 using the transitive closure of path relations (local conditions) instead of dispute-wheel freeness (a global condition).

Corollary 3.10. *Given a GSPP instance, if there is a cycle in its evaluation digraph, then the corresponding relation* $\bigcirc = (\ominus \cup \oslash \cup \odot_1 \cup \odot_2)^*$ *on permitted paths is not a partial order.*

Proof. If $P_1 \bigcirc P_2$, then there is a path in the dispute digraph from P_1 to P_2 (as in the proof of Theorem 3.9). If the relation $\bigcirc$ is not a partial order, there are two paths $P_1 \neq P_2$ such that $P_1 \bigcirc P_2$ and $P_2 \bigcirc P_1$; the two corresponding paths form a dispute-digraph cycle. Analogously, if there is a cycle in the dispute digraph, there are paths $P_1 \neq P_2$ corresponding to nodes in this cycle such that $P_1 \bigcirc P_2$ and $P_2 \bigcirc P_1$; thus $\bigcirc$ cannot be a partial order. The result then follows directly from Theorems 3.3 and 3.9. □

Remark 3.11. The original set of relations defined in [6] for SPP partial ordering could assume linear selection functions; thus both types of the "nonlinear selection" relation were not used. However, the "linear selection" relation was also different, defined as follows: assuming that ω is a ranking function, $P_1 \oslash P_2$ iff $\omega(P_1) \leq \omega(P_2)$. In this version of the definition, both paths begin at the same node, and the extension of P_1 to u in Definition 3.7 was captured in the transitive closure of $\oslash$ with the subpath relation $\ominus$. If we defined a similar selection relation, *i.e.*, $P_1 \oslash P_2$ iff there exists some S such that $\sigma(\{P_1, \ P_2\} \cup S) = P_1$, then any IRR violation would automatically introduce a cycle in the dispute digraph (this fact follows directly from Definition 2.2). Because not all such IRR violations cause protocol oscillations (given other nodes' policies), subsuming one subpath relation into the three selection relations eliminates these spurious cycles from dispute digraphs. Consequently, the example dispute cycles in the next subsection will appear different than in [6, 7].

Using the generalized dispute digraph, one can diagnose the cause of oscillations: cycles that involve nonlinear-selection edges are the result of IRR violations, and cycles only involving linear-selection and subpath edges are manifestations of basic policy disputes not based on IRR violations.

3.3 Example GSPPs and Dispute Digraphs

In the following diagrams of generalized dispute digraphs, we will use the following convention for edges: solid lines correspond to subpath relations, dashed lines correspond to linear-selection relations, and dotted-and-dashed lines correspond to nonlinear-selection relations. Of these, those with solid arrowheads are of the first type while those with white arrowheads are of the second type.

Example 3.12. We begin with the generalized dispute digraph for MED-EVIL, the GSPP from Example 2.8. To simplify the diagram, we have condensed ASes 1, 2, and 0 into a single AS 0 connected to routers C, D, and E; we can write analogous selection functions that maintain the policies and MED-induced oscillation in the original MED-EVIL. The graph is shown in Figure 3.

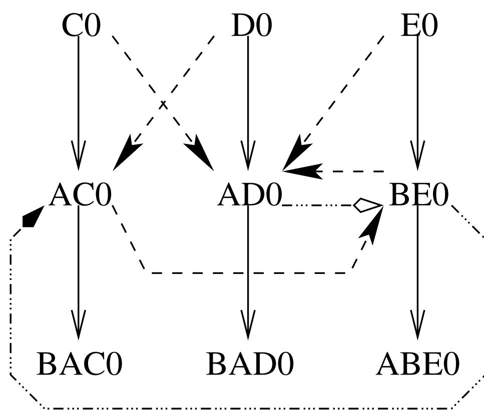


Figure 3: Generalized dispute digraph for MED-EVIL.

This digraph has two cycles, $AC0 - BE0$ and $AD0 - BE0$; as expected, both of these involve nonlinear selection edges and the paths that cause IRR violations. The policies of MED-EVIL create no oscillation when MEDs are ignored: note that the linear-selection edges do not form any cycles. Involving MEDs creates relations between paths that are not consistent with a partial order. To achieve a partial order, we can attempt to change local policies to change the relations, *i.e.*, break the cycle; *e.g.*, we can force node A to always choose the path $AC0$.

Example 3.13. We now revisit two canonical policy-induced oscillations represented by SPP instances first given by [7]. The instance DISAGREE is shown in Figure 4; it contains two stable solutions but does not predictably converge to either one, thus its dispute digraph contains a cycle. The instance BAD GADGET is shown in Figure 5; it has no solution, so its dispute digraph also contains a cycle. Because these instances have linear selection functions, the route-selection functions are shown as a linear order on permitted paths next to each node such that the most preferred path is listed on top.

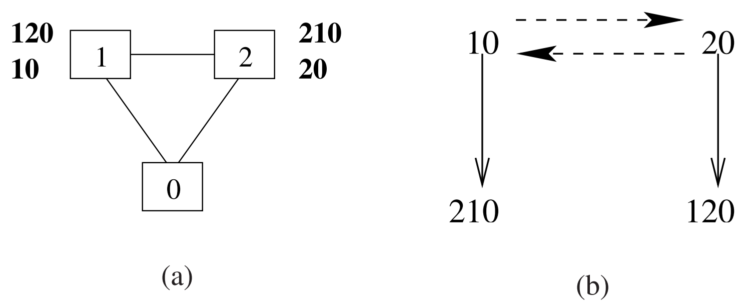


Figure 4: (a) The SPP instance DISAGREE and (b) its corresponding generalized dispute digraph.

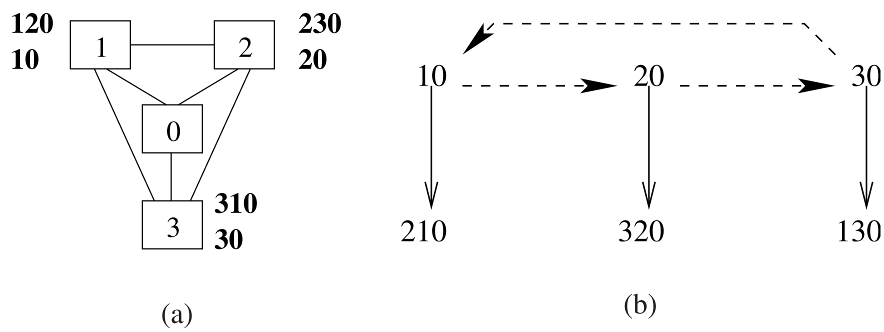


Figure 5: (a) The SPP instance BAD GADGET and (b) its corresponding generalized dispute digraph.

4 Applications to Protocol Design

In this section we examine some strategies for constraining policies to guarantee robust protocol convergence. Although dispute wheels and dispute digraphs are useful tools for studying policy interactions, using them can be impractical for real network configurations. The dispute digraph has size proportional to the number of loopless paths in a network; checking for dispute wheels is at least as hard, because there is no known way to directly produce a dispute wheel without an instance's dispute digraph or evaluation digraph. Furthermore, it is almost impossible to obtain Internet-wide policy information to generate these structures, and the structures will be different every time nodes make policy changes. Ideally, we want constraints on the protocol specification or policy-configuration language that applies to a broad set of networks and routing configurations—we would like to use the sufficient condition from the previous section while allowing for as much policy expressiveness as possible.

Previous work [5, 6, 13] has given concrete local constraints on policies that guarantee robustness. Unfortunately, generalizing these types of constraints is hard because our new model allows for nonlinear selection functions, which removes any notion of path-rank values, and because our model broadens the notion of the protocol's route-selection procedure arbitrarily.

Some obvious, draconian constraints, *e.g.*, preventing the advertisement of any route that causes an IRR violation, can be trivially shown to prevent routing anomalies, but these are very strict and harshly limit expressive power. Below we review a specific proposal to review MED-induced oscillations in BGP, and we use our tools to suggest an improvement. In the following subsections, we discuss two other conjectured solutions and, using the results from the PVPS and GPVPS frameworks, prove them to be true.

4.1 Multiple-Path Broadcast

Basu *et al.* [1] and Musunuri and Cobb [11] proved that a modification to BGP's update messages will prevent MED-induced oscillations. They suggested that nodes broadcast not only best routes, but any route that remains after step 3 in the BGP route-selection process (see Example 2.8), *i.e.*, all routes with minimal MED values, possibly one for each AS, are broadcast, not only the one with minimal IGP distance to the egress point. This prevents routes that cause IRR violations from being broadcast and withdrawn repeatedly. In the case of MED-EVIL in Example 2.8,[2] node B would then always broadcast the route $BE20$, even though it would never select it. Because the route is never chosen elsewhere due to its high MED value, this introduces no consistency problems. However, it (1) allows other nodes to make the correct choice of routes with respect to MED values and (2) stops the oscillation by making that choice stable.

We can see the effect of such a change by examining the cyclic traces in the evaluation digraph. The MED-induced cycle of MED-EVIL is shown in Figure 6. The nodes show the selections of nodes A and B, and the labels on arrows show the causes of transitions (routes being advertised or withdrawn). Note the IRR violation is clear in the transition between the first and second states; node A switches from $AD0$ to $AC0$ by learning a different route, $ABE0$. With multiple-path broadcast, the withdrawal of $ABE0$ never takes place; therefore the state $(AC0, BAC0)$ becomes a sink state and a stable assignment.

This effect easily generalizes to all GSPP instances involving MEDs: any MED-induced oscillation corresponds to an evaluation-digraph cycle of the above form, and preventing a route withdrawal by broadcasting additional routes will

[2] As in Example 3.12, we simplify the instance by condensing AS 1, AS 2, and AS 0 into a single AS 0 and modifying the selection functions accordingly.

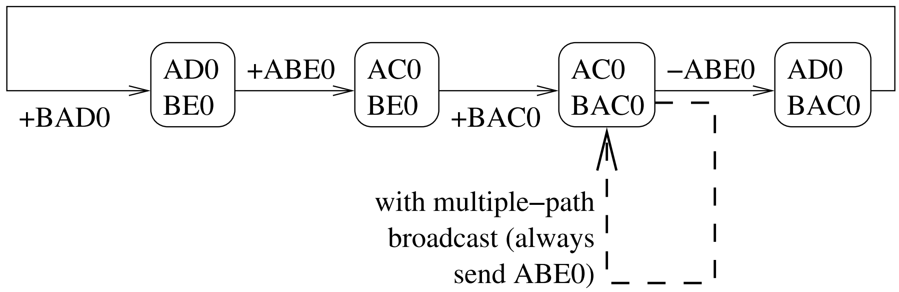


Figure 6: Cycle in the evaluation digraph of MED-EVIL.

break the cycle by preventing one (or more) of the cycle's transitions. In addition, because more routes are always broadcast, nodes will not choose higher-MED-valued routes when lower-MED-valued routes are available, thus preserving the intended behavior of the MED attribute. More generally, if routes causing IRR violations are always broadcast, the resulting route-selection functions with restricted domain have no IRR violations.

Multiple-path broadcast can increase the size of routing tables and update messages. However, we propose that IRR violations can be detected dynamically, precisely when a newly learned route causes a switch in selection without selecting the new route. Requesting that the new route always be broadcast will prevent a future oscillation due to withdrawal of that route without any route inconsistencies. Maintaining one extra route as needed is more storage-efficient than the multiple-path broadcast proposed by [1, 11]. Although this solution requires further modification to BGP, dynamic detection of IRR violations is possible by examining protocol-execution traces. In practice, whenever a BGP update message is received, the route selection before and after the update message can be compared. If the new selection is neither the old selection or the newly learned route, the route points to an IRR violation (this is clear from Definition 2.2. Requesting this IRR-violating route to be broadcast as long as it is available prevents any induced oscillations because the route essentially becomes fixed, breaking the cycle of withdrawals and advertisements in the evaluation digraph. Formally, we have the following.

Proposition 4.1. *An oscillation due to an IRR violation can be dynamically detected and stopped by requesting one additional route to be broadcast permanently.*

Proof. Given a cycle in the evaluation digraph involving an IRR violation, there are transitions in this cycle involving an advertisement or withdrawal of a route that is never selected. This route can be detected by comparing path assignments in the states adjacent to these transitions. If the withdrawal transition is prevented by forcing the route to be advertised as long as it is available, even if it is not chosen, the withdrawal transition cannot take place and the cycle is broken. □

This procedure breaks cycles in a "snapshot of time," *i.e.*, for a static routing configuration that induces a protocol oscillation. If changes occur and routes are introduced or withdrawn for legitimate causes, the resulting GSPP instance will have a different evaluation digraph; however, the relevant IRR-violating routes can be detected for this new configuration in the same way. If the IRR-violating route is no longer available, the broadcasting node can send the appropriate withdrawal—this still allows the receiving node to detect new IRR violations involving other routes. Furthermore, if any IRR-violating selections are superseded by learning new routes that are always more preferred or by other IRR-violating routes, the original routes are not needed and the broadcast can be stopped.

4.2 Compare All MEDs

Some routers have an option to change the route-selection procedure involving MEDs: In step 3 of the BGP procedure described in Example 2.8, instead of eliminating multiple paths to the same AS by choosing the one with lowest MED value, MED values are compared across all paths so that, regardless of AS next-hop, only paths with the lowest MED values are retained for possible selection.

This option essentially changes the route-selection procedure so that it is linear: for each path, the preference of that path depends, in order, on its local preference, then path length, then MED value, and finally IGP distance. Therefore, IRR violations are no longer possible, and previous convergence constraints apply. In fact, because local-preference, AS-path length, and MED values do not change during intra-domain BGP (iBGP) sessions, and because IGP distances increase as paths are extended, the absolute rank value associated with paths increases on extension. This obeys the strict-monotonicity constraints of [6, 13], so MED-induced oscillations cannot occur. (Of course, more general policy-induced oscillations due to, *e.g.*, local-preference settings, can still occur.)

Formally, comparing all MEDs changes the route-selection procedure such that selection functions are linear, compatible with the rank map $\omega(l, P, m, d) = (-l, |P|, m, d)$, lexically ordered, where l is the local preference, P is the AS path (path vector), m is the MED value, and d is the IGP distance.

4.3 AS-Distinct Local-Preference Settings

McPherson *et al.* in [10] suggest a workaround for MED-induced oscillations that prevents BGP from having a conflict when it reaches the MED step. If only routes from one AS remain when MEDs are considered, then all routes have their MED values compared and, similar to above, IRR violations are not possible. One simple way to do this is to assign local-preference values such that no two routes from different ASes have the same value; then the first step of the BGP selection process will automatically eliminate all routes except those from a single AS. (One can also assign distinct local-preference values to equidistant ASes; then the first two steps eliminate all routes but those from one AS.)

This route-selection procedure is consistent with linear selection functions because, just as above, the rank of a route independently depends on four criteria in order. Once the MED value is considered, all remaining routes have the same local preference, path length, next-hop AS, and MED value, again leaving the strictly monotonic IGP distance to be used to break ties. Therefore, this modification to BGP prevents MED-induced anomalies.

5 Conclusions and Future Work

In this paper, we have fully extended the notion of convergence conditions on SPPs to the generalized version of the problem allowing arbitrary route-selection procedures instead of those based on some notion of a linear path rank. Doing so allows us to fully understand the causes of policy-induced routing anomalies from the perspective of an underlying mathematical consistency between nodes' policies. The generalized version is able to capture the behavior of protocols, *e.g.*, BGP with MEDs, that do not satisfy independent route ranking (IRR). In examining these generalized policy interactions, we rigorously defined protocol-convergence properties and several useful graph structures that illustrate protocol behavior. We provided two equivalent sufficient conditions for robust convergence, both of which involve structures that are significantly smaller than examining the execution states of the protocol directly. We also discussed applications of these results to protocol design, including some simple (but strict) local-policy constraints and rigorous examinations of proposed solutions to MED-induced oscillations.

There are two obvious directions for future work left open in our paper. The first is the development of analogous constraints to monotonicity for arbitrary selection functions. The original condition highly depends on path-rank values, which are not available in the generalized version, but the constraint provided an equivalent local condition to dispute-wheel freeness. We have only been able to develop very simple local-policy constraints, and these limit expressiveness quite a bit. Because route-selection procedures are so broad, we expect it to be difficult to generalize previous notions of constraints. One method that may prove fruitful is examining a restricted set of route-selection procedures; *e.g.*, the application of

PVPSes to class-based systems [9] produced a set of robust protocols with a varying but concrete balance of local and global constraints. Similar work could be done while still allowing IRR violations.

The second is a further broadening of the model to capture the static semantics of policy interactions when multiple-path broadcast is used; more generally, the results can be extended to SPPs with set-valued arbitrary selection and broadcast functions. The notion of a broadcast function is similar to a selection function: it returns a subset of paths, as dictated by the protocol specification, that are advertised to neighbors, given nodes' routing tables as inputs. (This allows more latitude in modeling step 3 of path-vector-protocol behavior discussed in Section 2.1.) Separating broadcast functions from export policies allows more freedom in implementing constraints at different parts of the route-advertisement process and keeps the separability property for policy functions. Unfortunately, it seems that the problem statement for arbitrary, set-valued selection and broadcast functions is difficult; in fact, the problem of finding a stable, consistent solution for such a routing configuration may not be in *NP* because there may be a set of oscillating routing tables for each node that, given particular broadcast functions, do give at least one stable solution. This extension to the model could help design and evaluate protocols or protocol modifications such as the ones reviewed briefly in Section 4.1 without having to study large evaluation digraphs.

References

[1] Anindya Basu, Chih-Hao Luke Ong, April Rasala, F. Bruce Shepherd, and Gordon Wilfong. Route oscillations in I-BGP with route reflection. In *Proceedings of ACM SIGCOMM'02*, pages 235–247, November 2002.

[2] Cisco Systems. Endless BGP convergence problem in Cisco IOS software releases. Field Note, October 2001. `http://www.cisco.com/warp/public/770/fn12942.html`.

[3] R. Dube and J. G. Scudder. Route reflection considered harmful. IETF Internet Draft, November 1998.

[4] Lixin Gao, Timothy G. Griffin, and Jennifer Rexford. Inherently safe backup routing with bgp. In *Proceedings of IEEE INFOCOM 2001*. IEEE Communications Society, IEEE Press, 2001.

[5] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *ACM/IEEE Transactions on Networking*, 9(6):681–692, December 2001.

[6] Timothy G. Griffin, Aaron D. Jaggard, and Vijay Ramachandran. Design principles of policy languages for path vector protocols. In *Proceedings of ACM SIGCOMM'03*, pages 61–72. ACM Press, August 2003. Extended version available as Yale University Technical Report YALEU/DCS/TR-1250, `ftp://ftp.cs.yale.edu/pub/TR/tr1250.pdf`.

[7] Timothy G. Griffin, Bruce F. Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *ACM/IEEE Transactions on Networking*, 10(2):232–243, April 2002.

[8] Timothy G. Griffin and Gordon Wilfong. An analysis of the MED oscillation problem in BGP. In *Proceedings of the 10th International Conference on Network Protocols (ICNP'02)*, pages 90–99, November 2002.

[9] Aaron D. Jaggard and Vijay Ramachandran. Robustness of class-based path-vector systems. In *Proceedings of the 12th International Conference on Network Protocols (ICNP'04)*, pages 84–93. IEEE Press, October 2004. Extended version available as Yale University Technical Report YALEU/DCS/TR-1296, `ftp://ftp.cs.yale.edu/pub/TR/tr1296.pdf`.

[10] D. McPherson, V. Gill, D. Walton, and A. Retana. Border gateway protocol (BGP) persistent route oscillation condition. RFC 3345, August 2002.

[11] Ravi Musunuri and Jorge A. Cobb. A complete solution for iBGP stability. In *Proceedings of IEEE ICC-04*. IEEE Press, June 2004.

[12] Y. Rehkter and T. Li. A border gateway protocol (BGP version 4). RFC 1771, 1995.

[13] João L. Sobrinho. Network routing with path vector protocols: Theory and applications. In *Proceedings of ACM SIGCOMM'03*, pages 49–60. ACM Press, August 2003.

[14] Kannan Varadhan, Ramesh Govindan, and Deborah Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32:1–16, 2000.

[15] Daniel Walton, David Cook, Alvaro Retana, and John Scudder. BGP persistent route oscillation solution. IETF Internet Draft, May 2002.