Where's Waldo? Practical Searches for Stability in iBGP

Ashley Flavel* Matthew Roughan* Nigel Bean* Aman Shaikh†
*School of Mathematical Sciences, University of Adelaide †AT&T Labs — Research

***Abstract*—What does a child's search of a large, complex cartoon for the eponymous character (Waldo) have to do with Internet routing? Network operators also search complex datasets, but Waldo is the least of their worries. Routing oscillation is a much greater concern. Networks can be designed to avoid routing oscillation, but the approaches so far proposed unnecessarily reduce the configuration flexibility. More importantly, apparently minor changes to a configuration can lead to instability. Verification of network stability is therefore an important task, but unlike the child's search, this problem is NP hard. Until now, no practical method was available for large networks. In this paper, we present an efficient algorithm for proving stability of iBGP, or finding the potential oscillatory modes, and demonstrate its efficacy by applying it to the iBGP configuration of a large Tier-2 AS.**

I. Introduction

BGP routing oscillation degrades network performance, but is surprisingly difficult to diagnose. With the appropriate measurement infrastructure, and given sufficient time, we can detect oscillations that have been occurring, but this is unsatisfactory. For a start, detection doesn't tell you how to fix the problem. More importantly, performance degradation will have already occurred by the time the problem is detected (if it ever is, given the infrastructure and analysis requirements). Inside a network, where an operator has complete control over BGP routing, oscillation should *never* occur. Oscillation should be prevented, not fixed after the fact.

Until now, the only viable approach to prevention was to follow a set of guidelines proposed by Griffin and Wilfong [1]. These guidelines specify sufficient, but not necessary conditions for iBGP (BGP used within a network) stability. Therefore, they unnecessarily restrict configuration flexibility, and in practice are violated often. Even when they are not intentionally violated, configuration changes or failures can lead to violations, resulting in oscillations and instability.

When the guidelines for preventing oscillations are violated, further analysis is required. However, the search for potential oscillations is NP-hard [1] which makes it extremely difficult to analyze large service provider networks due to scale and dynamism involved. In this paper, we present an algorithm to detect potential BGP route oscillations inside a network based on iBGP and IGP configurations. The algorithm creates a directed graph of routers based on the notion of a "reliance". A router is said to be reliant on another when the latter's BGP route selection can impact the former's selection. When more than one router in a reliance graph form a strongly connected component [2], the routers' decisions in this component are dependent on one-another and consequently there is the possibility for route oscillation. In large networks, where route-reflection [3] is often used, the reliance graph allows us to prove that such strongly connected components can only be present in a subset of route-reflectors. We then use an algebraic approach [4] to prove or disprove the oscillatory properties of each strongly connected component.

Our approach leads to a significant reduction in the number of routers that require further analysis since the number of route-reflectors tend to be much less compared to the overall size of a network. This in turn makes the algorithm extremely scalable, allowing an operator to not only detect potential oscillations in a network design and proposed changes to it, but also perform detection as the network undergoes changes. We demonstrate the efficacy of our algorithm by employing it on a topology derived from a large Tier 2 provider. When an oscillation is actually detected, our algorithm also pinpoints the exact set of routers that cause the problem, allowing an operator to more easily fix it. Finally, the algorithm leads us to recommend a change in the BGP route selection process that can eliminate the potential for oscillations altogether.

The remainder of the paper is organized as follows. We provide background information in Section II. In Section III, we formalize the notion of stability. In Sections IV and V, we present the reliance graph theory for detection of potential oscillations and the subset of routers where they are likely to occur. Section VI then uses the algebraic approach to prove if an oscillation will actually occur. Through most of the paper we assume the final tie-breaking step in the BGP route selection process is based on router-ID. This step is directly incorporated into our reliance graph theory. However, BGP also allows the use of the "oldest-route" as the tie-break. Unfortunately, route selection becomes dependent on the timing of messages in BGP, leading to more complicated stability problems. We present a modified algebra in Section VII to show that despite this complication, the oldest-route tie-break is more appealing because it reduces the likelihood of oscillations. In Section VIII, we demonstrate the efficacy of our algorithm in the Tier 2 AS. Finally, we present related work in Section IX, and conclude in Section X.

II. Background

A. Border Gateway Protocol (BGP)

The Border Gateway Protocol (BGP) [5] is the de-facto standard used in the Internet to exchange reachability information between Autonomous Systems (ASes). Each router learns of available routes from neighboring routers and independently

1-4244-2507-5/08/$20.00 ©2008 IEEE

selects its best route by considering attributes attached to routes (see Fig. 1). Only the selected route is propagated to neighboring routers. Inbound and outbound filters are also applied to routes to enforce a local policy (see [6] for details).

1.	Highest Local Preference
2.	Shortest AS Path Length
3.	Lowest Origin Type
4.	Lowest MED (Multi-Exit Discriminator)
5.	Prefer Closest Egress (Lowest IGP Distance)
6.	Tie Breaking (Lowest-Neighbor-ID)

Fig. 1. Summarized BGP Decision Process [5], omitting vendor dependent steps.

After a router learns a route from a neighboring AS, Internal BGP (iBGP) is used to propagate routes to all other routers within the AS. The router which learns the route directly from a neighboring AS is the *egress* router (for traffic). It was originally conceived that iBGP would connect all routers in a full mesh. However, scalability concerns resulted in the introduction of a hierarchical configuration known as route-reflection [3]. Although route-reflection can have multiple hierarchical levels, in this paper we consider the commonly used two-level hierarchy (though our ideas can be extended to the more complex general case [7]). All routers are either route-reflectors (RRs) or clients of RRs. Clients propagate external routes (learned directly from neighboring ASes) to their parent RR(s). RRs select the best route and 'reflect' routes differently depending on who they are learned from. A RR's best route is reflected as follows:

Source	Reflect to:
another RR	all clients
a client	all iBGP neighbors

Scalability is achieved because the number of iBGP sessions is reduced. This comes at a price: routers now learn only a subset of potentially available routes. It has been shown that this resection has more serious consequences than suboptimal routing. It can also lead to persistent oscillation as a result of the MED attribute [8] or purely as a result of the internal topology [1]. The MED attribute is set by neighboring ASes, so an AS has no control over its values, but an operator can configure routers such that MEDs have no effect, and hence avoid MED oscillation. Although our techniques are extensible [7], in this paper we ignore MEDs and focus on the oscillation resulting from an AS's topology.

B. Best Path Selection

Routes learned externally which are discounted by steps 1-4 of the BGP decision process are *never* chosen as the final best route by any router in the network [9]. Therefore, we only need to consider the routes that survive as equally good routes after step 4. However, a router may not learn all of the globally available routes. A change in the locally *available* routes can result in a router changing its decision, and hence advertising different routes to neighbors, changing their locally available routes and so on. This is the crux of the oscillation issue we are examining.

At each router, two local decision steps determine which of the available routes are selected. Firstly, the route with the lowest IGP distance to the egress router is selected. If multiple routes have equal IGP distances, the tie is broken by selecting the route with the lowest router-id (we consider the second tie-break option of "oldest-route" later). Such decision steps are topology based and are not timing dependent. Thus, given a set of available routes, A, there is a strict preference of routes $a_1 > a_2 > a_3 > ... > a_n$. We define a ranking function λ_u for a router u, such that if a_k is preferred over a_j at router u, then $\lambda_u(a_k) > \lambda_u(a_j)$.

C. Interior Gateway Protocol

Step 5 of the BGP decision process is to prefer the closest border router, where shortest-path "distances" (these need not be geographic distances) are calculated by the Interior Gateway Protocol (IGP) — used for routing inside an AS. The key issue is that IGP distances are often unrelated to the iBGP topology. BGP sessions are routed, and so may extend over multiple physical hops. A RR's client may not be "close" to the RR. There are even good reasons (*e.g.*, redundancy) why another RR's client may be closer!

The complicated interaction between iBGP and IGP requires us to make clear distinctions between the underlying IGP network topology (which we term the physical topology) and the logical iBGP signaling topology. In this paper we will use Griffin and Wilfong's terminology [1] in which an iBGP configuration C is a pair $C = (G_P, G_S)$ where G_P is the physical graph and G_S is the signaling graph.

D. Physical Graph

The physical graph represents the physical topology of the network. It is defined by the quartet $G_P = (V, B, E_P, d)$. Each node $u \in V$ represents a router in the network. $B \subseteq V$ is the set of border (or egress) routers with physical connectivity to external networks. The set of uni-directional edges between routers is E_p, and $d(e)$ is the IGP distance administratively assigned to edge $e = (u, v) \in E_P$. A path P is a sequence of edges $P = e_1 e_2 ... e_n$. The length of P is the sum of the distances $d(e)$ for all edges e of P, and the IGP is used to compute the shortest paths.

E. Signaling Graph

The directed signaling graph $G_S = (V, A_S)$ represents the propagation of BGP routes between routers within V. An arc in G_S represents an iBGP session between two routers and is overlaid on some *path* in G_P.

The set of arcs A_S is partitioned into three sets **over**, **up**, and **down**. An arc $(u, v) \in$ **over** represents a vanilla iBGP session from router u to v. If $(u, v) \in$ **over**, then $(v, u) \in$ **over**. An arc $(u, v) \in$ **down** represents an arc from a RR u to one of its clients v. Inversely, an arc $(u, v) \in$ **up** represents an arc from a client u to its RR v. An arc $(u, v) \in$ **down** if and only if $(v, u) \in$ **up**. Arcs in **up** are acyclic — consistent with a hierarchy rather than an arbitrary network design.

A valid signaling path S satisfies the following properties. The path S can be split into sub paths $S = PQR$ where P is either empty or consists of a single arc $p \in$ **up**, R is either

empty or consists of a single arc $r \in$ **down** and Q is either empty or consists of a single arc $q \in$ **over**.

F. Egress Instance

An *egress instance* [1] $I = (C, X)$ can be defined as a pair of configuration C and a set of border routers $X \subseteq B$. The routers in X represent border routers each of which learns an external BGP route to a particular prefix. Implicitly [9], X represents routers that learn routes which are equally good for the first four steps of the BGP decision process (*i.e.*, all of the steps before choosing closest egress). All other routes will be eliminated by earlier steps in the decision process.

Note that although a border router may learn multiple routes (to a prefix) it will only advertise its best route to neighbors. It is irrelevant which route is advertised (assuming we have already passed steps 1-4). Hence, there is a one-to-one mapping from border routers X to available routes. We will refer to a border router and its available route interchangeably.

III. Defining Stability

Griffin and Wilfong define an egress instance to be *signaling correct* [1] if it is guaranteed to deterministically arrive at a unique (predictable) routing. However, we need additional terminology to describe all of the possible behaviors of egress instances, and we do so by drawing on the dynamic systems literature. We say a system is in *equilibrium* when it is in a single-state, or it cycles through a subset of states such that the cycle persists indefinitely in the absence of external influences. We call a single-state equilibrium *stable*, and a cycle *oscillatory*, by analogy to previous works (although in dynamic systems stability would be otherwise defined). An egress instance may have *more than one* possible equilibrium cycles/states, and we characterize an egress instance as *signaling unstable* if there is at least one oscillatory equilibrium, or as *signaling stable* if only stable equilibria exist. A signaling correct egress instance must be signaling stable, but if there is more than one possible equilibrium, then the equilibrium we reach for a particular egress instance is non-deterministic and so a signaling stable instance is not necessarily signaling correct.

Any configuration may have $2^{|B|}-1$ possible egress instances (though in practice all of these will not occur). If all possible egress instances are signaling correct/stable, then the configuration C is signaling correct/stable.

A. Complexity of Determining Signaling Correctness

Griffin and Wilfong [1] construct a generalized configuration $\mathcal{G}$ and demonstrate that determining if it is signaling correct is NP-hard. However, they outline a sufficient condition to ensure signaling correctness: A RR's clients should be closer (IGP distance wise) than all non-client routers. This is a *sufficient* condition, *not* a necessary condition. Networks violating this condition may be signaling stable or even signaling correct.

IV. Router Reliance Graph

A router can easily select its best route from a set of routes A that it learns. In a RR topology, the set A is dynamic and relies on other routers' decisions. However, there are many possible routes that the router would never choose in equilibrium. For instance, a router that learns a route directly from a neighboring AS will always have this route in A, and so will never select any route that is worse. We can use this simple fact to reduce the complexity of our problem dramatically. We do so through the use of a *router reliance graph* that captures only those reliances (or dependencies) that can influence the decision of a router.

The reliance graph is calculated per egress instance I. The vertices of the graph are routers, and if a router's decision is dependent on another router, then we say it is reliant and create a directed edge in the reliance graph. In other words, if u_i is reliant on u_j, we write $u_i \rightsquigarrow u_j$. The reliance graph contains only a subset of arcs from the signaling graph A_S.

Note that the arrow direction in figures and the notation used for reliance parallels the information flow in the signaling graph.

A. Reliance Rules for a Route Reflector Topology

In a two-level RR hierarchy the rules for constructing a reliance graph for an egress instance $I = (C, X)$ are:

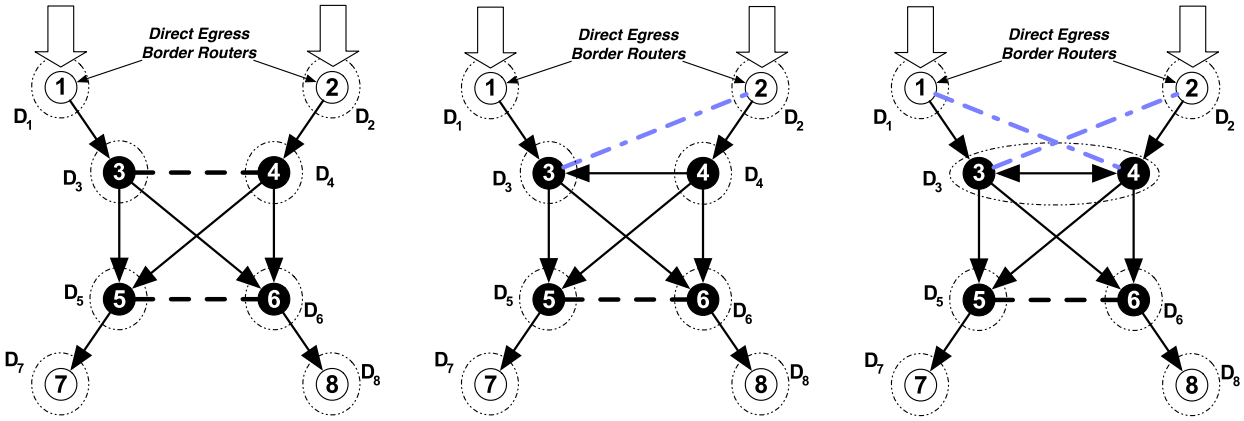
1) a router in X, that is with a direct egress, will always choose this egress, and so is not reliant on any other router's decisions;
2) a client router without a direct egress is reliant on the decisions made by its parent RR(s); and
3) a route reflector u is reliant on
   - its "best" client router
   - any other RR v whose best client router is better than u's best client, from u's perspective.

The recommendation of Griffin and Wilfong [1] amounts to configuring one's network such that a RR's own clients (where there is at least one) are always its best choice.

Formally, we define the best client egress router for RR $u \notin X$ as $\Lambda(u) \in X$, where best is with respect to rules 5 and 6 of the BGP decision process. If a RR u has no client in X, then for convenience we define $\lambda_v(\Lambda(u)) = -\infty \ \forall \ v \in V$ (recall V is the set of all routers). Now, there are three classes of directed edges in the signaling graph and they all lead to potential directed edges in the router reliance graph. Consider the three cases for an arc (u, v):

1) **up**: a client u is reliant on its RR v iff $u \notin X$.
2) **down**: a RR u is reliant on its best client egress router $\Lambda(u) \in X$, and on no other client.
3) **over**:
   a) a RR u is reliant on another RR v iff
   $$\lambda_u(\Lambda(v)) > \lambda_u(\Lambda(u)).$$
   b) A client u with an over connection to another client v is reliant on v iff $u \notin X$ and $v \in X$.

Griffin and Wilfong's condition is that for all u such that $\lambda_u(\Lambda(u)) > -\infty$ and for all $v \neq u$ we need
$$\lambda_u(\Lambda(u)) > \lambda_u(\Lambda(v)).$$



(a) $\lambda_3(1) > \lambda_3(2)$, $\lambda_4(2) > \lambda_4(1)$ (b) $\lambda_3(1) < \lambda_3(2)$, $\lambda_4(2) > \lambda_4(1)$ (c) $\lambda_3(1) < \lambda_3(2)$, $\lambda_4(2) < \lambda_4(1)$

Fig. 2. A simple egress instance. The direct egress set $X = \{1, 2\}$, is indicated by large arrows. Black nodes are RRs, white nodes are client routers, dashed lines represent iBGP sessions with no corresponding reliance, and solid lines indicate a reliance. Dash-dot lines indicate preferred clients, where these are not direct clients. Dotted ellipses indicate co-reliance groups.

B. Co-reliance Groups

Oscillation in a network occurs when two routers r_i and r_j alter their decision in response to each other's change. Consequently, by the design of the reliance graph, for oscillation to occur there must be a path in the reliance graph from r_i to r_j and from r_j to r_i. Formally, r_i and r_j must be strongly connected[1]. We define a *co-reliance group* D_k to be a strongly connected component of the reliance graph, and we denote $D(I)$ as the set of all co-reliance groups of an egress instance I. According to graph theory, the co-reliance groups form a partition of the routers [2], that is, each router is in exactly one co-reliance group.

Let us consider an example RR topology — one satisfying the sufficient condition of Griffin and Wilfong. Fig. 2(a) shows such a RR hierarchy, black nodes denote RRs, and white nodes denote client routers.

Suppose two egress routers (1 and 2) have equally good routes through step 4 of the BGP decision process, and hence $1, 2 \in X$. These routers will *always* egress via the direct egress. Hence they do not rely on any other router decisions. RRs $3, 4 \notin X$, and hence rely on the decisions of 1 and 2, respectively. This reliance is illustrated by an arrow in Fig. 2(a). RRs 5 and 6 rely only on the decisions made by 3 and 4, and client routers 7 and 8 rely on RRs 5 and 6 respectively. In this simple example, each router is part of its own co-reliance group and thus there is a unique solution for router decisions — hence the system is signaling correct.

In Fig. 2(a) we assumed that IGP distances are such that $\lambda_3(1) > \lambda_3(2)$, *i.e.*, that the RR 3's client router 1 is preferred over router 2. Likewise we assumed $\lambda_4(2) > \lambda_4(1)$. Now suppose that $\lambda_3(1) < \lambda_3(2)$ (violating the sufficient condition of Griffin and Wilfong). In this case, the decision at RR 3 is dependent on the decision made by RR 4. If RR 3 learns of router 2, via RR 4, then it will prefer this egress point. Otherwise it will prefer its client. Hence, there is an additional reliance of 3 on 4, as shown in Fig. 2(b). However, each co-reliance group still contains exactly one router and there is a unique solution, so the system is again signaling correct.

If we further change the network (see Fig. 2(c)) such that RRs 3 and 4 both prefer each others client router. That is, $\lambda_3(1) < \lambda_3(2)$ and $\lambda_4(2) < \lambda_4(1)$, then this introduces a further reliance between 4 and 3, and these two then form a single co-reliance group D_3. In this case, the equilibrium choice of routes will depend on the timing of messages *inside* the co-reliance class D_3. As multiple solutions are possible, it is not signaling correct. However, we will show that this system will not oscillate and hence is signaling stable.

V. Where Can An Oscillation Occur?

Routing oscillations can only occur when the configuration instance C is not signaling stable, and only within a co-reliance group. So our search for an oscillation can be reduced to a search for co-reliance groups. We can reduce this search still further by eliminating singleton co-reliance groups. Note the sufficient condition of Griffin and Wilfong ensures no routers are strongly connected and hence all co-reliance groups are singleton. Let us now examine where in a general RR topology a non-singleton co-reliance group can occur. In the following we define the *downstream egress set* $E \subseteq V$, as the union of X and the parent RRs of X (thus $X \subseteq E$), and we use $\overline{E}$ to denote its complement.

Theorem 5.1: For all $u \in E$ and $v \in \overline{E}$, $u \nleftrightarrow v$.

Proof: Assume there exists a router $u \in E$ and a router $v \in \overline{E}$, such that $u \leftrightsquigarrow v$. Consider the formal reliance rules for route-reflection in Section IV-A:

1) If $(u, v) \in$ **over**, then rule 3 applies. Since v has no downstream egresses, $u \nleftrightarrow v$.
2) $(u, v) \notin$ **up**, since $u \in E$ and $v \in \overline{E}$.
3) If $(u, v) \in$ **down** then as $v \notin X$, rule 2 implies $u \nleftrightarrow v$.

Thus our assumption is false. ■

Corollary 5.2: A co-reliance group cannot have routers in both E and $\overline{E}$.

[1]For any two vertices u and v in a strongly connected component of a directed graph there exists a path from u to v, and the component is the maximal such set containing these vertices.

Theorem 5.3: A non-singleton co-reliance group D does not exist in $\overline{E}$.

Proof: Assume a co-reliance group D has routers $u_1, ..., u_n \in \overline{E}$. Then there must exist a u_i, u_j, $u_k \in D$ and $u_i \neq u_j$, $u_j \neq u_k$ (u_i, u_k need not be distinct) such that $u_i \leftsquigarrow u_j$ and $u_j \leftsquigarrow u_k$. Once again we must consider when reliances between these routers can exist.

1) If $(u_i, u_j) \in$ **over**, since $u_i, u_j \in \overline{E}$, rule 3 implies that u_i, u_j have no reliance.
2) If $(u_i, u_j) \in$ **down**, then as $u_j \notin X$, rule 2 implies $u_i \not\leftsquigarrow u_j$.
3) If $(u_i, u_j) \in$ **up**, then rule 1 implies that $u_i \leftsquigarrow u_j$ since $u_i \notin X$.
    a) If $(u_j, u_k) \in$ **over**, then by 1), u_j and u_k have no reliance.
    b) If $(u_j, u_k) \in$ **down**, then by 2) $u_j \not\leftsquigarrow u_k$.
    c) $(u_j, u_k) \notin$ **up**, as we have a two level hierarchy.

Thus our assumption is false. ■

Corollary 5.4: A non-singleton co-reliance group D must be a subset of E.

These theorems show that non-singleton co-reliance groups can only occur in the downstream egress set E. The direct egress set X will typically have only a few routers in it. Even a large network might only peer at a few dozen locations, creating on the order of a few dozen routers in X. Each such border router might have two RRs (for redundancy), but rarely would they have substantially more. So E is likely to be much smaller than the complete network. Hence we need to search only a small portion of a network for potential oscillation. We can restrict our search even further due to the following result.

Theorem 5.5: A non-singleton co-reliance group D contains only RRs in E. So $D \subseteq E \backslash X$.

Proof: By Corollary 5.4 all non-singleton co-reliance groups are in E. All border routers in E are also in X and select their direct external route. Hence they do not rely on any other router. ■

In any network the number of RRs must be an order of magnitude smaller than the total number of routers (otherwise there is little point to having a RR hierarchy). In addition, the number of RRs in the downstream egress set is generally a fraction of the total number of RRs (as all must have clients with equally good routes through step 4). Thus the search space for co-reliance groups can be dramatically reduced. To locate strongly connected components there are standard graph algorithms, and given the size of the problems (a few tens of nodes) there are no performance problems on reasonably designed networks. The actual size of these groups in practice is very small — it is quite hard to construct reasonable network designs for which the group size is larger than three.

A non-singleton co-reliance group is necessary for oscillation, but not sufficient — we need to perform further analysis to classify the behavior of these groups, which we do in the following sections.

VI. Algebraic Description of Co-reliance Groups

We have shown that an oscillation can only occur within a co-reliance group, and non-singleton co-reliance groups will only ever exist between the parent RRs of direct egress routers X. Consequently, *every* arc in the co-reliance group is an **over** edge, and *every* node in the co-reliance group will know a route learned from a client. A reliance on another RR implies that the route learned from the RR is *better* than the client route. Thus, if available, the route learned indirectly from another RR is selected. By the rules of iBGP, if a RR learns a route from another RR, it will not tell another RR about this route. Given this, we can use an algebraic abstraction of routing along the lines of [4], [10] to characterize this set of rules, and analyze the behavior of co-reliance groups. We create a set of labels for edges and nodes in the graph, though we describe them with reference to nodes as the description is simpler:

- **direct (d):** A node selects its direct downstream route.
- **indirect (i):** A node selects a route learned from another node.
- **null route (ϕ):** No route is selected.

The null route, ϕ, is used for completeness. However, as every node in the co-reliance group will have a downstream egress, no equilibrium solution will ever include ϕ after a finite time. We use these labels in a routing algebra in the same vain as Sobrinho [4]. We define the labeling set of possible route selections as defined above:

$$\Sigma = \{d, i, \phi\},$$

with the preference relation:

$$i \succ d \succ \phi,$$

that is, any route is always preferred over no route, and the indirect route is preferred over a direct route because of the construction of the co-reliance group. A node's route decision is made by applying this preference to the labels of its incoming reliance arcs.

The other element of the algebra is a mapping function $\oplus$ which is applied when exporting a router's best route to neighboring routers. In iBGP, routes are exported to iBGP neighbors, but we need not consider the whole signaling graph. We only need to consider the information flow along the arcs of the reliance graph, as these are the only information flows that can affect a router's decisions. We label outgoing arcs on the reliance graph by applying the operator

$$\oplus \quad = \quad \begin{cases} d & \rightarrow & i \\ i & \rightarrow & \phi \\ \phi & \rightarrow & \phi \end{cases}$$

that is, a router won't propagate an indirect route (so it uses the null label ϕ), and a direct route becomes indirect after propagation. A stable labeling is one in which no node has a better available route than the current chosen route. However, multiple stable labelings are possible.

As an example, consider the two node co-reliance group (D_3) shown in the example of Fig. 2(c). We represent the two solutions of the co-reliance group in Fig. 3. Both nodes

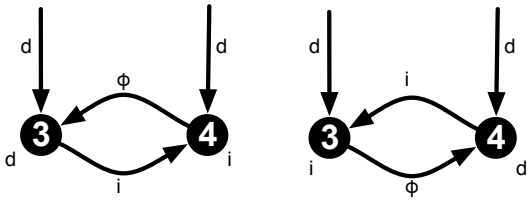


Fig. 3. Solutions for a two node co-reliance group, showing algebraic labels on edges and nodes. In addition to the co-reliance group, we also explicitly show the arcs from the direct egress set X, though in subsequent examples we will omit these because every node in the co-reliance group implicitly has such an edge available.

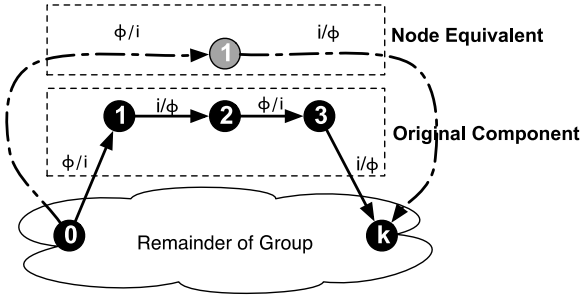


Fig. 4. A three node path is able to be replaced by a function ($Output = \overline{Input}$) or equivalently a single node.

have direct routes available via clients, however, if they ever learn of the other RR's route, they will select this indirect route. Message timing determines which solution is realized, however, the system is guaranteed to settle to a solution and hence will not persistently oscillate[2].

We can use an algebraic representation to characterize the behavior of a co-reliance group, for instance by simply enumerating states. Most co-reliance groups will be small, and so this is computationally tractable, but it is sometimes useful, for larger groups, to be able to reduce it to a smaller group with identical oscillatory properties, and hence reduce the computational complexity.

A. Reducing the Size of Co-reliance Groups

We will discover that the complexity of oscillation detection in an n-node co-reliance group is 2^n. Hence a reduction in n can significantly affect the computation time. We now present one such reduction.

Theorem 6.1: An acyclic component can be reduced to a single component with a multiple input/multiple output (MIMO) function.

Proof: The decision of each node in an acyclic component will be reliant only on its parents, and so will be a deterministic function of their decisions. Repeat this process back up to the input. Hence, the output of the acyclic component will be a MIMO function of the inputs. ■

More importantly, there is a simple algorithm (a breadth first traversal) for computing the MIMO function for a given input. This algorithm is linear in the number nodes n, so the complexity for computing the full behavior of this component will be 2^m when there are m input edges, rather than 2^n.

Consider the example shown in Fig. 4. Here the original acyclic component consists of a single input/single output three node configuration. We can replace the original component by a function that flips the input edge label. Note that we can represent the function by a node equivalent component (a single node in this example).

As we prove the stability of co-reliance groups in turn, we can imagine storing their reduced form in a *library*. Any new co-reliance group that can be reduced to a form already stored in the library does not require further enumeration. For example, any odd-node path can be reduced to a single node path as shown in Fig. 4.

[2]We assume there is enough jitter in the system such that the probability of nodes simultaneously changing decisions is small.

Group Name	**State Machine Graph Properties**
Good	No cycles
Asymptotically Good	• No oscillatory modes • At least one cycle • At least one sink
Naughty	• At least one oscillatory mode • At least one sink
Bad	No sinks

Fig. 5. Properties of Oscillation Classes.

B. Oscillation Detection

For a co-reliance group with no prior knowledge in the library with or without reduction, we use a state-machine to determine its oscillatory properties. Each state is a labeling of nodes in the reliance graph. For example, *idi* represents one possible labeling of three nodes (Node 0: *i*, Node 1: *d*, Node 2: *i*). Each transition in the state machine represents a node altering its decision as it has a better available route than currently selected. Recall we assume there is enough jitter in the system such that the probability of nodes simultaneously changing decisions is small. The actual transition depends on the timing of messages and any transition from a state has a positive probability of occurring. As we are now dealing with two graph structures, we refer to states and transitions when considering the state-machine, and nodes and edges when referring to the reliance graph.

There are two possible equilibria for the state machine. First, a stable state is such that all nodes have selected their best currently available route. Such a state is a sink in the state-machine graph. Second, an oscillatory mode is a subset of communicating states with no sink and no transitions out of the subset. If we enter such a mode, then the state machine oscillates persistently.

C. Oscillation Classes

We can partition co-reliance groups into four disjoint categories describing their oscillation characteristics: **Good, Asymptotically Good, Naughty** and **Bad.** Each class is summarized in Fig. 5 and their relationship to signaling

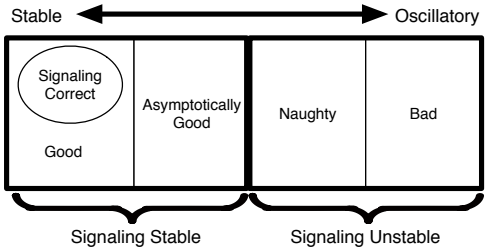


Fig. 6. Oscillation classes Venn diagram. More stable classes are to the left and more oscillatory classes are to the right.

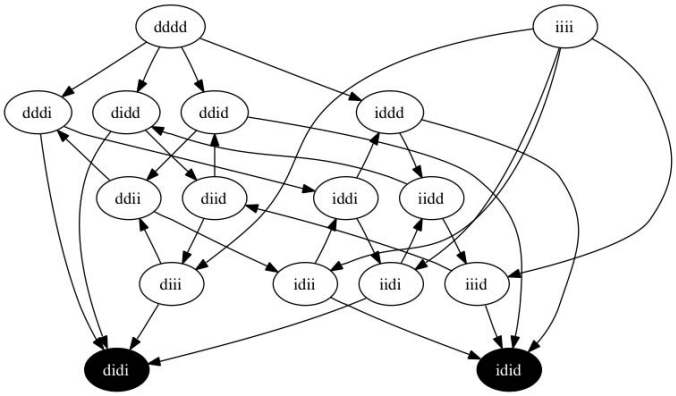


Fig. 7. Four node 'Good' state machine. Black states represent stable sink states. White nodes represent transient states.

stable, correct and unstable is shown in Fig. 6. An egress instance/configuration is classified as the most oscillatory class across all its co-reliance groups.

A **Good** co-reliance group is one where every state can be visited at most once. Thus, there must exist at least one sink state which by definition is stable. By default, all singleton co-reliance groups are classified as 'Good'. A more interesting example is a co-reliance group with four nodes $\{0,1,2,3\}$. We arrange them such that n is reliant on node $n+1 \ mod \ 4$ (each node has exactly one inbound and outbound reliance). In practice, this four node single cycle co-reliance group would be reduced to its equivalent two node representation, however, for demonstration purposes we examine it unreduced. In Fig. 7 we show the state-machine. It has 16 states. The state-machine is acyclic with two sinks (*didi* and *idid*). This configuration has similar properties to the *Good Gadget* [11], although it is not signaling correct as multiple sinks exist and we cannot determine which one of them the state machine will reach.

No sink nodes exist in a **Bad** co-reliance group. An infinite number of transitions will occur given any message ordering. An example of a 'Bad' co-reliance group is the three node single cycle shown in Fig. 9(a). Fig. 8 shows all possible labelings in the three node configuration and the transitions between labelings. As there are outbound transitions from every state, no state is stable and hence the configuration is oscillatory with a cycle consisting of $(idi, idd, iid, did, dii, ddi)$. This configuration is similar to the *Bad Gadget* [11].

A **Naughty** co-reliance group has at least one sink. However, it also has an oscillatory mode. The five node configuration shown in Fig. 9(b) demonstrates an

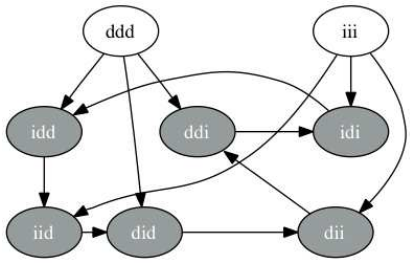


Fig. 8. Three node 'Bad' state machine. No stable sink states.

example (state-machine omitted). There exists a stable state $(ididi)$, however, there also exists a cycle $(diiid, ddiid, idiid, iddid, iidid, didid, diiid)$ from which exit is impossible. Consequently, depending on the starting state and the ordering of messages, the configuration *may* be oscillatory. This configuration has similar properties to the *Naughty Gadget* [11]. This simple example demonstrates that if a network is currently stable, it may not remain stable in the future!

An **Asymptotically Good** co-reliance group will settle on a stable labeling after a finite time. Such a co-reliance group has a state machine with a cycle in it. However, every cycle is unlocked in that it has an 'exit' such that a sink is reachable. For example, the co-reliance group in Fig. 9(c) (state-machine omitted) has a cycle $(diii, ddii, idii, iddi, iidi, didi)$. However, there is a transition $(diii, diid)$ which results in it eventually escaping the cycle and reaching the sink $(idid)$.

D. Reliances between Co-reliance Groups

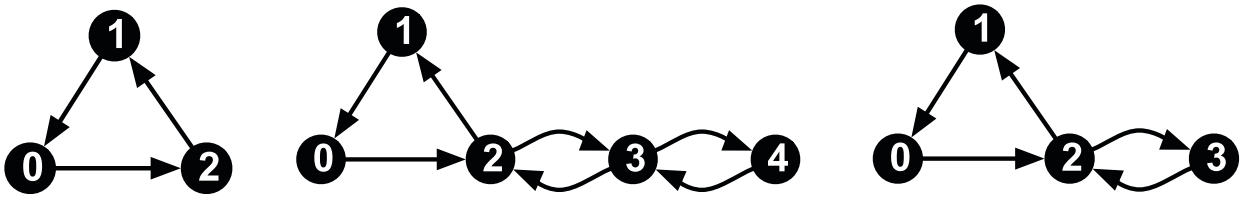
So far we have investigated co-reliance groups in isolation. However, co-reliance groups can be reliant on other co-reliance groups containing RRs. By definition, if a co-reliance group is affected by another co-reliance group the reverse cannot be true. Hence any inbound edge to a co-reliance group is *fixed*.

An inbound edge from another RR can *only be labeled ϕ or i by the definition of the algebra*. When an inbound edge is labeled ϕ, no additional information is available in the co-reliance group and thus is equivalent to the group being considered in isolation. However, when it is labeled i, the reliant *node* in the co-reliance group is fixed to be i. This makes a number of states in the state machine inaccessible. Let us now look at the impact on the oscillation classes of co-reliance groups.

Theorem 6.2: If an inbound edge to a co-reliance group is labeled i, then its state machine is a sub-graph of its state machine when considered in isolation.

Proof: If an inbound reliance edge is connected to node u_j and is labeled i, node u_j is *fixed* to select i (as no route can be better than i). Consequently, only states in the isolated co-reliance group state machine with u_j labeled i are feasible. Also, no new transitions between states are possible. Hence, the state machine of the co-reliance group is a subgraph of the isolated co-reliance group state machine. ■

Corollary 6.3: If a co-reliance group is classified 'Good' in isolation, then it will be classified as 'Good' with any inbound edges.



(a) 'Bad' co-reliance group (b) 'Naughty' co-reliance group (c) 'Asymptotically Good' co-reliance group

Fig. 9. Example co-reliance groups for oscillation classes.

Proof: Suppose a node in a 'Good' co-reliance group (classified in isolation) has a reliance on a RR outside the co-reliance group. If the inbound edge is ϕ, no additional information enters the co-reliance group and is equivalent to the isolated co-reliance group. If the inbound edge is i, by Theorem 6.2 the state machine of the new co-reliance group is a subgraph of the state machine of the isolated co-reliance group. Since no cycles exist in the original state machine, no cycles can exist in the subgraph. Hence, it is also 'Good'.■

The same cannot be said for other oscillation classes. A subgraph of the state machine can result in an unlocked cycle becoming an oscillatory mode or becoming acyclic, both of which will alter the oscillation class. For example, an inbound edge labeled i to node 3 in Fig. 9(c) will fix 3 to be labeled i. The unlocked cycle in the figure now becomes locked.

Given an egress instance, we can now determine its oscillatory properties. If an egress instance contains all singleton co-reliance groups, or all non-singleton co-reliance groups are signaling stable, then the egress instance is signaling stable. If all feasible egress instances in an iBGP configuration are signaling stable, then the configuration is signaling stable.

VII. Oldest-Route Tie-breaker

The benefit often associated with the lowest-router-id tie-breaker is the determinism associated with it. However, as we have seen, even with the lowest-router id, an egress instance can be non-deterministic. The solution the system will settle on depends on message timing. For instance, for a two-node co-reliance, that is, two RRs who each prefer the other's client, the stable states are *di* and *id* (see Fig. 3), and the solution that is eventually chosen typically depends on which RR learns of the other's client first. In contrast, the oldest-route tie-breaker was designed to restrict oscillation. We show here that this indeed is the case.

All of the reliances we have currently considered are *strong* reliances. That is, if a RR learns a route from one of its reliances on another RR, it will select that route. This is reflected in the algebra, *i.e.*, $i \succ d$. However, when the oldest-route tie-breaker is used, this is not always the case. Strong reliances still exist when the IGP distance "breaks the tie". However, if the IGP distance is equal for multiple routes and the oldest-route is the tie-breaker then the reliance is *weak*. That is, if a route is learned from another node the weak reliance implies the node *may* select the route learned from this node. Such a configuration is not as simple to describe as the preference is now dependent on message timings. However, when a reliance is weak, if a node ever selects its direct route, it will never change its selection (as it is always available and is the oldest).

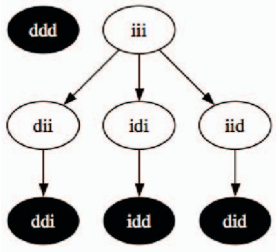


Fig. 10. The state machine of the single cycle three node co-reliance group with all 'weak' reliances. Four states are now stable.

	Reliance	
Node Label	Weak	Strong
d	i_w	i_s
i_*	ϕ	ϕ
ϕ	ϕ	ϕ

Fig. 11. Table showing the result of $\oplus$ for weak and strong reliances.

Fig. 10 shows the state-machine for the three node oscillatory configuration from Fig. 9(a) with the new tie-break rule such that all reliances are now weak. The state-machine is dramatically simplified, as compared to the state-machine shown in Fig. 8. Many of the state transitions have been removed because they will never occur under the new tie-break rule. The result is a state machine that now has four stable sink states.

The new tie break rule still allows for strong reliances where there is no tie in the IGP distances. Our approach to model this new case is to introduce an extended algebra with strong (i_s) and weak (i_w) indirect routes. That is $\Sigma = \{d, i_s, i_w, \phi\}$ with the preference relation:

$$i_s \succ i_w \simeq d \succ \phi,$$

where under the $\simeq$ operator, the oldest route is chosen. It would perhaps be more elegant to include timing into the algebra directly to resolve the $\simeq$ preference, but this complicates it substantially, and we do not need this for the proof to follow. The arcs in the reliance graph are now labeled *weak* or *strong* and the mapping function $\oplus$ now depends on this labeling as shown in Fig. 11.

Theorem 7.1: A single cycle consisting of at least one weak reliance will never oscillate.

Proof: Consider a single cycle of reliances $r_1 \leftsquigarrow r_2 \leftsquigarrow \cdots \leftsquigarrow r_n \leftsquigarrow r_1$ such that at least one of these reliances is weak, that is, $r_i \overset{w}{\leftsquigarrow} r_{i+1}$. At some point in time the information available to r_i will lead to a decision about its state, and that state $x \in \{d, i_w\}$ because of the weak reliance. If the state $x = d$, then by construction the direct route is always available, so r_i will continue to use $x = d$ (as it will from now on be the oldest available route). As soon as one node is fixed, it breaks the cyclic dependence, and removes the possibility of oscillation.

If $x = i_w$, then r_i will transmit ϕ to r_{i-1}, who will therefore choose its direct route d, and subsequently r_{i-2} will receive i_s or i_w, and make the appropriate decision, again transmitting this to its upstream neighbor. This will continue around the cycle until we return to r_i. When the cycle returns to r_i there are only two possibilities. Either r_i receives i_w, in which case, the current state is stable, or r_i receives ϕ, in which case it changes it choice to d, and the situation reverts to the case discussed above. ■

It follows that a cycle of strong reliances is required for a co-reliance group to oscillate. Consequently, we can essentially discard all weak reliances in a co-reliance group to analyze oscillatory properties! However, such weak reliances may provide inbound edge labels with the same properties as in Section VI-D.

Thus, if a co-reliance group contains no strong cycles, it is signaling stable. Hence, in general the oldest-path tie-breaker, although possibly less deterministic (as there are more stable states), is likely to be less oscillatory than the lowest-router-id tie-breaker.

VIII. Discussion

We must remember we are trying to solve an NP-hard problem. In the worst case, the problem is still exponential in the number of routers. However, in practice, for most cases the actual number of nodes we need to include in our analysis is much smaller, and so the exponential complexity of the problem becomes more manageable.

First, we do not need to calculate the entire reliance graph for an egress instance; only the reliance graph of the downstream egress set (from Corollary 5.4). We then restrict our attention to individual co-reliance groups. These are often smaller than the egress set because RRs are typically close to their clients (with respect to IGP distances). The result is that the sufficient condition of Griffin and Wilfong will be true for many routers, and these routers will not belong to a non-singleton co-reliance group. We can further reduce the size of non-singleton groups by replacing common acyclic components with a reduced component. Finally, we enumerate the state-machine of the reduced component using standard graph algorithms to detect sinks, and oscillatory modes.

Second, a large network has many potential egress instances, $|I| = 2^{|B|} - 1$, where $|B|$ is the number of border routers, and in principle we would need to analyze all of them to prove network stability. However, we can restrict this enumeration substantially by considering how networks are designed: import policies on border routers prevent a number of egresses ever being used in combination. For instance, an AS would ignore routes learned from peers to its own customers. Also, the structure of the network will lead to identical reliance graphs for multiple egress instances.

We have implemented our reliance graph analysis on an adapted[3] topology of a large (about 500 routers) Tier 2 AS. We found all current egress instances used over a 2 hour interval (954 unique egress instances). The maximum number of border routers in an egress instance was 17. All combinations of current egresses were analyzed. That is, if border routers A, B and C were in an egress instance, then all 7 non-empty subsets were also analyzed. This raised the number of egress instances requiring analysis to $204,621$. We found the reliance graph (of the downstream egress set) for all egress instances and found $60,304$ egress instances violated the sufficient condition of Griffin and Wilfong [1]. That is, there were reliances between route-reflectors in the downstream egress set — all such reliances were a result of equal IGP distances and the lowest-router-id tie-break. However, none of these reliances resulted in a non-singleton co-reliance group — hence the current set of egress instances would not oscillate (even when the sufficient condition of Griffin and Wilfong was violated). This analysis took under 15 minutes to carry out. We leave the complete results of this application including analyzing all possible egress instances (based on import policies) to future work due to space restrictions.

A. Dealing with Network Dynamics

Networks are dynamic systems changing on a regular basis. However, such changes typically involve incremental changes to the current network. The approach we present is highly amenable to an incremental implementation that analyses only those portions of the network that have changed. For instance, a change to the IGP distance between a RR and client only affects a subset of egress instances — those in which the client has an external route. Further, many IGP changes will not affect the reliance graph representation of the egress instance and hence no re-evaluation is required. If the properties of reliance graphs are stored in a *library* similar to that used to keep co-reliance group properties, many new reliance graphs need not be evaluated. Note that only the *structure* of the reliance graph and its oscillatory nature is retained in the library. The actual egress instance and the distances between routers is irrelevant.

These features of our reliance graph representation make it applicable to an online tool to detect oscillatory properties in a network. Further, our technique pinpoints the *exact location of oscillation* providing the network operator with the ability to fix the problem quickly (and test their fixes do not introduce further oscillation prior to implementation). The same technique could be used for 'what-if' purposes such as failure analysis or planned configuration changes.

[3]The route-reflector topology of the examined network had three levels and used the oldest-route tie-break. We compressed the topology to two levels and used the lowest-router-id tie-break for this analysis.

B. Preventing Oscillation

The above approach — *stability by design* — allows network operators great flexibility while ensuring stability. A simple alternative highlighted through the algebraic approach above would be to introduce an extension to the BGP protocol by adding this rule prior to step 5: *"a RR prefers its clients"*. This would shift the burden of ensuring stability onto the BGP decision process from network design and configuration. It may result in sub-optimal routing on some occasions, but so does the RR hierarchy itself.

IX. Related Work

iBGP has been shown to oscillate with ([8], [12], [13]) and without ([1]) the MED attribute affecting the BGP decision process. The oscillation resulting from the MED attribute was first described by McPherson *et al.* [8]. This prompted substantial investigation into its causes and conditions to avoid it [11], [14]–[20]. However, even with the MED attribute filtered, or compared AS-wide, it is possible that an iBGP configuration can oscillate [1], but even where MEDs are present, our techniques can be extended, by modifying the reliance rules and extending the algebra [4].

Varadhan *et al.* [21] investigated the abstract preferences of routes, finding that certain combinations of preferences across ASes result in oscillation. They developed a concept of *return* graphs with similar motivation to our reliance graph. However, their work was focused on the abstract problem of stability in path-vector protocols, while our work is focused on the practical issue of determining iBGP stability.

Griffin and Sobrinho [10] outlined an algebraic representation of BGP between ASes. This work, together with earlier work by Sobrinho [4] that described an algebraic representation of iBGP, prove general properties of the BGP. We do not attempt to design an algebraic representation of iBGP as a whole. Instead, as we are interested in oscillation, we use a much simpler algebraic representation that captures exactly what is needed to prove oscillatory properties of a co-reliance group – and thus a configuration – without the complexities of protocol idiosynchrocies.

Griffin and Wilfong [1] introduced a sufficient condition to ensure stability of iBGP. This condition was algebraically proven by Sobrinho [4]. Feamster and Rexford [9] demonstrated how to find the selected routes given this sufficient condition. In this paper, we consider the consequences when the condition is violated. Although many networks are designed to satisfy this condition, link failures, configuration errors, the addition/deletion of routers or iBGP sessions can increase the likelihood that the sufficient condition is violated. Moreover, as this is not a necessary condition, our techniques allow an operator to deviate from it. Our approach can discover if a configuration will result in a signaling correct (or a less restrictive definition of signaling stable) configuration and hence stable routing, and can point to locations in the network that are the causes of oscillation.

X. Conclusion and Future Work

The interaction between IGP and iBGP is complex. In this paper we have abstracted away the complex details, analyzed the properties of the resulting reliance graph and discovered locations where the unwanted network property – an oscillation – can occur. The approach uses careful algebraic modeling of the problem to reduce the computational complexity dramatically.

For the purposes of this paper, we have analyzed the oscillatory properties of an iBGP configuration. Further, we believe our model of iBGP can be used for applications such as determining the decisions of routers when the sufficient condition for stability does not hold, identifying the influence of route announcements from neighboring ASes, and other what-if analyses within an AS [7]. Similar concepts might also be extended to inter-AS relationships to predict the propagation of routes. We plan to look at these in the future.

Acknowledgment

Matthew Roughan and Nigel Bean were supported by Australian Research Council grant DP0557066. Ashley Flavel would also like to acknowledge the generous support of the ARC Communications Research Network.

References

[1] T. Griffin and G. Wilfong, "On the Correctness of IBGP Configuration," in *Proc. ACM SIGCOMM*, 2002.
[2] J. A. Storer, *An Introduction to Data Structures and Algorithms*. Springer, 2002.
[3] T. Bates, R. Chandra, and E. Chen, "BGP Route Reflection - An Alternative to Full Mesh IBGP," RFC 2796, 2000.
[4] J. Sobrinho, "An Algebraic Theory of Dynamic Network Routing," *IEEE/ACM Trans. Netw.*, vol. 13, no. 5, October 2005.
[5] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4," RFC 4271, January 2006.
[6] J. W. Stewart, *BGP4. Inter-Domain Routing in the Internet*. Addison Wesley, 1999.
[7] A. Flavel, "PhD Thesis, University of Adelaide," 2008.
[8] D. McPherson, V. Gill, D. Walton, and A. Retana, "Border Gateway Protocol (BGP) Persistent Route Oscillation Condition," RFC 3345, 2002.
[9] N. Feamster and J. Rexford, "Network-Wide Prediction of BGP Routes," *IEEE/ACM Trans. Netw.*, vol. 15, no. 2, pp. 253–266, 2007.
[10] T. Griffin and J. Sobrinho, "Metarouting," in *Proc. ACM SIGCOMM*, 2005.
[11] T. Griffin, F. B. Shepherd, and G. Wilfong, "The Stable Paths Problem and Interdomain Routing," *IEEE/ACM Trans. Netw.*, vol. 10, no. 2, pp. 232–243, 2002.
[12] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route Oscillations in I-BGP with Route Reflection," in *Proc. ACM SIGCOMM*, 2002.
[13] T. Griffin and G. Wilfong, "Analysis of the MED Oscillation Problem in BGP," in *Proc. ICNP*, 2002.
[14] T. Griffin and G. Wilfong, "An Analysis of BGP Convergence Properties," in *Proc. ACM SIGCOMM*, 1999.
[15] T. Griffin, F. B. Shepherd, and G. Wilfong, "Policy Disputes in Path Vector Protocols," in *Proc. ICNP*, 1999.
[16] L. Gao and J. Rexford, "Stable Internet Routing Without Global Coordination," *IEEE/ACM Trans. Netw.*, pp. 681–692, 2001.
[17] T. Griffin and G. Wilfong, "A Safe Path Vector Protocol," in *Proc. IEEE INFOCOM*, 2000.
[18] L. Gao, T. Griffin, and J. Rexford, "Inherently Safe Backup Routing with BGP," in *Proc. IEEE INFOCOM*, 2001.
[19] M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan, "How to Construct a Correct and Scalable iBGP Configuration," in *IEEE INFOCOM*, Barcelona, Spain, April 2006.
[20] N. Feamster and H. Balakrishnan, "Correctness Properties for Internet Routing," in *Proc. Forty-third Allerton Conference on Communication, Control, and Computing*, 2005.
[21] K. Varadhan, R. Govindan, and D. Estrin, "Persistent Route Oscillations in Inter-Domain Routing," *Computer Networks*, 2000.