Resolving BGP Disputes

Cheng Tien Ee
Vijay Ramachandran
Byung-Gon Chun
Scott Shenker

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-39
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-39.html

April 13, 2006

Resolving BGP Disputes

Cheng Tien Ee
UC Berkeley

Vijay Ramachandran
ICSI

Byung-Gon Chun
UC Berkeley

Scott Shenker
ICSI and UC Berkeley

***Abstract*— The Border Gateway Protocol (BGP) allows each autonomous system (AS) to select routes to destinations based on semantically-rich and locally-determined policies. This autonomously exercised policy-freedom can cause instability, where unresolvable policy-based disputes in the network result in interdomain route oscillations. Moreover, several recent works have established that such instabilities can only be eliminated by enforcing a globally accepted preference ordering on routes (such as shortest path). To resolve this conflict between policy autonomy and system stability, we propose a distributed mechanism that enforces a preference ordering only when oscillations due to these disputes occur. This preserves policy freedom when possible, and imposes stability when required.**

I. Introduction

The Border Gateway Protocol (BGP) [11] establishes connectivity between the independent networks, called *autonomous systems* (ASes), that comprise the Internet. BGP computes routes by a series of local decisions based on each ASes' individual routing policies. These policies are semantically rich in order to accommodate the complex rules that govern route choices in today's commercial Internet, such as business relationships and traffic engineering. However, this expressiveness in routing-policy configuration, coupled with ASes' freedom in implementing their policies autonomously, can cause instability in interdomain routing, as observed in [13].

The problem of understanding and preventing policy-induced routing anomalies has been the subject of much recent study. While some work characterized these anomalies using global models [6], [7], [12], other research proved that global and local constraints on policies could guarantee routing stability. The good and bad news from this literature are:

Good news: if the AS graph has an underlying business hierarchy and local policies obey sensible constraints arising from this hierarchy, then routing converges [5], [9]

Bad news: if ASes have complete freedom to filter routes (that is, exclude routes from consideration) then the only policies that are *a priori* guaranteed to converge are generalizations of shortest-path routing [3].

Thus, there are two choices: we can hope that natural business arrangements provide a stabilizing hierarchy, or we can remove all policy autonomy (but not filtering autonomy) by imposing some generalized form of shortest-path routing.

This paper advocates a "third way". Rather than rely on the vagaries of the marketplace to define a suitable hierarchy, or eliminate policy autonomy because of its potential to induce route oscillations, we propose a simple extension to BGP that constrains policy choices only after an oscillation is detected. Oscillations can be characterized by the presence of *dispute wheels* in the network [7], and our method provably finds and breaks dispute wheels. We tag each route advertisement with a *precedence value*, where a lower value corresponds to higher precedence. This goes at the top of the BGP decision process: available routes are chosen first based on their advertised precedence, with ties broken using the usual BGP decision process. The global precedence attribute changes only in the presence of a persistent oscillation; if there is no oscillation, we effectively use only the normal BGP decision process. Since configuration is not constrained unless absolutely necessary, ASes' freedom to decide on local policies is preserved.

We first review related work in Section II and then define and discuss dispute wheels in Section III. We describe the precedence metric and prove its ability to prevent dispute wheels in Section IV. Sections V and VI describe how this theoretical result can be put into practice. We evaluate the resulting algorithm via simulations in Section VII. Next, we discuss briefly the effect of misbehaving ASes on the rest of the network, before concluding in Section IX.

II. Related Work

Varadhan, Govindan, and Estrin [13] observed and documented persistent oscillations in BGP. The cause was not the policy configuration of one AS alone; they occurred because of interaction between the policies of several ASes. These anomalies occurred without any misconfiguration and were difficult to diagnose and resolve, because ASes tend to keep routing policies private.

Griffin, Shepherd, and Wilfong [7] introduced the Stable Paths Problem (SPP) as a formal model for BGP (and policy routing with path-vector protocols, in general). Using their framework, they were able to give a sufficient condition for protocol convergence, namely, the absence of *dispute wheels*. These structures characterize the conflicting policies of the nodes involved in a route oscillation (see the formal definition in Section IV). Unfortunately, the only known method to check for dispute wheels requires examining all the routing policies in a network, which is presently an impractical task. In addition, Griffin *et al.* showed that the problem of detecting whether a stable

routing exists, given all the policies in the network, is NP-complete. Worse yet, they showed that the existence of a stable solution does not automatically imply that a routing protocol can find it.

Gao and Rexford [5] showed that Internet economics could naturally guarantee route stability. A hierarchical business structure underlying the AS graph, along with policies that matched the various business agreements between ASes, is sufficient for protocol convergence. In this structure, assume that relationships between ASes are either *customer-provider*, *i.e.*, one AS purchases connectivity from another, or *peer-peer*, *i.e.*, two ASes mutually agree to transit traffic. No customer-provider cycles are allowed (*i.e.*, no AS, through a chain of providers, is an indirect customer of itself), and additional rules exist on how to set route preferences and when routes can be shared with other ASes. These assumptions capture the structure and economics of today's commercial Internet (but violations of these assumptions due to complex agreements, business mergers, or misconfigurations could induce route oscillation). These positive results were later confirmed by Gao, Griffin, and Rexford in [4], in which the combination of an underlying business structure and economically sensible policies was shown to prevent occurrences of dispute wheels, even when backup routing is allowed. Jaggard and Ramachandran [9] generalized this result but still required some assumption about the AS graph to prevent oscillations.

Dispute-wheel freeness and an AS business hierarchy are examples of *global constraints*, because they require that some condition is enforced involving the policies of many ASes at once[1]. In a highly decentralized environment like the Internet, enforcing such a global condition is unrealistic. Thus, later research attempted to find *local constraints*—conditions that could be checked individually for each AS—that are sufficient for route stability. However, results here were mostly negative. Sobrinho [12] and Griffin, Jaggard, and Ramachandran [6] proved that any dispute-wheel-free routing configuration is equivalent to a generalization of lowest-cost routing. This means that many seemingly sensible policies — in fact, all purely local policies not driven by some shared metric — could lead to oscillations. For example, it was shown that ASes could not use policies that always prefer routes through one neighbor over another—a type of policy commonly used today. Feamster, Johari, and Balakrishnan [3] further strengthened this result by showing that only generalizations of lowest-cost routing can guarantee stability while preserving the ability of ASes to *filter* routes (that is, to remove them from consideration). Thus, the theme of these results is that the only way to *a priori* guarantee stability is to essentially eliminate policy-configuration autonomy.

Most of these results exclude policies with *any possibility* of inducing routing anomalies, whether or not they actually do in a particular network. (This is because checking all the policies of a network is too difficult.) In this paper, we present an extension to BGP that detects oscillations and responds by breaking the corresponding dispute wheel. Griffin and Wilfong also presented such an algorithm, called SPVP, in [8]. Our protocol differs in several ways. First, SPVP's update-message size grows with the number of nodes in an oscillation, while our protocol simply adds a value to route advertisements. Second, SPVP records the changes in route choices due to the propagation of a route; this reveals more private policy information than necessary. Third, our protocol answers an open question left by [8], in that we present a minimal-impact solution to resolving disputes: our resolution algorithm requires little overhead and is engaged only when an oscillation is detected, and BGP is allowed to function normally otherwise. Our use of an extra attribute in route advertisements is similar to that in [4], [9]; however, those solutions still required a global constraint and preemptively excluded some oscillation-free policy configurations that our solution does not exclude.

III. Dispute Wheels

We begin by describing the notation used in this paper. We represent the network as the AS graph $G = (V, E)$, where each node $v \in V$ corresponds to one AS, and each edge $\{u, v\} \in E$ corresponds to an *BGP session* between ASes u and v, meaning that these ASes are physically connected and share route advertisements. We assume that links between ASes are reliable FIFO message queues with possibly arbitrary delays; this accounts for network asynchrony. At most one link is assumed to exist between ASes, and all the internal and border routers of an AS are condensed into one node (or one point of routing-policy control); this model captures eBGP, or interdomain connections[2].

A path P is a sequence of nodes $v_1 v_2 \cdots v_k$ such that $\{v_i, v_{i+1}\} \in E$; we write $v \in P$ if path P traverses node v. Paths can be concatenated with other nodes or paths; *e.g.*, if $P = u \cdots v$, $Q = v \cdots w$, and $\{w, d\} \in E$, we may write PQd to represent the path starting at node u, following P to node v, then following Q to node w, and finally traversing the edge (w, d). We assume that paths are directed from source to destination.

BGP essentially computes routes using the following iterative process: (1) Nodes receive *route advertisements* from their neighbors, indicating which destinations are reachable and by what routes; (2) for each destination, a node chooses the best route from those available, based on local policy; (3) if the current route to a given destination has changed, an advertisement is sent to neighboring nodes. We assume that these three steps occur as an atomic action

[1] In this paper, as is standard for BGP discussions, the term *global* really means "not purely local". A global value, for instance, is not one that necessarily all ASes share, but that applies to more than one AS.

[2] Analogous sufficient conditions for protocol convergence do not exist in the model that captures iBGP and MEDs. Therefore, in order to prove correctness, we restrict ourselves to the eBGP model in this paper. See [11] for the definition of these acronyms.

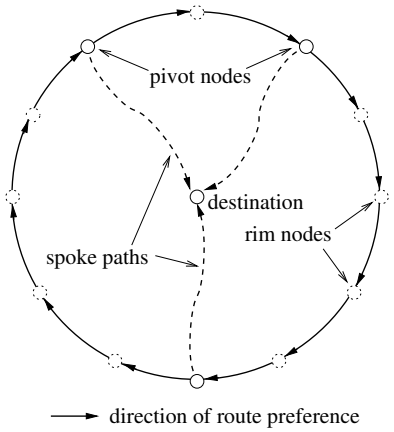


Fig. 1. *Example of a dispute wheel: elements of the wheel include the spoke paths, pivot nodes, and rim nodes.*

in response to receiving a route advertisement. The content of advertisements, or update messages, is also governed by routing policy; nodes are not required to share or consider all available routes—*i.e.*, routes may be *filtered*. The process begins when a destination advertises itself to its neighboring ASes; routes to that destination then propagate through the network as transit nodes choose routes and send updates. Because route choices are computed independently for each destination, we will assume, without loss of generality, that there is some fixed destination node $d \in V$.

We say the network has converged when each AS $v \in V$ is assigned a path $\pi(v)$ to the destination, such that the assignment is *stable* and *consistent*. By consistent, we mean that the paths form a forwarding tree to the destination; if $\pi(v) = vuP$, then $\pi(u) = uP$. By stable, we mean that $\pi(v)$ is the "best" available route for each node v, where "best" is determined by node v's routing policy; *i.e.*, if $\pi(v) = v\pi(u)$, there is no other node w such that the path $v\pi(w)$ is more preferred at v than $\pi(v)$. If the network is not *safe*, then there is some sequence of route updates that does not converge, in which every node gets a chance to update infinitely often (*i.e.*, no node is "shut out" in this sequence). Because there are only a finite set of route choices, such a sequence must be a route oscillation. The sequence may or may not be dependent on particular delays in update-message queues. A configuration is *safe* if any sequence of route updates, in which no node is shut out, converges.

Griffin, Shepherd, and Wilfong [7] showed that any such oscillation can be characterized by a *dispute wheel* in the network, shown in Figure 1. The dispute wheel captures the interaction between the routing policies of a set of nodes that are involved in a route oscillation. Formally, we have the following.

Definition 3.1: A *dispute wheel* is a set of nodes $p_0, p_1, \ldots, p_{k-1}$ called *pivots*, such that

1) at each pivot p_i, there exists a *spoke path* Q_i from p_i to the destination;
2) at each pivot p_i, there exists a *rim path* R_{i+1} to the next pivot p_{i+1} (assume all subscripts are modulo k);
3) each pivot prefers the path $p_i R_{i+1} p_{i+1} Q_{i+1} d$ over the path $p_i Q_i d$.

Note that the rim and spoke paths are not necessarily disjoint. We refer to non-pivot nodes along the rim paths R_i as *rim nodes*.

Because dispute wheels lie at the heart of BGP policy instabilities, we now walk through an example of BGP dynamics in the presence of a dispute wheel. Consider the four-node network shown in Figure 2.

In the figure, the paths considered by a node are listed in the shaded box next to that node; the more preferred path is on top. The oscillation is shown in Figure 3. (i) Assume that the destination node D sends an initial advertisement to nodes A, B, and C. (ii) Nodes A, B, and C then choose the direct paths to D and advertise their choices to nodes C, A, and B, respectively[3]. (iii) Upon receiving this advertisement, each node prefers the route through its neighbor, rather than the direct path to D, and chooses it. Doing so requires advertisement of these new paths;

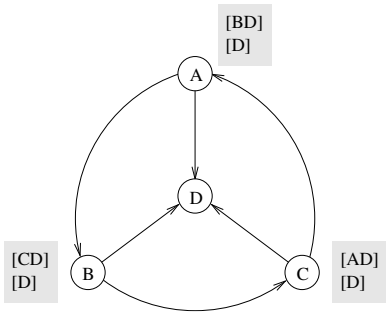


Fig. 2. *A simple dispute wheel: node D is the destination. Shaded boxes show route choices in order of preference.*

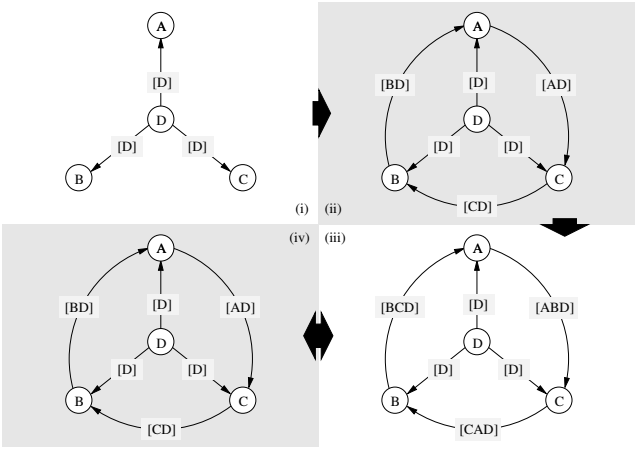


Fig. 3. *Simple example of dispute wheel oscillation: The simple local policy enforced at each node is the import filtering of routes with more than 2 hops. Routing oscillates between (iii) and (iv).*

[3]Assume, in this example, that advertisements are not propagated in the opposite direction.

with the longer paths selected, the direct paths to D are no longer advertised. (iv) When node A learns that node B has selected BCD, its preferred choice of ABD is no longer available; so, node A reverts to choosing the direct path to D. By symmetry, this happens at nodes B and C as well. This state is identical to the start; therefore, the sequence of route updates repeats, and nodes A, B, and C oscillate forever between their two route choices.

Any policy-induced oscillation can be characterized by the relationship of policies between nodes given by a dispute wheel; thus, the absence of dispute wheels is sufficient to guarantee that BGP is always safe. However, the presence of a dispute wheel does not necessarily guarantee an oscillation; even if it were possible to oscillate[4], BGP could non-deterministically converge[5]. Rather than exclude all potentially troublesome policy relationships *a priori*, the method we describe in the next section triggers a mechanism to resolve the corresponding dispute wheel whenever an oscillation is detected.

IV. The Precedence Metric

In this section we describe the precedence metric, which we subsequently show eliminates route oscillations due to dispute wheels.

Each route advertisement is tagged with a *global*[6] *precedence value* that is non-negative: a numerically greater value means lower precedence. We denote the precedence value, say v, associated with path P by (P, v). Each AS maintains a *history table* of observed route advertisements from its immediate neighbors. In this table, we associate every route with a *local precedence value* starting from 0. This local precedence value is obtained from the route's rank in the history table, and is determined via the usual BGP decision process. Thus the route in the ith entry of the table has a local precedence of $i-1$ and is preferred over all routes with precedence greater than that. Entries in this history table are strictly ordered, or ranked: no two routes of equal local precedence exist. Also, we augment BGP's decision process, prepending it with an additional step that selects routes of higher global precedence.

Suppose the winning route has an incoming global precedence of t, and a local precedence value of j. Then, the outgoing route advertisement is tagged with $t+j$. Thus, a route that is most preferred for all AS along its path is tagged with 0 at all hops. Figure 4 gives an example of this update process. Without loss of generality, we assume for the rest of this paper that the destination AS advertises routes with global precedence value of 0.

For distance-vector or link-state routing, usage of such a metric can result in permanent routing loops. In the case of path-vector protocols, the inclusion of all hops along the path removes the occurrence of such loops.

We show in the following section that this precedence metric prevents the formation of dispute wheels.

[4]That is, starting at some initial conditions would lead to an oscillation.

[5]For instance, a four node dispute wheel can converge into one of two stable configurations.

[6]Again, the term global only means that this precedence value has meaning across more than one AS, not that all ASes share this precedence value.

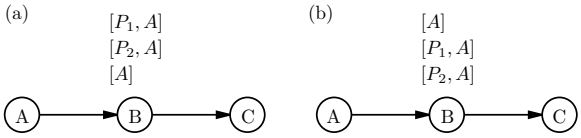


Fig. 4. *(a) AS B's preference for a direct route to destination AS A is the third in its history table. Propagation of this route to AS C will result in a lowering of its global precedence by 2. (b) AS B now considers the direct route to AS A to have the highest precedence. Route propagation to AS C will not alter the precedence value.*

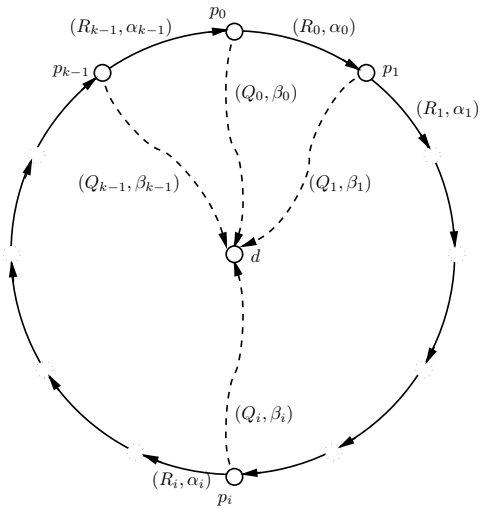


Fig. 5. *Dispute wheel illustration and notation used in our proof.*

A. Dispute Wheel Elimination

Proposition 4.1: If the precedence mechanism is prepended to the BGP decision process, then no policy-induced oscillations can occur.

Proof: Griffin, Shepherd, and Wilfong [7] prove that absence of dispute wheels is sufficient for safety, *i.e.*, that no policy-induced oscillations can occur. Therefore, it suffices to show that the precedence mechanism precludes dispute wheels. Using proof by contradiction, we begin by assuming that a dispute wheel exists.

Figure 5 is used to illustrate our proof, in which we consider a single destination d. Nodes $p_0, p_1, \ldots, p_{k-1}$ are the subset of nodes that are in the dispute wheel and have stable paths to the destination, that is, these are the pivot nodes. (Q_i, β_i) is the tuple consisting of Q_i, the spoke path from source p_i to destination d, and β_i, the precedence value associated with path Q_i. The tuple (R_i, α_i) on the other hand consists of the rim path R_i, which leads from p_{i+1} to p_i, and α_i, the change in precedence along R_i, including node p_{i+1}. In other words, if γ is the precedence value for path $R_i p_{i+1} Q_{i+1} d$, then $\gamma = \beta_{i+1} + \alpha_i$.

Suppose $p_0, p_1, \ldots, p_{k-1}$ each receive route advertisements from their immediate next hops along $Q_0, Q_1, \ldots, Q_{k-1}$ with global precedence values $\beta_0, \beta_1, \ldots, \beta_{k-1}$, respectively. Node p_i then selects the winning route Q_i, updates the value, and advertises that.

We next assume that the dispute occurs: node p_i prefers path $(R_i p_{i+1} Q_{i+1} d, \beta_{i+1} + \alpha_i)$, over route $(Q_i d, \beta_i)$. In Figure 5, this corresponds to each node picking its immediate neighbor in the clockwise direction as the next hop. In this proof, we assume, without loss of generality, that the route advertisements received and stored in the history table include those encountered during oscillations[7]. Note that we do not need *all* routes encountered during one oscillation period to be in the table, merely one that has higher local precedence than the stable spoke route. Then, the dispute wheel implies

$$\begin{aligned} \beta_1 + \alpha_0 &\leq \beta_0 \\ \beta_2 + \alpha_1 &\leq \beta_1 \\ &\vdots \\ \beta_0 + \alpha_{k-1} &\leq \beta_{k-1} \end{aligned}$$

Summing, we obtain

$$\begin{aligned} \sum_{i=0}^{k-1} \beta_i + \sum_{i=0}^{k-1} \alpha_i &\leq \sum_{i=0}^{k-1} \beta_i \\ \text{or} \quad \sum_{i=0}^{k-1} \alpha_i &\leq 0 \end{aligned}$$

Since, by definition, $\alpha_0, \alpha_1, \ldots, \alpha_{k-1}$ are non-negative, we have

$$\alpha_i = 0 \qquad \forall\, i$$

which implies that all nodes $p_0, p_1, \ldots, p_{k-1}$ locally prefer routes through $Q_0, Q_1, \ldots, Q_{k-1}$ respectively. This means that if the dispute wheel exists and each $R_i p_{i+1} Q_{i+1}$ is chosen over Q_i, it must be because of the precedence values. Thus, for the dispute wheel to form, we will require

$$\beta_0 < \beta_1 < \quad \cdots \quad < \beta_{k-1} < \beta_0$$

which is not possible. Therefore, by contradiction, no dispute wheel can exist. ■

Proposition 4.2: If there are non-zero precedence values advertised once the protocol converges, this must mean that dispute wheels exist.

Proof: Assume that the destination node advertises routes with precedence value 0. Thus, a non-zero value advertised somewhere means that there exists some node v with an incoming set S of routes of precedence value 0, $|S| > 0$, and an advertised route vP, $P \in S$, with positive precedence value. If this happens, then P must not be the most *locally preferred* route; suppose that route is Q. The precedence value of Q must be positive, otherwise v could have chosen it. This means there must be some node w along Q that increases its precedence value; w is similar to v, in that it must have some other path Q' with positive precedence value, causing it to choose Q. Thus, we can repeat this process at w and subsequent similar nodes. As the destination node is never encountered, because it always advertises routes with precedence value 0, we must ultimately encounter a node already traversed. The resulting cycle of nodes naturally form a dispute wheel that has been resolved using the precedence mechanism. ■

[7]Other routes will at most merely increase the precedence value, and not affect the correctness of the proof.

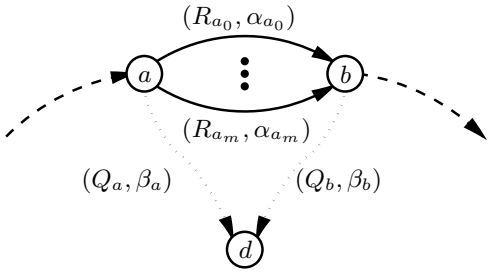


Fig. 6. *Multiple paths advertised by neighboring nodes can cause the global precedence value of a route to increase by more than 1.*

TABLE I
HISTORY TABLE OF NODE b IN FIGURE 6

Route	Global Precedence	Local Precedence
R_{a_0}	α_{a_0}	0
R_{a_1}	α_{a_1}	1
$\cdots$	$\cdots$	$\cdots$
R_{a_m}	α_{a_m}	m
Q_b	β_b	$m+1$

Corollary 4.3: From Propositions 4.1 and 4.2, positive global precedence values exist when routing converges if and only if dispute wheels causing oscillations exist.

Corollary 4.4: No dispute wheel with equal route precedence advertised at all hops can exist.

Proof: Assume that such a dispute wheel forms. Since the precedence value advertised by all nodes are the same, this implies that the route selected by each node is its most preferred. This in turn implies that the destination node must be part of the dispute wheel, which is a contradiction. ■

B. Autonomy Loss in Presence of Disputes

Corollary 4.3 showed that only the presence of dispute wheels can cause positive global precedence values to exist after routing converges. The increased value advertised by the pivot nodes depends on the number of paths advertised in parallel by immediate neighboring nodes.

We use Figure 6 to explain this. In the figure, node b has a spoke path Q_b to destination d. Assuming that b locally

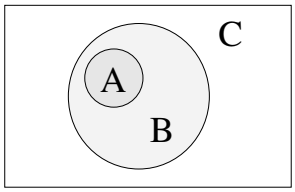


Fig. 7. *Region A is encompassed by nodes involved in a dispute wheel. Routes advertised in the external region B have global precedence values one higher than those in A. Similarly, if the nodes around the edges of B are in dispute, the global values in C will be one higher than those in B.*

prefers routes advertised by neighboring pivot node a along $R_{a_0}, R_{a_1}, \ldots, R_{a_m}$ compared to Q_b, we have the history table shown in Table I. Clearly, the advertised route will be $(bQ_bd, \beta_b + m + 1)$.

A non-uniform increase in global precedence values around the dispute wheel causes the rest of the network, *i.e.*nodes not in dispute and not along spoke paths, to lose autonomy. To correct this, instead of increasing the winning route's value by its local precedence, we bound the increase by 1. We call this the *precedence+* metric.

Proposition 4.5: Usage of the precedence+ metric eliminates oscillations caused by dispute wheels.

Proof: The following constraint is added to the proof of Proposition 4.1:

$$\alpha_i \leq m_i \qquad \forall\ i$$

where m_i is the total number of nodes along R_i, including n_{i+1} and excluding n_i. The rest of the proof follows. ■

Proposition 4.6: Usage of the precedence+ metric results in an increment in global precedence value at steady state only in the presence of dispute wheels that result in route oscillations.

Proof: Exactly the same as the proof for Proposition 4.2. ■

Corollary 4.7: From Propositions 4.5 and 4.6, the global precedence value increases by one if and only if a dispute wheel exists and causes routes to oscillate.

Precedence values can take on multiple non-negative values as opposed to just binary 0 or 1 values. With reference to Figure 7, the presence of a dispute wheel causes routes beyond the nodes in and within the wheel, that is, nodes in region B and not A, to be advertised with the same incremented value. Nodes in region B can still be in dispute, in which case the global precedence will be incremented again.

Corollary 4.8: Only nodes that prefer routes through nodes in dispute lose autonomy.

Proof: Trivial. ■

V. Practical Deployment

In Section IV-A, we assumed that the history table consists only of routes that are encountered in the presence of oscillations due to dispute wheels [8]. In practical scenarios, it is difficult to determine this set of routes from all routes ever seen. In many cases, we expect routes available during convergence to disappear when the network becomes relatively stable.

We propose using a *sliding history window* together with the history table to resolve this issue. Entries in the history table consists of routes received during the window period. Implementation-wise, each received route is stored in the history table together with an expiration time, which is the duration of the history window. If the route has not been updated when timeout occurs, in the sense that the same route had not been advertised within the expiration time, then it is removed. Thus, the history window removes routes that are seen initially but not thereafter.

To solidify our discussion, we describe the steps involved in this process. We divide time into periodic intervals, the duration of which is equivalent to the interval between route advertisements. To simplify matters we assume just one network destination. The following route update process runs at each time interval.

Expiring routes: Expired routes in the history table are removed.

Adding routes: Using the same rules as the decision process, the rank, or local precedence, of each new route in the history table is determined, and the route is inserted accordingly with its expiration time set to the maximum. If the route already exists in the history table, we reset its expiration time, and update its global precedence value if necessary.

Choosing routes: We prepend the usual BGP decision process with an additional step: we begin by selecting routes with the lowest global precedence values. Tie-breaking, if necessary, takes place using the decision process.

Proposition 5.1: If no dispute wheel exists, the network stabilizes with each node selecting its most locally preferred route. That is, there is complete autonomy of route selection.

Proof: We prove by induction as follows:
Base case: We assume that the destination node advertises its routes with global precedence value 0.
Inductive step: Assume that, in the recent past defined by the duration of the history window, all nodes receive advertisements from neighbors with global precedence values 0. This implies that all entries in the history table will also have value 0. Thus, each node is allowed to select any route, which results in all winning routes being advertised also with a value of 0. ■

Thus, with the history window and no dispute wheel present, routes selected with and without the precedence mechanism in place will be the same.

A. Route Fluctuation

Although we focus on dealing with policy-induced BGP oscillations, there are several other reasons the network

[8] Again, we need only at least one route with higher local precedence value than that of the stable spoke path.

might oscillate, such as router reboots, link fluctuation due to congestion or malfunctioning hardware. Consider an oscillation between a stable route and an unstable route (this language suggests a link flapping, but these could be any two routes the system is oscillating between); assume, without loss of generality, that the unstable route is preferred. Assuming that the history window is sufficiently large, and that changes are not due to policy conflicts, this results in route advertisements of precedence value 0 when the unstable route is up, and 1 otherwise. Thus, outgoing advertisements will fluctuate both in the route as well as the precedence value.

A further consideration is the implementation of route flap damping (RFD) in routers, the disadvantage of which is discussed in [10]. For non-policy-conflict-based oscillations, RFD can suppress updates up to a maximum of about 30 minutes. Although the primary objective is to cause the network to oscillate in "slow motion", it is detrimental to dispute resolution delay.

The key observation here is that RFD and precedence mechanisms are both unable to distinguish between oscillations due to policy conflicts and other causes. In this case, however, policy conflicts can be eliminated as a possible cause of instability.

VI. History Windows

In this section, we elaborate on the setting of the sliding history window size parameter, used to eliminate routes that are only present during routing convergence. When using the precedence metric, we say that the network has converged if both the routes and the advertised global precedence values remain stable. We assume that route advertisements and updates occur periodically, at intervals of W, which is our basic unit of window size. Thus, a window size of $4W$ is four times as along as the update period.

A. Simple Example

The sliding history window allows a node to remember more preferred, but unstable, routes along the dispute wheel, so that the global precedence value of the current winning route can be incremented. We demonstrate this using the simple network configuration given in Figure 2. Here, the local route precedence for each of the nodes is given. For instance, node A prefers the indirect route through B rather than the direct one to D. Figure 8 steps through the stabilizing process when the precedence metric is used.

Figure 8(i) Destination D advertises its reachability with a value of 0. With each node having only one route to the destination, the corresponding history tables each consists of only that route. Thus, it is also the most preferred locally.

Figure 8(ii) The routes are advertised with a value of 0, and are the most preferred for the next hop node. Thus, these routes have higher rankings in the history table and are selected.

Figure 8(iii) The most locally preferred routes are advertised, but are filtered by the next hops. For history windows longer than W, the most preferred routes will not have expired and thus still exist in the history table.

Figure 8(iv) The only acceptable route, which is the direct route, is less preferred and is advertised with an incremented value of 1. This is updated in neighboring history tables. Thereafter, the direct routes are always chosen, and the routing is stable.

From the simple example, we note that once convergence occurs, routes that are more locally preferred are always advertised with a positive global precedence value. This in turn causes the pivots to select the less locally preferred but stable spoke path. As a result, memory of the more locally preferred routes is no longer required.

B. An Adaptive History Window

From Section VI-A, we see that the history window is instrumental in increasing global precedence value, and thereby stabilizing the network. In general, the number of rim nodes, $\#_{rim}$, between two otherwise neighboring pivots can be thought of as delaying the route advertisements from one pivot to another by $\#_{rim}W$, so that the duration of the history window needs to be at least $\#_{rim}W$. However, the role of an AS, whether it is a pivot, rim or normal node, is difficult to obtain in practice. It is therefore even harder to determine the number of rim nodes and set the window size accordingly.

To resolve this issue, we use an adaptive history window that has the following properties:

Initialize It begins with a short duration, t_0, so that networks with small values of $\#_{rim}$ or not containing disputes can converge relatively quickly;

Grow Its duration increases when convergence does not occur after some time t_c, so that disputes involving large $\#_{rim}$ can be resolved; and

Contract Its duration decreases after the network stabilizes, since the history window is not required after convergence, and subsequent disputes formed due to topology changes need not be the same size or involve the same nodes as the previous ones.

There are two parameters of interest: the history window size t_w, and the period of time t_c over which stability of routes is assumed to mean convergence of the network. We now describe how each of these are set.

Window Size: Initial, Growth and Bound Clearly, t_w should begin at zero, so that networks without disputes or small values of $\#_{rim}$ can converge fast. We use an exponentially increasing window size to quickly resolve disputes involving large numbers of rim nodes. Since the number of rim nodes is unknown, the window size can potentially increase to a large value during convergence. This causes routes to expire later, possibly resulting in delayed convergence of the rest of the network.

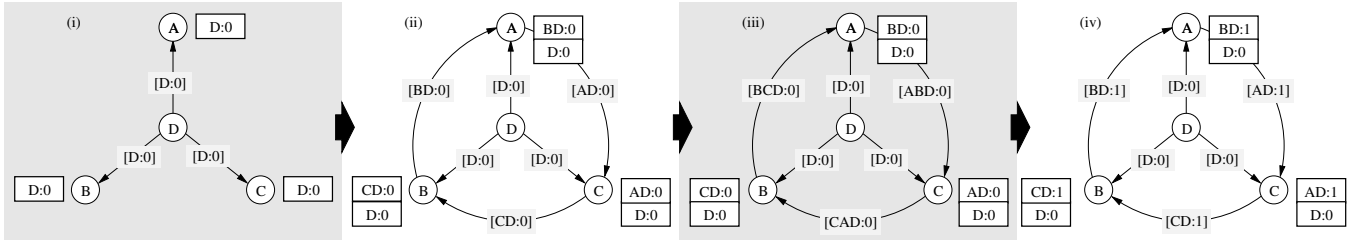


Fig. 8. *Dispute wheel formation and elimination: the simple local policy enforced at each node is the import filtering of routes with more than 2 hops. History tables for each node is shown in boxes beside them. With the precedence metric and the use of a history sliding window, the memory of the route with the highest local precedence causes the lower one to be advertised with a greater global precedence value in (iv). Thereafter, the routing stabilizes.*

TABLE II
VARIABLE DEFINITIONS FOR ALGORITHMS 1 AND 2

Variable	Description
L	route update interval
w_h	history window size, in units of L thus $t_w = w_h L$
w_c	current convergence period, in units of L thus $t_c = w_c L$
$w_{c,next}$	convergence period to use next
R_{win}	current winning route
$R_{win,old}$	previous winning route
max_AS_length	maximum AS path length of winning routes seen during current convergence period
$route\_change$	boolean indicator of whether route changed during current convergence period

To reduce the convergence time, we note that the path traversing the rim nodes must appear in one of the route advertisements received. Since in general it is difficult to determine this path, we simply upper bound the history window size using the maximum path length seen during the convergence period.

Convergence Period The second parameter of interest is the period of time t_c during which route changes are interpreted to mean the network is still converging. This period is necessary to determine if the history window size should be increased. Considering a dispute wheel without the presence of rim nodes, we see that the network is stable if there are no route changes from one iteration to the next. In the case where rim nodes are present, we need to ensure that there are no changes during the period of time routes are propagated through these nodes. Thus, we see that the convergence period can again be upper bounded by the maximum path length observed. We therefore choose $t_c = WL$, where L is that maximum path length and W is the route advertisement interval.

Algorithms 1 and 2 provide pseudo-code that govern the initialization and updating of variables associated with the history window and convergence period. Table II describes the notation used in the pseudo-code.

We evaluate the impact of this adaptive history window on convergence time in the following section.

Algorithm 1 History Window, Convergence Period Initialization

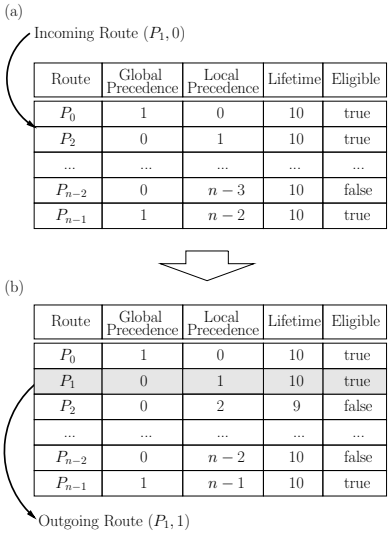
```
w_h ⇐ 0
w_c ⇐ 1
w_{c,next} ⇐ 1
R_{win,old} ⇐ ∅
max_AS_length ⇐ 0
route_change ⇐ FALSE
```

Algorithm 2 History Window, Convergence Period Update

```
if w_c ≡ 0 then
   w_c ⇐ w_{c,next}
   if route_change ≡ FALSE then
      w_{c,next} ⇐ 1
      max_AS_length ⇐ 0
      w_h ⇐ 0
   else
      route_change ⇐ FALSE
      w_{c,next} ⇐ max_AS_length
      if w_h ≡ 0 then
         w_h ⇐ 2
      else
         w_h ⇐ max(2w_h, max_AS_length)
      end if
   end if
else
   w_c ⇐ w_c − 1
end if
update history table, determine winning route R_win
if R_win ≠ R_{win,old} then
   route_change ⇐ TRUE
   max_AS_length ⇐ max(max_AS_length, length(R_win))
   R_{win,old} ⇐ R_win
end if
```

VII. SIMULATIONS

In this section we evaluate the performance of the precedence mechanism proposed in earlier sections. The primary parameters of interest are (1) network size and density, (2) size of dispute wheel, that is, the total number of nodes that are in dispute, (3) ratio of rim and pivot nodes, specifically, $\frac{\#rim+\#pivot}{\#pivot}$, and finally (4) whether an adaptive or static history window is used. The performance metric we use in all cases is the convergence time in units of route update intervals, *i.e.*W in Section VI. This interval can be thought of as the equivalent of the Minimum Route Advertisement



Route	Global Precedence	Local Precedence	Lifetime	Eligible
P_0	1	0	10	true
P_2	0	1	10	true
...	...	...	...	...
P_{n-2}	0	$n-3$	10	false
P_{n-1}	1	$n-2$	10	true

Route	Global Precedence	Local Precedence	Lifetime	Eligible
P_0	1	0	10	true
P_1	0	1	10	true
P_2	0	2	9	false
...	...	...	...	...
P_{n-2}	0	$n-2$	10	false
P_{n-1}	1	$n-1$	10	true

Fig. 9. *History table insertion and route selection process. (a) An incoming route is inserted at its appropriate location based on its rank relative to the other entries, using the rules in the BGP decision process. (b) We first select eligible routes with the least global precedence, and then amongst them the route with the highest rank, or smallest local precedence value.*

```
1: for each entry in history table do
2:    decrement lifetime
3:    remove entry if lifetime is zero
4: end for
5: for each route R received do
6:    if route from neighbor N is filtered then
7:       set last eligible route from N ineligible
8:    else if different route received from neighbor N then
9:       set previous route from N ineligible
10:      compute R's local precedence
11:      insert R into history table
12:      reset R's lifetime
13:      set R eligible
14:   else if same route received from neighbor then
15:      if previous eligible route R' not equal R then
16:         set R' ineligible
17:      end if
18:      update R's global precedence value
19:      reset R's lifetime
20:      set R eligible
21:   else if first route R is received from neighbor then
22:      compute R's local precedence
23:      insert R into history table
24:      reset R's lifetime
25:      set R eligible
26:   end if
27: end for
28: select set S of eligible routes
29: select set S' with lowest global precedence from S
30: select route R with lowest local precedence from S'
31: return R
```

Fig. 10. *Pseudo-code for updating routes in the history table and determination of the winning route.*

Interval (MRAI) used in routers today[9]. We describe the general setup of the simulations below.

A. Experimental Setup

The simulator is event-based, with all events synchronized, and we assume that all route update intervals are equal. Figure 9 illustrates the update and selection processes that take place within each router, and Figure 10 provides the steps that occur at each iteration for every destination prefix. We simulate the use of the precedence+ metric, since it is shown in Section IV-B to have significant advantages over the basic precedence metric. Two types of graphs are used in our simulations, namely simple and power-law. We describe them below.

1) Simple Graphs: Simple graphs consist only of rim, pivot and destination nodes, which are immediate neighbors of pivot nodes. Figure 2 shows an example of a simple graph. Whilst these graphs are not representative of a real network in general, they are still useful in determining properties of a dispute wheel.

2) Power-law Graphs: These are generated using the BRITE topology generator [1] and the Barabasi-Albert model [2]. This model implements preferential connectivity, which refers to a new node's inclination to connect to existing nodes that are more connected, and incremental growth, where nodes are gradually added to an existing network. We alter the network density by adding differing number of edges when a new node is introduced.

To deterministically generate disputes, we make use of an additional data structure in our simulation that may be of different form in real-life deployments. We call this the *preference table*, which is local to each node. An entry exists for each neighbor of the node, and routes from neighbors of higher ranks in this table are more locally preferred than those of lower[10]. To break ties between routes sent from the same neighbor but at different iterations, we use AS path length, where a shorter path length is more preferred, as well as lexical order.

Next, to create a dispute wheel of a particular size, we randomly pick a node in the graph and perform depth-first search. The search terminates when a previously encountered node is reached, and the length of the path is that of the desired wheel size. Then, at each node in the dispute wheel, we set the next node along the wheel to have the highest rank in its preference table. For the rest of the nodes and entries, the ranks of neighboring nodes are set randomly. This ensures that there is at least one dispute wheel present but not strictly one, since the random assignment of local preferences may cause others to form.

B. Results

We begin this section by observing the convergence process for various sizes of dispute wheels, as well as for different ratios of rim to pivot nodes.

Figure 11 shows the minimum convergence times pos-

[9]The MRAI is by default set to 30 seconds, and is used to reduce the computation load at a router.

[10]This can be thought of as the local preference rule in BGP's decision process.

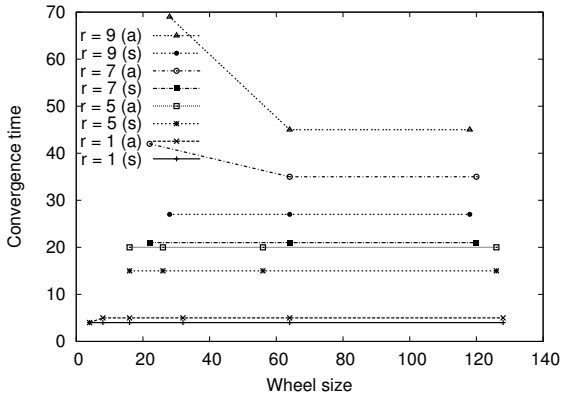


Fig. 11. *Comparison between static and adaptive history window sizes. Simple graphs.* (a) *denotes results are obtained from using adaptive windows, whereas graphs labeled* (s) *use static windows.* r *is the ratio* $\frac{\#rim+\#pivot}{\#pivot}$.

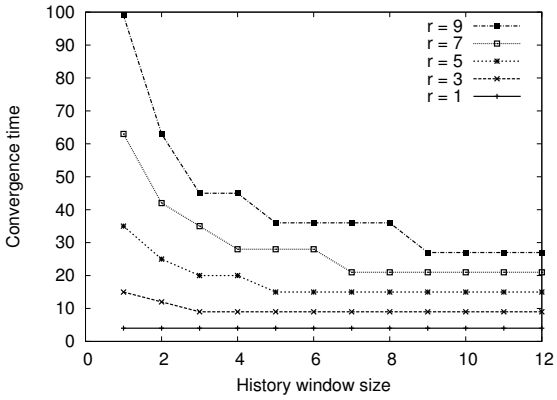


Fig. 12. *Comparison of varying ratios* $r = \frac{\#rim+\#pivot}{\#pivot}$. *Simple graphs.*

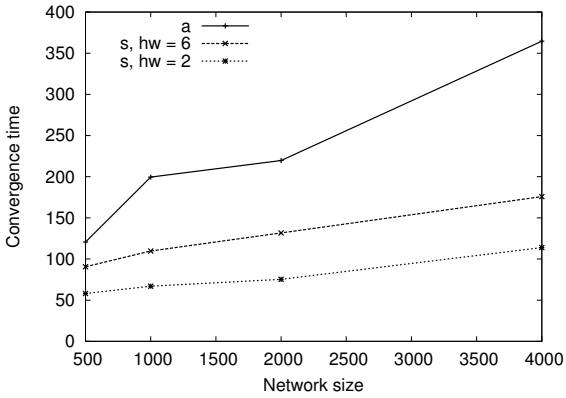


Fig. 13. *Comparison of varying history window sizes. Power-law graphs, sparse network.* s *and* a *denote results for static and adaptive history windows respectively;* hw *gives the static history window size.*

sible[11] for varying number of nodes in dispute, ratios of rim to pivot nodes, as well as static and adaptive history windows. We elaborate on them below.

Static windows We begin by looking at the graphs for static windows, labeled (s). Although in practice it is difficult to obtain the number of rim nodes and set the history window sizes accordingly, we still investigate the effect of rim nodes on convergence time to gain better understanding of its impact. Each graph shows the dependence of convergence time on different ratios r of rim to pivot nodes. Clearly, we see that (1) convergence time is a function only of the r and not the wheel size; (2) the increase in convergence time is approximately linearly proportional to r; and finally, from additional simulations performed but not explicitly shown, (3) convergence time of the entire network is determined solely by the maximum number of rim nodes between any two otherwise neighboring pivots. Since we assume that the route update interval is uniform for the network, this implies that the convergence time is thus dependent on the maximum delay between any consecutive pivot nodes.

Adaptive windows We next compare adaptive and static history windows. Adaptive windows are initialized and updated using Algorithms 1 and 2. From the figure, we see that having adaptive window sizes results in performance that is relatively close to the optimal case. Even though these window sizes are likely to overshoot their optimal values, the resulting convergence times do not increase significantly.

Next, we investigate the effect of varying static history window sizes, w_h, in Figure 12. We note that (1) convergence time generally increases with the ratio of rim to pivot nodes; and (2) the optimal w_h resulting in the lowest convergence time is equal to the ratio itself; and (3) an increase in window size beyond a certain point results in no further reduction in convergence time. This suggests initially that overestimating w_h is less detrimental than underestimating it. However, this observation is not conclusive since simple graphs are used, where the pivot nodes are immediate neighbors of the destination and thus usage of AS path length is a relatively good estimate of the maximum number of consecutive rim nodes. We return to this discussion after evaluating power-law graph results.

Power-law graphs are used for the networks from which we obtain the results in Figures 13 and 14. Each data point is the average of 30 simulation runs, except in the case of static windows, where we ignore runs that do not converge when the window sizes are set too low. Out of a total of 240 runs, 13 did not converge when using static windows. On the other hand, usage of adaptive windows always results in convergence. For the sparse network, two edges are used to connect an additional node to the network, and twice as many edges are used in the case of the dense network. From the two figures, we can see that (1) denser or larger networks takes slightly more time to converge, which is

[11]We vary the static window sizes, and use that which results in the least convergence times.

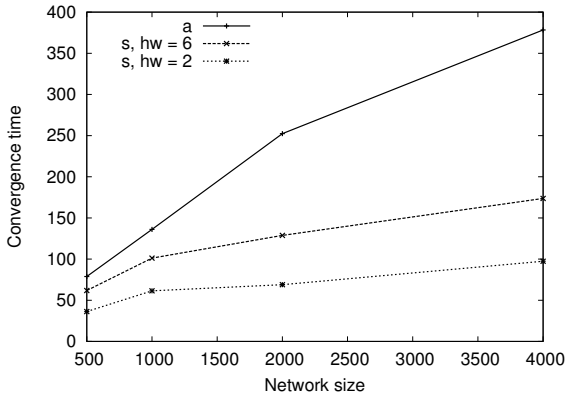


Fig. 14. *Comparison of varying history window sizes, Power-law graphs, dense network. s and a denote results for static and adaptive history windows respectively; hw gives the static history window size.*

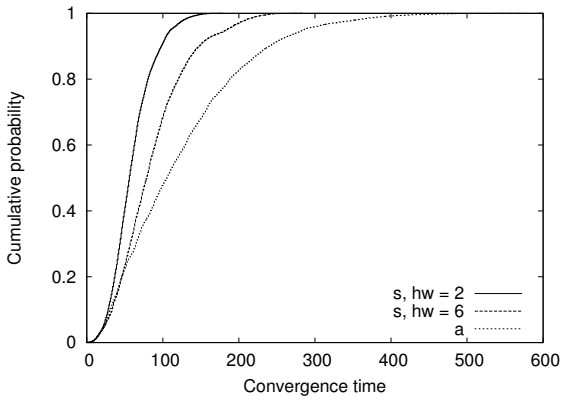


Fig. 15. *CDF of varying history window sizes. Power-law graphs, sparse network. s and a denote results for static and adaptive history windows respectively; hw gives the static history window size.*

expected as more paths are available and explored before routing stabilizes; (2) larger history windows also delay network convergence, since transient routes linger in the history table for a longer period of time before expiring; finally, (3) although usage of adaptive windows appear to increase convergence time compared with static windows, this is not a wholly valid comparison since static windows can still result in route oscillations.

We next examine the distribution of convergence time in power-law graphs. Figure 15 shows that a change in static window size from 2 to 6 results in the maximum time-to-converge increasing from 150 to about 230 iterations. The graph is also more spread out: for history window size of 2, we have 80% of nodes converged within 80 iterations, whereas for $w_h = 6$ it took around 120. For adaptive windows, 80% of nodes converge within 190 iterations. Also, the total convergence time of the network using adaptive windows is around 480 iterations. Again, we note that the comparison is not wholly correct since usage of static windows can still result in route oscillations.

Figure 16 shows results obtained from a denser network.

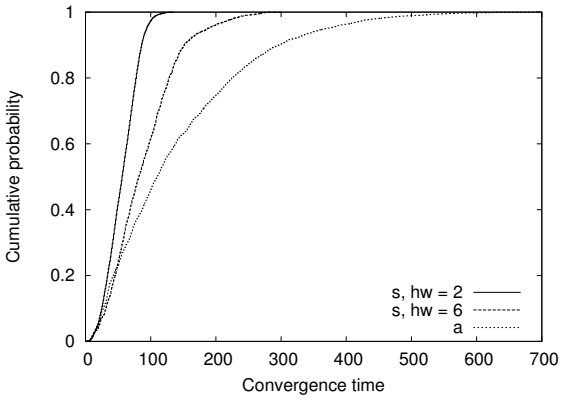


Fig. 16. *CDF of varying history window sizes. Power-law graphs, dense network. s and a denote results for static and adaptive history windows respectively; hw gives the static history window size.*

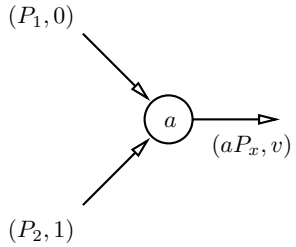


Fig. 17. *Basic scenario used to describe misbehavior: node a receives two routes and advertises one. Detection of misbehavior can be performed by observing incoming and outgoing routes.*

We observe that the graphs are similar to their counterparts in Figure 15. This shows that the network convergence time does not significantly change with network density when using the precedence metric.

C. Results Summary

The main take-aways from these simulation results are

- static windows cannot result in routing convergence;
- the minimum history window size that results in network convergence is dependent on the maximum number of consecutive rim nodes in a dispute wheel;
- slight overestimation of the optimal history window is preferred to underestimation,
- larger history windows increase network convergence time, and finally
- network convergence time remains relatively unchanged when using the precedence+ metric.

VIII. Misbehavior

In this section we discuss the impact of misbehaving nodes on the rest of the network.

Since the global precedence metric can in general restrict the autonomy of an AS, there may be incentives for not adhering to the general rule. We discuss various ways whereby nodes can misbehave, and detection methods which rely on the ability to observe the incoming and outgoing routes of an AS.

A. Simple Example

Clearly, one type of misbehavior is the selection of an available route with the highest local precedence regardless of its global value. We describe several scenarios using Figure 17, focusing on the routes advertised from a.

- $(P_2, 0)$: there is definite misconduct, since the outgoing route's precedence is less than its incoming's. This is true even if a filters $(P_1, 0)$.
- $(P_2, 1)$: there is no misconduct only if a permanently filters route $(P_1, 0)$. In this case, route $(P_2, 1)$ is the only incoming route and therefore also the most locally preferred. Thus, the outgoing route's precedence is not incremented.
- (P_x, v) where $v > 2$ for $x = 1$ and $x = 2$. In this case, node a is artificially increasing the outgoing precedence value. This has the effect of not allowing upstream ASes to select a route traversing this AS. While some may construe this as misbehavior, it may be used as a means of indicating that certain links are used as backup. For instance, the destination node can advertise a global precedence value of 1 on backup links, and 0 on normal links.

From this simple example, we can determine that an AS is misbehaving if one of these two conditions are satisfied: (1) an outgoing route has a global precedence value that is less than its corresponding incoming route, or (2) an outgoing route has a global precedence value that is greater than its corresponding incoming route by more than one.

B. Adaptive Filtering

Misbehavior that is more difficult to detect involves *adaptive filtering*, which we now describe. Let M be the node representing a misbehaving AS. Clearly, if M is always filtering its spoke path, it will never become a pivot node, and thus cannot influence the convergence process. However, M involved in a dispute can initially accept routes from neighbors along the spoke and wheel. When routing stabilizes and the precedence metric forces selection of the spoke path, M can subsequently decide to effectively filter that in order to select the locally preferred path along the wheel.

In this case, two scenarios can occur as illustrated in Figure 18. In part (a), the total number of pivot nodes in dispute is an odd number. The selection of a next hop that is more locally preferred but having a higher global precedence value eventually results in M not having a valid route. Subsequent removal of the filter causes the system to oscillate again.

In part (b), an even number of pivot nodes can cause the system to settle in a stable state even if M misbehaves. In this case, M is able to use the path it locally prefers.

In general it is difficult to determine the number of pivot nodes in dispute, and therefore hard to know if the implementation of adaptive filtering in M can result in oscillations (which ultimately does not benefit M).

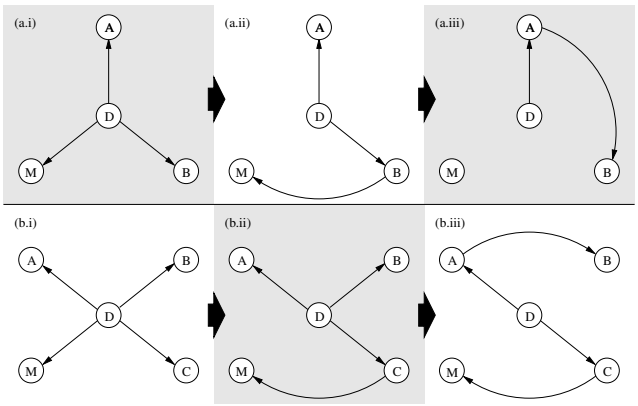


Fig. 18. *A misbehaving AS, represented by node M, can have differing effects on the network.* **(a)** *For a dispute wheel with an odd number of nodes, M eventually lacks a route if it initially filters the spoke one.* **(b)** *For a wheel with an even number of nodes, M does not destabilize the network.*

Furthermore, we believe that the alternative, though less preferred, routes are still acceptable by the ASes, otherwise these would have been filtered.

In summary, many misbehaviors can be detected through monitoring. Undetectable misbehaviors may or may not give the cheating AS a better stable route (depending on the dispute wheel configuration) and, even if it does, it does not greatly harm other ASes. However, we don't claim to fully understand all the incentive issues inherent in our approach, and it is the subject of future study.

IX. Conclusion

This paper tries to reconcile two desirable, but seemingly incompatible, goals. On the one hand, it is a business reality that ASes would like to set policies according to their own specialized needs — whether these arise out of business, or traffic engineering, or other concerns — and they would like to keep these policies private. On the other hand, every AS would like to have a stable Internet, where routes didn't oscillate. Unfortunately, recent theoretical results make clear that to ensure *a priori*, without knowing the policies beforehand or relying on assumptions about the structure of business relationships, that routing will be stable, ASes must be deprived of essentially all policy autonomy.

In this paper we no longer require an *a priori* guarantee, but instead seek to remove policy-induced oscillations when they arise. This allows us to preserve policy freedom when possible, and impose stability when required. While more experimentation will be required to fully validate our proposal, our theoretical and simulation results suggest that this might be a promising approach.

References

[1] *Boston University Representative Internet Topology Generator (BRITE).* http://www.cs.bu.edu/brite.

[2] A. Barabasi and R. Albert. Emergence of Scaling in Random Networks. Science, October 1999.

[3] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA, 2005. ACM Press.

[4] L. Gao, T. G. Griffin, and J. Rexford. Inherently safe backup routing with bgp. In *Proceedings of IEEE INFOCOM 2001*. IEEE Computer Society, IEEE Press, April 2001.

[5] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Netw.*, 9(6):681–692, 2001.

[6] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design principles of policy languages for path vector protocols. In *SIGCOMM '03: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 61–72, New York, NY, USA, 2003. ACM Press.

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

[8] T. G. Griffin and G. Wilfong. A safe path vector protocol. In *Proceedings of IEEE INFOCOM 2000*. IEEE Communications Society, IEEE Press, March 2000.

[9] A. D. Jaggard and V. Ramachandran. Robustness of class-based path-vector systems. In *Proceedings of ICNP'04*, pages 84–93. IEEE Computer Society, IEEE Press, October 2004.

[10] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz. Route flap damping exacerbates internet routing convergence. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 221–233, New York, NY, USA, 2002. ACM Press.

[11] Y. Rekhter, T. Li, and e. Susan Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, January 2006.

[12] J. L. Sobrinho. An algebraic theory of dynamic network routing. *ACM/IEEE Transactions on Networking*, 13(5):1160–1173, October 2005.

[13] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32(1):1–16, March 2000.