Resolving Inter-Domain Policy Disputes

Cheng Tien Ee
Vijay Ramachandran
Byung-Gon Chun
Kaushik Lakshminarayanan
Scott Shenker

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2007-27
http://www.eecs.berkeley.edu/Pubs/TechRpts/2007/EECS-2007-27.html

February 11, 2007

Resolving Inter-Domain Policy Disputes

Cheng Tien Ee*, Vijay Ramachandran†, Byung-Gon Chun*, Kaushik Lakshminarayanan‡, Scott Shenker*§
*UC Berkeley, †Stevens Institute of Technology, ‡India Institute of Technology, Madras, §ICSI

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. 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 disputes resulting in oscillations exist. This preserves policy freedom when possible, and imposes stability when required.**

I. Introduction

The Border Gateway Protocol (BGP) [12] establishes connectivity between the independent networks, called *autonomous systems* (ASes), that together form 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 manifesting in the form of persistent route oscillations [17].

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 [7], [8], [14], other research proved that global and local constraints on policies could guarantee routing stability. The good and bad news from this literature can be summarized as follows:

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], [10].

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 [2].

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 [8], and our method provably finds and breaks dispute wheels, including those involving non-strict preferences. 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 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 §II and then define and discuss dispute wheels in §III. The precedence metric is described next and its ability to prevent dispute wheels proven in §IV. §V and VI describe how this theoretical result can be put into practice. We evaluate the resulting algorithm in §VII and discuss several issues in §VIII before concluding in §IX.

II. Related Work

Varadhan, Govindan, and Estrin [17] were the first to discuss the possibility of persistent route 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 since ASes tend to keep routing policies private.

Griffin, Shepherd, and Wilfong [8] 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 §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 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, it is assumed 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, although violations of these assumptions due to complex agreements, business mergers, or misconfigurations can still 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 [10] 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] However, policy autonomy is at the heart of the philosophy that led to BGP, and ISPs will be loathe to relinquish it. Accordingly, later research attempted to find *local constraints*—conditions that could be checked individually for each AS—that are sufficient for route stability. Unfortunately, results here were mostly negative. Sobrinho [14] and Griffin, Jaggard, and Ramachandran [7] 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 risk oscillations if they use policies that always prefer routes through one neighbor over another—a type of policy commonly used today. Feamster, Johari, and Balakrishnan [2] 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). Overall, 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 determining whether the network policies will result in oscillations 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 [9]. Our protocol differs in several ways. First, SPVP records the changes in route choices due to the propagation of a route; this reveals more private policy information than necessary. Second, our protocol answers an open question left by [9], in that we present a minimal-impact solution to resolving disputes: our resolution algorithm is engaged only when an oscillation is detected, and BGP is allowed to function normally otherwise. Third, SPVP's update-message size grows with the number of nodes in an oscillation, while additional fields used by our protocol scales with the *number of resolved disputes* encountered along a path. This is similar to that in [4], [10]; however, those solutions still required a global constraint and preemptively excluded some oscillation-free policy configurations that our solution does not exclude.

Another class of runtime solution involves diffused computation [1], which uses the observation that, as long as a change in path results in reception of another with a local preference value at least as high as that of its current path, then stability is guaranteed. In this case, an AS is required to ask any other AS whose path currently traverses it if a change in path is acceptable. Such a solution would restrict a provider's route choices based on inputs from customers, which is typically not the case in practice.

Finally, we allow ASes to exercise full autonomy *unless* the particular set of policies and topology results in an oscillation, and in that case, and only in that case, AS autonomy is revoked. What distinguishes this from much of the previous literature is that it does not place *a priori* restrictions on ASes, only *post hoc* restrictions. This enables a far greater degree of freedom, and we believe that ASes might be willing to accept the limitations as the price to pay for stability.

III. Dispute Wheels

We begin by describing the notation used in this paper. The network is represented 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 a *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 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).

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

[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.

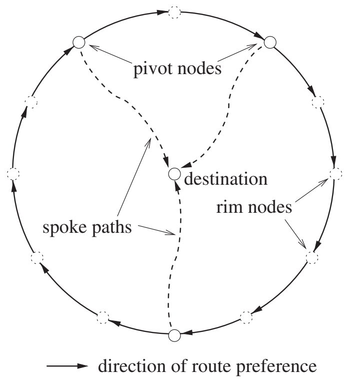


Fig. 1. *Example of a dispute wheel: elements of the wheel include the spoke paths, pivot nodes, and rim nodes.*

finally traversing the edge (w, d). We assume that paths are directed from source to destination.

BGP, at a schematic level, 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. 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 focus our attention on, without loss of generality, on a single 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*, *consistent* and *safe*. 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, given the other nodes' path assignments, where "best" is determined by node v's routing policy; that is, 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 $\pi(v)$.

Safety is slightly more subtle. By unsafe, we meant that there is some sequence of route updates that does not converge, in which every node gets a chance to update infinitely often. 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 receiving route updates. A configuration is *safe* if any sequence of route updates, in which no node is shut out, converges.

Griffin, Shepherd, and Wilfong [8] showed that any such oscillation can be characterized by a *dispute wheel* in the network, shown in Figure 1. The dispute wheel captures the

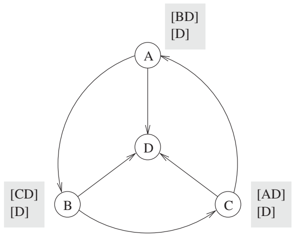


Fig. 2. A simple dispute wheel: node D is the destination. Shaded boxes show route choices in order of preference.

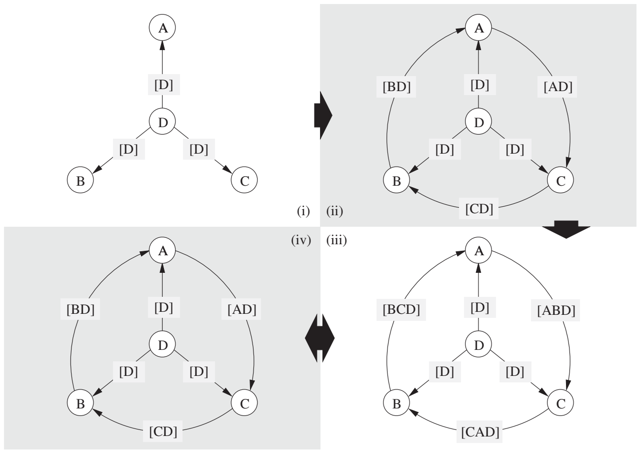


Fig. 3. Simple example of dispute wheel oscillation: The simple local policy enforced at each node is the import filtering of routes with more than 2 hops. Routing oscillates between (iii) and (iv).

interaction amongst 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 $p_0, p_1, \ldots, p_{k-1}$ (assume all subscripts are modulo k) 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};
3) each pivot prefers the path $p_iR_{i+1}p_{i+1}Q_{i+1}d$ over the path p_iQ_id.

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*.

Since 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, paths considered by a node are listed in the shaded box next to that node in decreasing order of preference. 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.(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; 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 occurs at nodes B and C as well. This state is identical to (ii); 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 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 there are some initial conditions that will lead to an oscillation, BGP could non-deterministically converge.[2] 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

We begin by augmenting BGP's decision process, prepending it with an additional step that utilizes a new metric which we call the precedence metric. We describe this metric below, and show that it eliminates route oscillations due to dispute wheels.

Each route advertisement is tagged with a *global*[3] *precedence value* that is non-negative: a numerically greater value translates to a lower precedence. We denote the precedence value, say v, associated with path P by $(P{:}v)$. Each AS maintains a *history* of observed route advertisements from its immediate neighbors. In this history, we associate every route with a *local precedence value* starting from 0. This local precedence value is obtained from the route's rank, and is determined via the usual BGP decision process. Thus the route ranked ith has a local precedence of i-1 and is preferred over all routes with local precedence greater than that. Strict ranking is performed, such that no two routes of equal local precedence exist.

Suppose the selected 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 ASes 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. We next show that

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

[3]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.

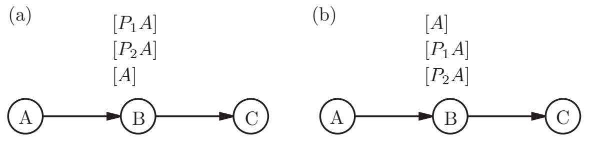


Fig. 4. *(a) AS B's preference for a direct route to destination AS A is ranked third. Propagation of this route to AS C will result in a lowering of its global precedence by 2. (b) AS B now considers the direct route to AS A to have the highest precedence. Route propagation to AS C will not alter the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

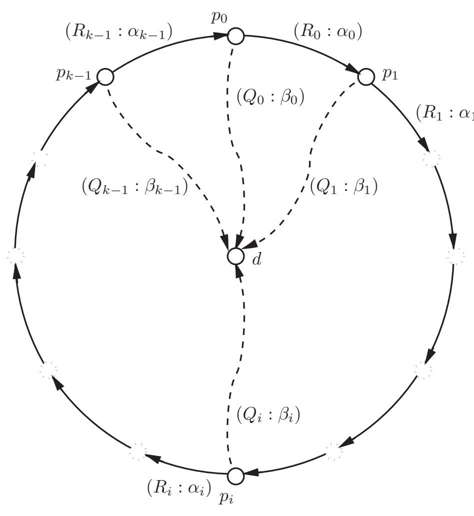


Fig. 5. *Dispute wheel illustration and notation used in our proof.*

this precedence metric prevents the formation of dispute wheels.

A. Dispute Wheel Elimination

Proposition 4.1: If routes encountered during previous policy-induced oscillations are stored and the precedence metric is used, then no further policy-induced oscillations can occur.

Proof: It is proven in [8] that the absence of dispute wheels is sufficient for safety, and hence 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. $(Q_i{:}\beta_i)$ 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 $(R_i{:}\alpha_i)$ 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 γ=β_{i+1}+α_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 route Q_i, updates the value, and advertises that.

We next assume that the dispute occurs: node p_i prefers path ($R_i p_{i+1} Q_{i+1} d$:β_{i+1}+α_i), over route ($Q_i d$:β_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 that the route advertisements received and stored as part of the history include those encountered during oscillations.[4] Note that we do not need *all* routes encountered during one oscillation period to be stored, 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

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

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 global precedence values.

Thus, for the dispute wheel to form, we will require

$$\begin{aligned} \beta_i + \alpha_{i-1} &< \beta_{i-1} \quad \forall\, i \\ \text{or} \quad \beta_{k-1} < \beta_{k-2} < &\cdots < \beta_0 < \beta_{k-1} \end{aligned}$$

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, and that the network has converged. 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 would 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 global precedence, 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. ■

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

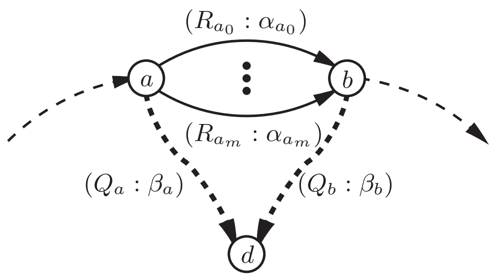


Fig. 6. *Multiple paths advertised by neighboring nodes can cause the global precedence value of a route to increase by more than 1.*

TABLE I
HISTORY OF NODE b IN FIGURE 6

Route	Global Precedence	Local Precedence
$R_{a_0} a Q_a d$	$\alpha_{a_0} + \beta_a$	0
$R_{a_1} a Q_a d$	$\alpha_{a_1} + \beta_a$	1
$\cdots$	$\cdots$	$\cdots$
$R_{a_m} a Q_a d$	$\alpha_{a_m} + \beta_a$	m
$Q_b d$	β_b	$m+1$

Corollary 4.3: From Propositions 4.1 and 4.2, global precedence values greater than that advertised by the destination exist when routing converges if and only if dispute wheels causing oscillations exist.

Corollary 4.4: A route traversing resolved disputes cannot advertise the same global precedence at all hops.

Proof: Assume that such a route exists. Since the precedence value advertised by all hops 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 pivot nodes.

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

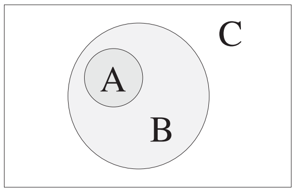


Fig. 7. *Region A is encompassed by nodes involved in a dispute wheel. Routes advertised in the external region B have global precedence values one higher than those in A. Similarly, if the nodes around the edges of B are in dispute, the global values in C will be one higher than those in B.*

state shown in Table I. Clearly, if the spoke path is selected, it will be advertised as (bQ_bd:β_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 selected 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:

$$\alpha_i \leq r_i \qquad \forall\ i$$

where r_i is the total number of nodes along R_i, including p_{i+1} and excluding p_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: Same as that 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 may lose autonomy.

Proof: Trivial. ■

For the rest of this paper, we focus solely on the precedence+ metric.

C. Accounting for Non-Strict Preferences

The precedence+ metric is proven to eliminate dispute-based oscillations for *strict preferences*; that is, routes can be ranked independent of others. In general, preferences are non-strict, and are encountered for instance in BGP's Multi-Exit Discriminators (MEDs). In this subsection we propose a minor extension to account for this.

The primary effect of having non-strict preferences is that an incoming route R_i causes route R_{cs} to be selected, where $R_i \neq R_{cs}$ and R_{cs} is not the previous route selected (R_{ps}). This is an *Independent Route Ranking (IRR)* violation [11]. In terms of strict preferences, it appears as though the existence of R_i results in the disappearance of R_{ps} from the most locally preferred rank. Thus, to capture this as part of the history of routes encountered, we associate a logical route R'_{ps} with R_i, and comparison of R'_{ps} with any other route should ignore the presence of R_i. This slight tweak is necessary for computing the local preference of the selected route. Since the goal is to determine if the global precedence should be incremented, we will be comparing R'_{ps} with R_{cs}, ignoring R_i. Since $R_i \neq R_{cs}$, we will not encounter the scenario when the two will be compared. In the case where R_{cs} becomes unavailable in the future and is replaced by R_i, we evict R'_{ps}.

V. From Theory To Practice

In §IV, we showed that usage of the precedence+ metric, coupled with the knowledge of routes encountered during oscillations, can cause the network to converge. The primary difficulty in implementing the solution is knowing precisely the relevant set of routes encountered during oscillations and not others. In this section we describe how this is achieved in practice. We begin by defining our goals:

One: We distinguish between transient and permanent oscillations, where the former disappear with the convergence of the network. The association of routes with disputes should be removed if the latter is found to be transient. Further, changes in network topology affecting resolved disputes should cause the removal of stored state associated with those disputes.

Two: The solution should not reveal any ISP policies.

Three: Only local information associated with incoming advertised routes is necessary; no global knowledge is required.

Four: Knowledge of potential pivot nodes should be provided as feedback by the protocol. The presence of resolved disputes causes precedence values to increase, thereby possibly restricting the choices of routes. In general we believe it is preferable to react by altering the local preferences at a subset of the pivot nodes so that disputes do not arise in the first place and route choices become unconstrained. Since access to the global view is not assumed and is probably unattainable, we seek an alternative means of identifying the potential pivots.

Our complete Precedence Solution consists of two main phases: detection and storage. We detect routes involved in oscillations, then store and maintain their associated information so that previously encountered disputes remain resolved and corresponding oscillations do not reoccur. *As discussed later in §VIII, if AS policies are based solely on next hop neighbors, then only the first phase, detection, is required, and the overall solution is simplified.*

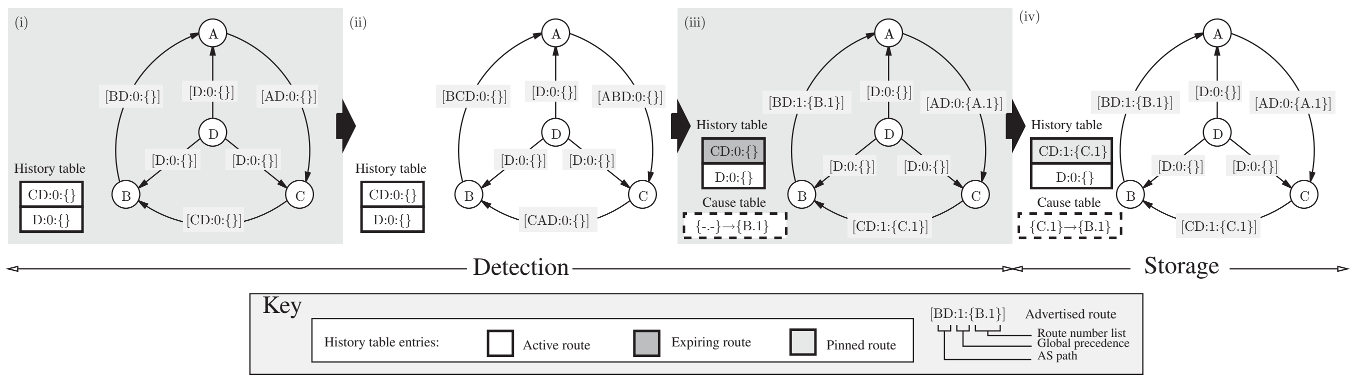


Fig. 8. Dispute wheel formation and elimination: the simple local policy enforced at each node is the import filtering of routes with more than 2 hops. The history table entries are ordered according to local preference. Since the network is symmetrical, only the history and cause tables for node *B* are shown. In phase 1, detection of disputes takes place, with -.- in (iii) indicating an increase in advertised route's precedence due to an expiring, more preferred route (as opposed to a feasible one). The route number *B*.1 represents route *BD*:1. In phase 2, incoming feasible routes cause outgoing ones to have increased values, and the cause tables are updated accordingly.

A. Detection & Short-Term Memory

In the detection phase, *short-term* memory of more preferred routes that are *infeasible*, result in less preferred but more stable routes being advertised with larger global precedence values. This mechanism determines if a possible dispute exists and operates locally, without requiring information beyond the routes received. Short term memories need only exist until it can be confirmed whether disputes resulting in permanent oscillations exist. This ensures that transient oscillations do not cause unnecessary suppression of routes. In general this amount of time is determined by the number of rim nodes between neighboring pivots, since rim nodes can be thought of as delaying route advertisements. As the number of rim nodes is difficult to obtain in practice, we can upper-bound it by using routes' path lengths. We store short term memory in a *history table*.

B. Storage & Long-Term Memory

Subsequently in the storage phase, incoming routes with larger global precedence values result in more stable ones being advertised. In this phase, short-term memories of the last unavailable routes are no longer required, but, as we shall show shortly, *long-term* memories are needed instead. Thus, the main difference between the two phases is the global precedence value of incoming routes: the first phase has incoming routes of the same value, whereas the latter has less preferred routes having smaller values. The two different types of memories together make up the required *history* mentioned earlier in §IV.

An incoming, more preferred route causing a less preferred one to be advertised with increased global precedence is stored in long term memory by being *pinned*. At the time of pinning, these incoming routes must either be *feasible*, i.e. currently being advertised by the neighbors, or their *ignore* list (containing the route demoting the current one's local rank in the presence of IRR violations) must be non-empty. Pinned routes are never evicted automatically from long term memory, and may only be considered when selecting routes if they are currently being advertised by neighbors. Also, pinned routes are unpinned only when the *causes* of their increased value have been eliminated, or a more preferred route is received and selected, or, in the case of IRR violations, a more preferred route (in the absence of the ignored route) is received. Thus, this implies that only routes associated with some number of causes can be pinned.

A *cause* of an increase in the selected route's value refers to a more preferred route that is currently advertised by a neighbor. It can be thought of as a pointer to a route, thus using it in place of the referred route reduces storage and communication overhead. We use a unique *route number* to represent a particular cause, obtaining a globally distinct number by concatenating the router's IP (say A) with a locally generated sequence number (say 24) giving us $A.24$. Rather than propagate route numbers all around the wheel, we instead maintain cause mappings in the *cause table*: when a new route is advertised with increased global precedence, we assign it a new route number, and associate with that the corresponding causes. Thus, an outgoing route number is no longer valid once all its incoming causes are removed. Long term memory consists of pinned routes and the cause mappings, which is stored in the cause table.

C. A Simple Example

We next use a simple example (Figure 8) to illustrate the resolution process. We denote a route using the 3-tuple *P*:*V*:*L*, where P refers to the AS path, *V* the global precedence value and *L* the list of route numbers. We assume strict preferences in this example and disregard the ignore list for now. We also assume that routes longer than two hops are filtered. Since the network and policies are symmetrical, we focus on a single node *B*. We begin with dispute detection; in (iii), the expiring of route *CD*:0:{} causes the less preferred route from D to be advertised with precedence value 1. Since route *CD*:0:{} is expiring and there is an absence of more preferred, feasible routes, we denote the cause by {-.-}, which we call the *null* route number. Note that the expiring route *CD*:0:{} is not pinned since its route number list is empty. In (iv), incoming route *CD*:1:{*C*.1} *causes* route *BD* to be advertised with precedence 1, and the cause table is updated accordingly. In this case, *CD*:1:{*C*.1} is pinned.

The astute reader will notice that in this example, long term memory, in the form of the cause table and pinned routes, is unnecessary to ensure convergence, since incoming routes with value 1 are always feasible. As described later in §VIII, if AS policies are based solely on next hops, the solution as described in this simple example would be sufficient. We motivate the necessity of long term memory in the next example.

D. A Complex Example

Figure 9 shows a scenario where multiple dispute wheels intersect at node *X*, where R_A denotes a route to destination with *A* as the last hop node. Again we assume strict preferences in this example. In (i), we assume that the first dispute, involving nodes *A*, *B*, *X* and *C*, has been resolved, with *X* selecting the less preferred route via *Z*. Subsequently in (ii), we have the converged network after the second dispute, involving nodes *Y*, *X* and *Z*, has been resolved. We note that:

1) Pinned routes ensure that previously encountered disputes that should still be resolved remain so. At pivot *A*, the absence of pinned route *BXR*$_Z$:1:{*X*.1} will result in the selected route *R*$_D$:0:{} being advertised with global precedence value of 0, which eventually un-resolves the first dispute.
2) Route numbers associated with selected routes are propagated unchanged. For instance, rim node *B* advertises numbers *X*.1 and *X*.2, ensuring that the next pivot (*A*) keeps route *BXR*$_Z$:1:{*X*.1} pinned.
3) The new route advertised by *X* is associated with a different route number *X*.2. In this case, since the routes of *E*.1 and *V*.1 are more preferred than *R*$_W$:0:{}, they are deemed to be causes of the latter route's increase in global precedence. Note that the cause table entries for previous route numbers, for instance *X*.1, are no longer updated.
4) At *A*, since received route BXR_W:1:{X.1,X.2} is the least preferred, *X.2* does not become a cause of *A.1*.
5) Changes in network topology is taken into account. For instance, breakage of link *XC* will, at *X*, eliminate *E*.1, which in turn removes the corresponding entry in X's cause table and thus *X*.1, and thereafter unpin (and remove expired) route *BXR*$_Z$:1:{X.1} at *A*, and so forth. This corresponds to the elimination of the first dispute wheel.

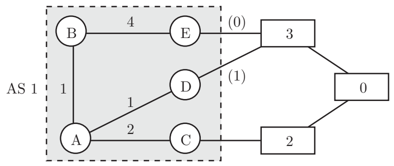


Fig. 10. The MED-EVIL example from [6].

TABLE II
MED OSCILLATION IN FIGURE 10

	A		B	
Step	Available	Advertised	Available	Advertised
1	D30, C20	AD30	E30	BE30
2	BE30, D30, C20	AC20	E30, D30	BE30
3	BE30, D30, C20	AC20	E30, AC20	BAC20
4	D30, C20	AD30	E30, AC20	BAC20
5	D30, C20	AD30	E30, D30	BE30
Repeat from step 2.				

E. A MED Example

A significant problem in BGP today is the occurrence of oscillations due to MED. MED selection rules are different from local preferences, AS path lengths etc. because they result in non-strict preferences. Figure 10 shows an example from [6]. Here, link weights in brackets denote MED values assigned to links from external ASes, whereas weights within AS 1 indicate the link's iBGP cost. Table II shows the sequence of routes advertised during an oscillation period.

We observe from Table II that the primary issue is the change in the most preferred route, from $D30$ to $C20$, with the reception of $BE30$. That is, the cause of $D30$ being demoted in rank is brought about by $BE30$. In order for the dispute detection to be effective, we create a logically different, expiring route $D30'$ that is still the most preferred (in the absence of $BE30$). $BE30$ is associated with $D30'$, the former is ignored when comparing the latter with other routes (for instance when determining whether other routes are more preferred). Subsequently, the selected route $AC20$ will be advertised with increased precedence. Pinning of the logical route $D30'$ at A (via incoming route $BE30$) and route $AC20$ at B takes place (long term memory), and the dispute is resolved. For the logical route $D30'$ to be unpinned, there must be an incoming route that is more preferred than it in the absence of $BE30$. As before, we note that no additional policies are revealed.

Denoting a stored route by the 4-tuple P:V:L:I, where I refers to the list of incoming routes to be ignored, or the *ignore list*, when computing this route's local precedence, the sequence of route updates is shown in Table III. Note that the ignore list is not sent to neighboring nodes.

F. Convergence Proof

We now show that the correct routes are pinned so that disputes are resolved and remain so. The proof focuses on individual pivot nodes.

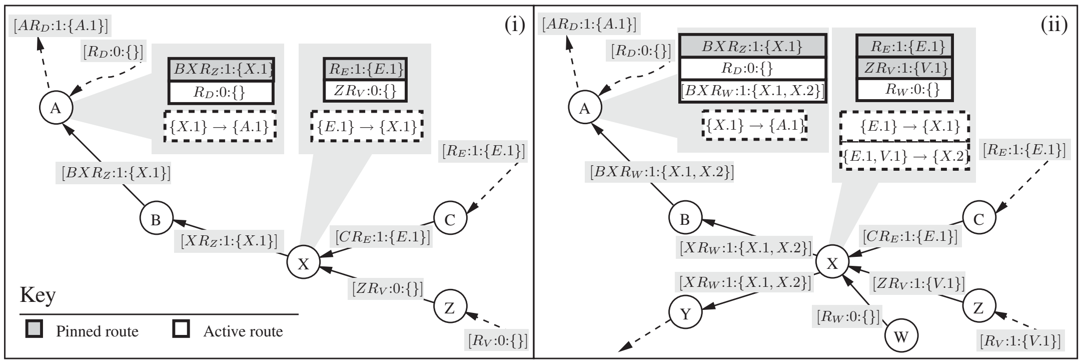


Fig. 9. R$_A$ denotes a route with A being the last hop node. (i) A simple dispute wheel is first resolved, with nodes *A* and *X* being the pivots, *B* and *C* the rim nodes, and *Z* is the last node along the least preferred route. (ii) Node X is involved in a second dispute, where pinned routes ensure that the first dispute remains resolved.

TABLE III
MED OSCILLATION ELIMINATION

Step	A	B

1	Available
	D30:0:{}:{}
	C20:0:{}:{}
	Advertised
	AD30:0:{}
		Available
		E30:0:{}:{}
		Advertised
		BE30:0:{}

2	Available
	D30':0:{}:{BE30}
	BE30:0:{}:{}
	C20:0:{}:{}
	D30:0:{}:{}
	Advertised
	AC20:1:{A.1}
		Available
		AD30:0:{}:{}
		E30:0:{}:{}
		Advertised
		BE30:0:{}

3	Available
	D30':0:{}:{BE30}
	BE30:0:{}:{}
	C20:0:{}:{}
	D30:0:{}:{}
	Advertised
	AC20:1:{A.1}
		Available
		AD30:0:{}:{}
		E30:0:{}:{}
		AC20:1:{A.1}:{} - pinned
		Advertised
		BE30:1:{B.1}

4	Available
	D30':0:{}:{BE30} - pinned
	BE30:1:{B.1}:{}
	C20:0:{}:{}
	D30:0:{}:{}
	Advertised
	AC20:1:{A.1}
		Available
		E30:0:{}:{}
		AC20:1:{A.1}:{} - pinned
		Advertised
		BE30:1:{B.1}

Proposition 5.1: With long term memory, the network eventually converges.

Proof: Let the new available route be R_n, the previous selected route be R_w, and the set of non-empty routes (including pinned routes) more preferred than R_w be $\mathcal{R}_p$.

Suppose R_n is not selected. This implies that R_n's global precedence is at least as low as R_w's. If R_n is less preferred than R_w, then there is no further effect. Otherwise R_n becomes one of the causes of the increase in global of R_w. In either case, no oscillations are introduced as the advertised route's AS path and global precedence remains unchanged.

On the other hand suppose R_n is selected and advertised.
Case 1: If R_n is more preferred than R_w, then its precedence is at most as high as R_w's, and we unpin (and remove expired) pinned routes less preferred than R_n, as well as remove their corresponding outgoing route numbers. Previous dispute-induced oscillations associated with the unpinned routes cannot reappear due to the presence of R_n, that is, we will not again encounter exactly the same routes advertised during previous oscillations. Since more preferred, pinned routes are not unpinned, and the maximum number of these routes is bounded, this scenario cannot occur indefinitely.

Case 2: If R_n is less preferred than R_w, then its precedence value must be strictly lower than R_w's. Since all pinned routes less preferred than R_w must have been unpinned before, none less preferred than R_n will exist. In other words, disputes previously resolved remain so.

■

G. Achievement of Goals

Based on the Precedence Solution proposed earlier, we next describe how our goals are met.

Handling transient and permanent oscillations: An oscillation that is eliminated is said to be permanent if the route numbers associated with outgoing, increased precedence value routes at each pivot are always advertised. In the case of transient oscillations, the expiration of more preferred routes eliminates the corresponding outgoing route numbers, thereby unpinning upstream routes. Alternatively, more preferred routes can be encountered later, in which case those less preferred and pinned will be unpinned and also removed if expired.

If instead network topology changes, such as link breakage, occur, incoming route numbers via that link is removed, the cause table is updated accordingly and its effect is subsequently propagated around the wheel. Similarly, the cause and history tables can be flushed whenever policy changes occur. Thus, the Precedence Solution allows for both transient oscillations and changes in network topology, and does not permanently suppress any routes.

Minimal revealing of policies: If the input routes of a router are known, the global precedence of the advertised route indicates whether the chosen route is the most preferred: if it is not, then its value increases. This is not much different from today's network: given the inputs, a route is not the most preferred if it is not advertised.

For routes stored in the history table, all have been previously advertised before and have been intended to be used for routing, none have been explicitly propagated for purposes of eliminating oscillations. Route numbers have been explicitly associated with these routes, and do not contain any additional information. Thus, we do not expose any additional AS policies.

No requirement for global knowledge: Both the detection and storage mechanisms operate solely on route advertisements received from neighbors, and are fully decentralized. No third party is required to gather and compute optimal routes for all ASes. The route numbers are propagated upstream only for the purposes of ensuring that disputes remain resolved, and do not affect nodes elsewhere, including other parts of the wheel.

Identification of potential pivot nodes: Although it is possible to use some other unique number as the route number, we believe that inclusion of the router IP gives the right amount of visibility to assist in network troubleshooting. If a node is forced to select a less preferred route, the cause table maps incoming route numbers to an outgoing one, appended to those already associated with the selected route. Otherwise the numbers associated with a selected route is propagated unchanged. Thus, the set of nodes identified by the list of numbers includes all potential pivots encountered downstream. Although not all pivots along the wheel can be identified from a single viewpoint, adjustment of just one such node's preferences is sufficient to break the dispute, reducing global precedence values and relaxing constraints on route selection.

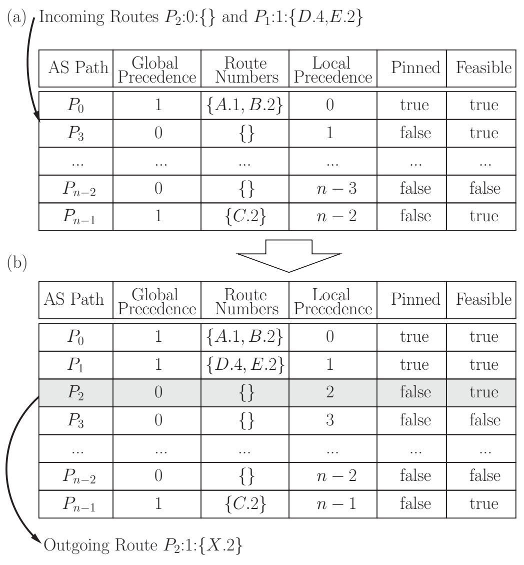
VI. Router Changes

In this section we describe extensions to a BGP router necessary to implement the Precedence Solution. The two main additions are the history and cause tables.

A. History Table

The history table stores routes received from neighbors, as well as information relevant to short and long term memories necessary for dispute detection and storage. Memory used to store routes may be shared amongst the different data structures, and is dependent on actual implementation. Thus, the history table can be thought of as an extension to other structures.

A route that is currently being advertised is *feasible*. Infeasible routes that are not pinned are removed, and only feasible routes can be considered for selection and pinning. A route is unpinned if there exists a selected route that is more preferred (or if no route is selected). Pinning of a route occurs when it has a non-empty route number list and it causes a less preferred route to be advertised with increased global precedence value.


(a) Incoming Routes P_2:0:{} and P_1:1:{$D.4$,$E.2$}
(b)
Outgoing Route P_2:1:{$X.2$}

(a)

AS Path	Global Precedence	Route Numbers	Local Precedence	Pinned	Feasible
P_0	1	$\{A.1, B.2\}$	0	true	true
P_3	0	{}	1	false	true
...	...	...	...	...	...
P_{n-2}	0	{}	$n-3$	false	false
P_{n-1}	1	$\{C.2\}$	$n-2$	false	true

(b)

AS Path	Global Precedence	Route Numbers	Local Precedence	Pinned	Feasible
P_0	1	$\{A.1, B.2\}$	0	true	true
P_1	1	$\{D.4, E.2\}$	1	true	true
P_2	0	{}	2	false	true
P_3	0	{}	3	false	false
...	...	...	...	...	...
P_{n-2}	0	{}	$n-2$	false	false
P_{n-1}	1	$\{C.2\}$	$n-1$	false	true

Fig. 11. (a) Before and (b) after a history table is updated.

Figure 11 shows an example of a history table being updated, and Figure 12 provides the pseudo-code. Entries in the table are arranged in order of local precedence, that is, the ordering is determined using the same rules as the decision process in use today. This ordering provides the local precedence value: the most locally preferred has value 0, the next 1, and so forth.

B. Cause Table

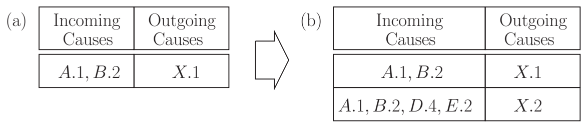
The cause table contains entries that map incoming causes' route numbers to outgoing ones. To recapitulate, the route number of a cause consists of a router interface's IP address and a locally unique sequence number. Figure 13 shows an example of a cause table being updated corresponding to the update in Figure 11, with Figure 14 providing the pseudo-code. An entry exists until all its incoming causes cease to be received. In the figure, the cause entry for $X.1$ will be evicted if both $A.1$ and $B.2$ are no longer advertised by the neighbors. A new route number $X.2$ is created with a change in the selected route. Only current causes, that is, those using the most recent sequence numbers, are updated based on received routes. Thus in Figure 13 new incoming causes will be added to the entry for $X.2$, but not for $X.1$.

C. Adaptive Convergence Window

As elaborated in §V, we require the use of short term memory to detect disputes. The convergence window is the period of time during which received routes are kept in memory. Assuming that one-hop route propagation delay W is similar to the Minimum Route Advertisement Interval (MRAI), the rim nodes (say there are r of them) can be thought of as delaying route advertisements from one pivot to another by rW, thus the window size should be proportional to this number.

```
if local routing has converged (no routing changes) then
    for each entry in history table do
        if not feasible and not pinned then
            remove entry
for each route $R$ received do
    if route from neighbor $N$ is filtered then
        set previous feasible route $R_p$ from $N$ infeasible
    else if different route received from neighbor $N$ then
        set previous route $R_p$ from $N$ infeasible
        compute $R$'s local precedence
        insert $R$ into history table, set feasible
    else if same route received from neighbor then
        if previous feasible route $R_p \neq R$ then
            set $R_p$ infeasible
        else
            update $R$'s global precedence value and route numbers
        set $R$ feasible
    else if first route $R$ is received from neighbor then
        compute $R$'s local precedence
        insert $R$ into history table, set feasible
select set $\mathcal{S}$ of eligible routes
select set $\mathcal{S}'$ with lowest global precedence, $\mathcal{S}' \subset \mathcal{S}$
select route $R$ with lowest local precedence, $R \in \mathcal{S}'$
for each entry in history table do
    if route $R_{mp}$ more preferred than $R$, feasible, has non-null route
    number list then
        pin $R_{mp}$
    else if route $R_{lp}$ less preferred than $R$ and pinned then
        unpin $R_{lp}$
return $R$
```

Fig. 12. Pseudo-code for updating history table and determination of the selected route for each destination.



(a)

Incoming Causes	Outgoing Causes
$A.1, B.2$	$X.1$

(b)

Incoming Causes	Outgoing Causes
$A.1, B.2$	$X.1$
$A.1, B.2, D.4, E.2$	$X.2$

Fig. 13. Updating of cause table corresponding to history table update in Figure 11. A change in selected route triggers the generation of a new sequence number, and only most recent outgoing causes are updated.

```
create empty route number set $\mathcal{S}_r$
let selected route be $R_s$
for each entry in history table do
    if route $R$ more preferred than $R_s$ and feasible then
        pin $R$
    else if $R_s$ more preferred than $R$ then
        unpin $R$
    if route $R$ is pinned then
        add $R$'s route numbers to $\mathcal{S}_r$
for each entry in history table do
    if route $R$ is pinned and none of $R$'s route numbers $\in \mathcal{S}_r$ then
        unpin $R$
if $R_s$ is different from previous then
    create new route number
if $\mathcal{S}_r \equiv \emptyset$ and $R_s$ is not most preferred and feasible then
    add null route number to $\mathcal{S}_r$
set current outgoing cause's incoming route numbers to $\mathcal{S}_r$
for each entry in cause table do
    if no incoming route numbers are present in $\mathcal{S}_r$ then
        remove cause entry
```

Fig. 14. Pseudo-code for updating cause table for each destination prefix.

```
for each adj-RIB-in do
    process incoming routes, update route table
update history table
update cause table
for each adj-RIB-out do
    update new routes' route number list
    advertise route to peer
```

Fig. 15. Primary steps in router batch updates.

The convergence window begins with a short duration (one MRAI), so that networks not containing disputes can converge relatively quickly. We double its duration when convergence does not occur after some time, so that disputes involving large numbers of rim nodes can be resolved quickly. An upper bound can be determined using the maximum observed path length during this period. Lastly, its duration is reset after the network stabilizes to remove effects of transient convergence.

VII. Evaluation

A. Simulator

We built an event-based, packet-level and asynchronous simulator. Route updates are *batched*, and take place every Minimum Route Advertisement Interval (MRAI). Figure 15 shows the main steps of the batch update process, whereas Figures 12 and 14 describe maintenance of the history and cause tables respectively. We set MRAI to 30 seconds, processing delay jitter to 1 second, and link propagation delay to 10 milliseconds.

B. Metholodgy

To better understand the basic performance of our solution, we use simple graphs, which consist only of rim, pivot and destination nodes. Figure 8 shows an example. Whilst these graphs are not representative of a real network in general, they are still useful in determining properties of a dispute wheel.

To evaluate the effectiveness of the Precedence Solution in practice, we use an AS-level network topology constructed using routing table dumps from RouteViews [13]. Route dumps from January 3rd 2007 were used to construct an AS-level network which consists of 24307 ASes and 56914 inter-AS links. Since complete policy information is impossible to obtain [3], [15], we sought an alternative method of generating local preferences. Restricting ourselves to next hop preferences, we note that a dispute-free configuration can be obtained as long as the most preferred neighbor lies along a cycle-free path to the destination. Thus, a shortest-path algorithm will generate local preferences that can guarantee convergence.

However, inter-domain routing typically does not result in shortest paths [16], and as we show later, the network convergence time as well as the degree of route exploration (and hence the number of routes encountered) are dependent on the ratio of actual versus shortest path lengths (*i.e.* route inflation). Thus, we focus on routing algorithms that provide approximately the same route inflation. We use a

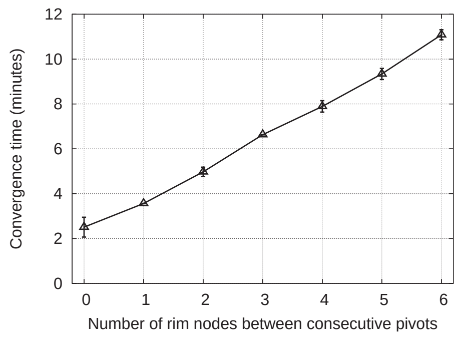


Fig. 16. Simple graphs: 3-pivot network's convergence time against rim-to-pivot ratio.

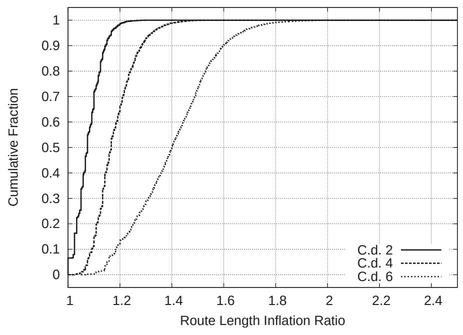


Fig. 17. An increase in the maximum depth of constrained depth-first routing results in more inflated routes. For constrained depth (c.d.) of 6, we obtain paths with inflation close to that in practice [16].

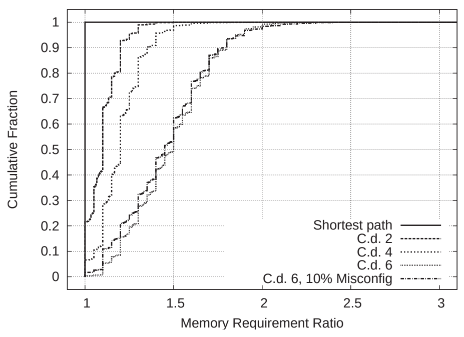


Fig. 18. As path lengths deviate away from shortest, the exploration of more paths before convergence results in more routes being stored for Precedence+.

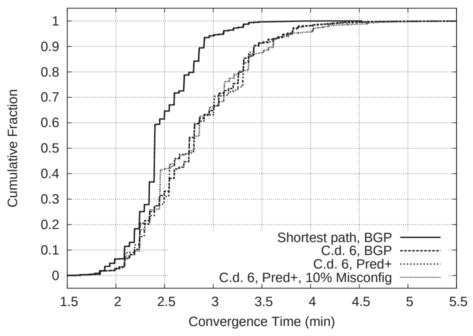


Fig. 19. Inflated paths result in an increase in convergence times. Precedence+ does not delay convergence, even in the presence of misconfigurations.

combination of depth-limited and breadth-first searches to obtain routing trees: depth-limited search is used whilst within the limit at each stage, otherwise BFS is used. In general, increasing the maximum depth at each stage results in greater route path inflation. The remaining neighbors' preferences are set in a random fashion. Finally, we simulated misconfigurations by selecting a subset of routers and randomly assigning local preferences.

C. Metrics

We use convergence time and memory requirement as metrics. We say that a node has converged at a certain time if its routing table no longer changes thereafter. As for memory requirements, we look at the ratio of routes stored when using our solution against normal BGP. This allows comparison across the entire network, taking into account routers with varying numbers of neighbors.

D. Results

Simple graphs: Using simple graphs, we determined that the convergence time is dependent on the rim-to-pivot ratio and not the total size of the network. We show representative results in Figure 16, where the number of pivot nodes is 3. Each data point in the figure is obtained from 20 samples; we see that the mean convergence time increases with this ratio, and there is little deviation in all cases. In all experiments the networks converged.

RouteView graph: We varied the maximum depth of each constrained depth-first iteration, obtaining the mean route length inflation ratios shown in Figure 17. A maximum depth of 6 results in route inflation that most closely match that in the Internet today [16].

Next, we investigated the impact of additional memory requirements for Precedence+ by varying route inflation. In all cases, we verified that usage of Precedence+ in networks with no disputes resulted in all nodes selecting their most preferred next hops: Precedence+ does not unnecessarily suppress routes. For normal BGP, the amount of memory required at a router is proportional to the number of its neighbors. From Figure 18, we observed that deviation from the shortest path results in more routes being explored and hence more being stored before convergence in the case of Precedence+. On average, Precedence+ requires 50% more memory for each destination prefix, which can be amortized across the network by jittering initial prefix advertisements. Furthermore, actual route exploration in the Internet may be to a lesser extent since route advertisement will be constrained by economic policies.

To investigate policy disputes, we randomly assigned next hop preferences to 10% of the nodes. We verified that dispute wheels do exist (normal BGP does not converge), and that the networks converged when Precedence+ is used. As shown in Figure 18, we required approximately the same

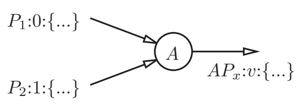


Fig. 20. Basic scenario used to describe misbehavior: node A receives two routes and advertises one. Detection of misbehavior can be performed by observing incoming and outgoing routes.

amount of memory as before.

Finally, we looked at the network convergence times (Figure 19). As we expected, local preferences assigned based on shortest-paths results in faster convergence. More importantly, convergence time is not significantly affected by usage of Precedence+, nor by the presence of misconfigurations (disputes) in the network.

VIII. Discussion

In this section we discuss two issues encountered in practice, namely constrained policies and misbehavior.

A. Constrained Policies

If the local precedence value of a route is determined first by the last hop, that is, if the first step of the BGP decision process selects routes based on next-hop ASes, then the Precedence Solution can be significantly simplified. In Figure 9(ii) at node A, if neighbor B's routes are more preferred than D's, then route BXR_W:1:{X.1,X.2} will be ranked higher than R_D:0:{}, thus the eviction of BXR_Z:1:{X.1} will not un-resolve the first dispute. In other words, pinning of routes and long-term storage, including cause tables and route numbers, are unnecessary. Communication overhead is reduced as well, since only the global precedence value need to be carried with each route advertisement.

B. Misbehavior

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 ASes can misbehave, and detection methods that rely on the ability to observe the incoming and outgoing routes. 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 20, focusing on the routes advertised from A.

P$_2$:0:{...}: there is definite misconduct, since the outgoing route's global 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 global precedence is not incremented.

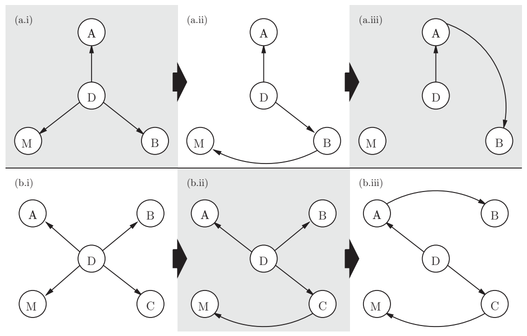


Fig. 21. A misbehaving AS, represented by node M, can have differing effects on the network. (a) For a dispute wheel with an odd number of nodes, M eventually lacks a route if it initially filters the spoke one. (b) For a wheel with an even number of nodes, M does not destabilize the network.

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.

C. 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 rim. 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 rim.

In this case, two scenarios can occur as illustrated in Figure 21. 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

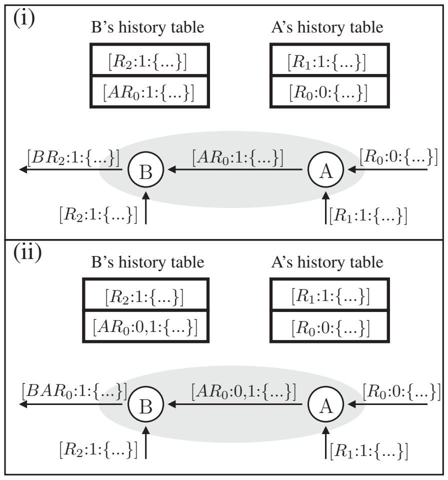


Fig. 22. (i) With multiple routers within an AS, indicated by the shaded region, external input and output routers can appear to indicate misbehavior even if they are operating according to protocol. (ii) By tagging the route with both ingress and egress precedence values, an AS' behavior becomes similar to that of a single router.

implementation of adaptive filtering in M can result in oscillations (which ultimately does not benefit M). To provide better control of the situation, we next propose a method to detect the various types of misbehaviors discussed above.

D. Misbehavior Detection

Most ASes are comprised of multiple routers, and are unlikely to provide access to the internal network. Thus, the usual assumption that an AS can be modeled by a single router does not hold. For instance, in Figure 22(i), router A selects, as it should, the less preferred route R_0:0:{...} and advertises AR_0:1:{...} to B. B subsequently chooses R_2:1:{...}. If we logically collapse A and B into a single node and aggregate their inputs, we see that even though the routers are behaving correctly, the output should have been ...R_0:1:{...} instead.

We propose a slight tweak to the protocol only within an AS: when an ingress router (i.e. A in the example) advertises a route to an internal peer, it appends the route's global precedence when received (the *ingress* value) and after updates (the *egress* value). Upon reception of that route, B uses the ingress value to determine the selected route. The egress value is then updated, and is lower-bounded by the previous egress value. Advertisements to neighboring ASes carry only the egress value.

Figure 22(ii) shows the same network with the tweaked protocol. Here, correct behavior will cause A to advertise AR_0:0,1:{...}, and B to advertise BAR_0:1:{...}. On the other hand, if A misbehaves and selects R_1:1:{...}, the output will clearly be incorrect.

With the slightly modified protocol, the conditions described in §VIII-B can be used to detect the occurrence of adaptive filtering. For a dispute to occur, a less preferred route (say R_{lp}) must have been advertised before the more preferred one is selected. Thus, R_{lp} must have been observed before, but not thereafter. A monitoring mechanism can be designed based on this as follows: we detect routes that should have been selected but aren't. These are then hashed and stored. Since the monitor is maintained by a third-party, hashing of the inputs provide anonymity. Output of any of the stored routes in the future signals reuse of those routes, and therefore adaptive filtering.

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.

References

[1] J. A. Cobb, M. G. Gouda, and R. Musunuri. A Stabilizing Solution to the Stable Paths Problem. In *Symposium on Self-Stabilizing Systems, Springer-Verlag LNCS*, pages 169–183. ACM Press, 2003.

[2] 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.

[3] L. Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745, 2001.

[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 Transactions on Networking*, 9(6):681–692, 2001.

[6] T. Griffin and G. T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 90–99, Washington, DC, USA, 2002. IEEE Computer Society.

[7] 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.

[8] 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.

[9] T. G. Griffin and G. Wilfong. A Safe Path Vector Protocol. In *Proceedings of IEEE INFOCOM 2000*. IEEE Communications Society, IEEE Press, March 2000.

[10] 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.

[11] A. D. Jaggard and V. Ramachandran. Robust Path-Vector Routing Despite Inconsistent Route Preferences. In *Proceedings of ICNP'06*. IEEE Computer Society, IEEE Press, November 2006.

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

[13] *University of Oregon RouteViews Project*. http://www.routeviews.org.

[14] J. L. Sobrinho. An Algebraic Theory of Dynamic Network Routing. *ACM/IEEE Transactions on Networking*, 13(5):1160–1173, October 2005.

[15] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proc. of IEEE INFOCOM 2002, New York, NY*, Jun 2002.

[16] H. Tangmunarunkit, R. Govindan, S. Shenker, and D. Estrin. The impact of routing policy on internet paths. In *Proc. of IEEE INFOCOM 2001, Anchorage, AK*, Apr 2001.

[17] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-domain Routing. *Computer Networks*, 32(1):1–16, March 2000.