Resolving Inter-Domain Policy Disputes

Cheng Tien Ee, Byung-Gon Chun
Department of Computer Science
University of California, Berkeley
Berkeley, CA 94720, USA
{ct-ee, bgchun}@cs.berkeley.edu

Vijay Ramachandran
Department of Computer Science
Colgate University
Hamilton, NY 13346
vijayr@cs.colgate.edu

Kaushik Lakshminarayanan
Department of Computer Science and Engineering
Indian Institute of Technology Madras
Chennai 600036, India
klkaushik@gmail.com

Scott Shenker
International Computer Science Institute (ICSI) &
University of California, Berkeley
Berkeley, CA 94704
shenker@icsi.berkeley.edu

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.

Categories and Subject Descriptors

C.2.6 [**Communication Networks**]: Internetworking

General Terms

Algorithms, Design, Theory

Keywords

Inter-domain routing, BGP, convergence, policy disputes

1. 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 §2 and then define and discuss dispute wheels in §3. The precedence metric is de-

SIGCOMM'07, August 27–31, 2007, Kyoto, Japan.

scribed next and its ability to prevent dispute wheels proven in §4. §5 and 6 describe how this theoretical result can be put into practice. We evaluate the resulting algorithm in §7 and discuss several issues in §8 before concluding in §9.

2. RELATED WORK

Varadhan et.al. [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 et.al. [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 §4). 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 et.al. 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 et.al. [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 et.al. [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. Finally, our protocol eases network troubleshooting by pinpointing the routers at which policy conflicts occur, carrying the information together with routes propagated upstream.

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.

3. 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

[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.

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 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 et.al. [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 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 $p_i R_{i+1} p_{i+1} Q_{i+1} d$ *over the path* $p_i Q_i d$*.*

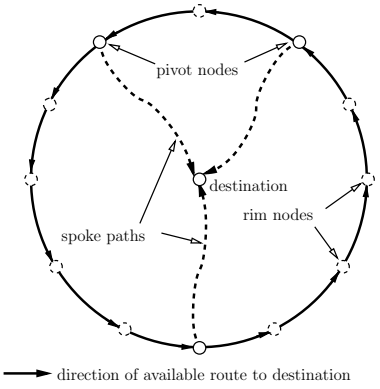


Figure 1: A dispute wheel example: elements of the wheel include spoke paths, and pivot and rim nodes.

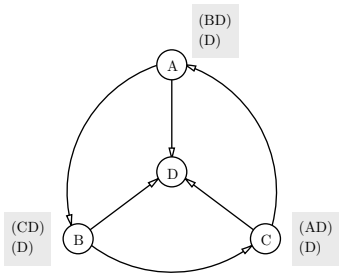


Figure 2: A simple dispute wheel: node D is the destination. Shaded boxes show route choices in order of preference.

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.Rather than exclude all potentially troublesome policy

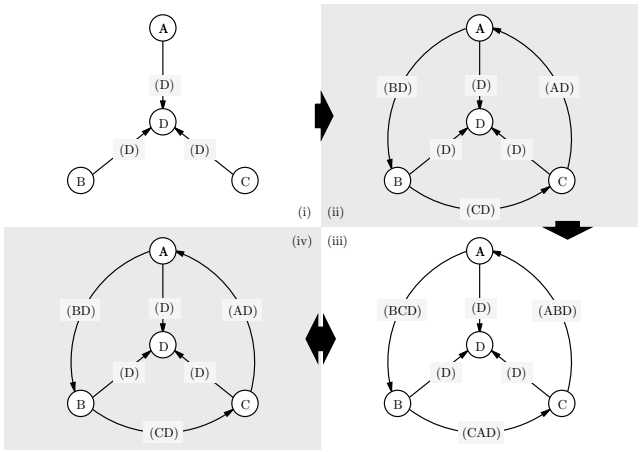


Figure 3: Simple example of dispute wheel oscillation: The simple local policy enforced at each node is the import filtering of routes with more than 2 hops.

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.

4. 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*[2] *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. In short, the selected route is first determined using the incoming global precedence value (since this step occurs before the current BGP decision process), followed by its local precedence value.

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 a global value of 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.

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 5 to explain this.

[2]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.

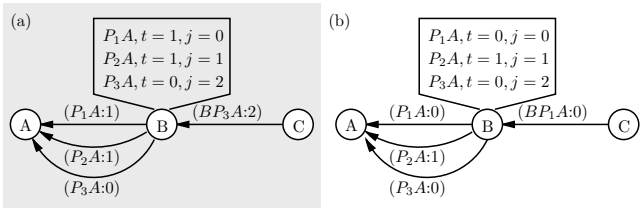


Figure 4: (a) AS B's local preference for route (P_3A:0) to destination AS A is ranked third. Advertisement of this route to AS C will result in the increasing of its global precedence by 2. (b) AS B now considers the route (P_1A:0) to AS A to be the most locally preferred. Advertisement of this route to AS C will not alter its global precedence value.

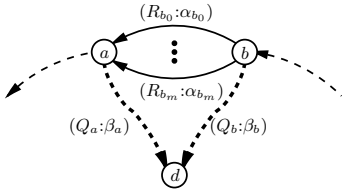


Figure 5: Multiple paths advertised by neighboring nodes can cause the global precedence value of a route to increase by more than 1.

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 $R_{b_0}, R_{b_1}, \ldots, R_{b_m}$ compared to Q_b, we have the history state shown in Table 1. 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.

Precedence values can take on multiple non-negative values as opposed to just binary 0 or 1 values. The presence of a dispute wheel causes routes beyond the nodes in and within the wheel to be advertised with the same incremented value. Nodes outside the wheel can still be in dispute, in which case the global precedence will be incremented again. We next show that this precedence metric prevents the formation of dispute wheels.

4.1 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 6 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.

Table 1: History of Node b in Figure 5

Route	Global Precedence	Local Precedence
$R_{b_0}aQ_ad$	$\alpha_{b_0}+\beta_a$	0
$R_{b_1}aQ_ad$	$\alpha_{b_1}+\beta_a$	1
$\cdots$	$\cdots$	$\cdots$
$R_{b_m}aQ_ad$	$\alpha_{b_m}+\beta_a$	m
Q_bd	β_b	$m+1$

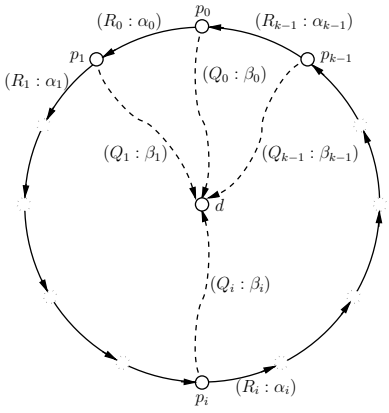


Figure 6: Dispute wheel illustration and notation used in our proof.

$(Q_i{:}\beta_i)$ 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 $R_ip_{i+1}Q_{i+1}d$, then $\gamma{=}\alpha_i{+}\beta_{i+1}$. We also have r_i be the total number of nodes along R_i, including p_{i+1} and excluding p_i. This implies

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

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 $(R_ip_{i+1}Q_{i+1}d{:}\alpha_i{+}\beta_{i+1})$, over route $(Q_id{:}\beta_i)$. In Figure 6, 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.[3] 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

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

Summing, we obtain

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

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 $R_ip_{i+1}Q_{i+1}$ is chosen over Q_i, it must be because of the global precedence values.

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

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

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. □

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. □

4.2 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

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

are non-strict, and are encountered for instance in BGP's Multi-Exit Discriminators (MEDs) [6]. In this subsection we propose a minor extension to account for this.

Non-strict preferences results in an incoming route R_i causing 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 eviction of R_{ps} from the most locally preferred rank. Thus, to capture the fact that R_{ps} used to be the most preferred before R_i arrived, we create and associate a logical route R'_{ps} with R_i, where $R'_{ps} = R_{ps}$ except that comparison of R'_{ps} with any other route[4] should ignore the presence of R_i. This slight tweak is used when computing the local preference of R_{cs}. 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. Furthermore, as $R_i \neq R_{cs}$, we will not encounter the scenario when R_i and R'_{ps} are compared. In the case where R_{cs} becomes unavailable in the future and is replaced by R_i, we evict R'_{ps}.

5. FROM THEORY TO PRACTICE

In §4, 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, and no global knowledge, is necessary.

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 probably unattainable, we seek an alternative means of identifying the potential pivots.

Our solution consists of a detection and a stable phase (Figure 7). We give a brief description of the two phases below, with elaborations in the subsequent subsections.

Detection Phase: Initially, a node locally determines that it *may* be a pivot and be involved in a dispute when its current selected and advertised route is less preferred than its previous one. By keeping in memory such withdrawn (also known as *infeasible*) routes, the precedence value of the advertised route is incremented. The timeout period for infeasible routes thus determines the duration of the detection phase, and is elaborated on in the following section.

Stable Phase: In this phase, incoming and more preferred but unstable routes have incremented precedence values, thus resulting in the pivot node thereafter always selecting a stable but less preferable route. On the other hand, if there is no such incoming route, then expiration of infeasible routes will cease to cause the advertised route's precedence to be incremented.

[4]when determining the local precedence value using the current BGP decision process

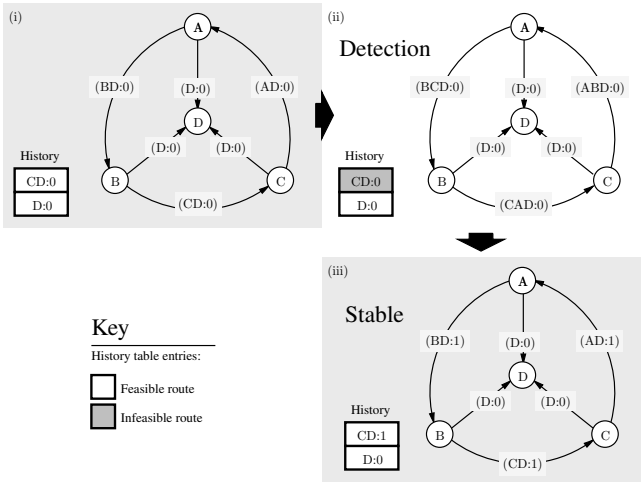


Figure 7: (i) Dispute wheel formation and elimination: simple local policy enforced at each node filters incoming routes with more than 2 hops. (ii) Detection phase: existence of an expiring, more preferred route causes the outgoing precedence value to increase. (iii) Stable phase: incoming more preferred routes have incremented precedence values, causing the stable, less preferred routes to be selected.

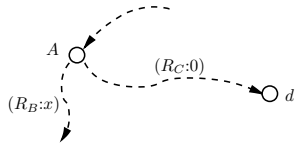


Figure 8: To ensure that the wheel size is measured, pivot node A updates and propagates the router counter associated with the more preferred route $\mathbf{R}_B$ that is received from a neighboring node along the wheel rather than that with the stable route $\mathbf{R}_C$.

5.1 Detection Phase

In the detection phase, infeasible routes are stored temporarily, resulting in less preferred but more stable routes being advertised with incremented precedence values. This mechanism determines if a possible dispute exists and operates locally, without requiring additional information beyond the routes received. These memories need only exist until it can be confirmed whether disputes resulting in permanent oscillations are present, ensuring that transient oscillations do not cause unnecessary suppression of routes.

The maximum period of time required for infeasible routes to be stored is that needed for the effect of a route change to propagate around the dispute wheel. This is in general proportional to the number of nodes N_d involved in the dispute, and can be obtained by multiplying N_d by the Minimum Route Advertisement Interval (MRAI). We obtain an

upper bound to this number by propagating a *router counter* that is initiated by pivots and incremented by each traversed node. Pivots begin sending these counters if routing has not converged and routes with incremented global precedence values are advertised.

An important point to note is that the router counter should not be thought of as being associated with a particular route, but rather with all routes with the same destination prefix. With reference to Figure 8, this means that pivot A uses, updates and propagates the router counter carried with the more preferred route R_B, even though it selects the stable route R_C, since (i) the latter could have a precedence value lower than that of the former, or (ii) the former is an infeasible route, or (iii) the former has been filtered due to implemented policies (and not because of loop detection). In the case where multiple, more preferred routes exist, pivot A uses the largest counter amongst these routes, since the goal is not to determine the precise number of nodes in the wheel, but to provide an upper bound. Rim nodes simply increment the counter associated with their selected routes.

The router counter is used to determine the period of time over which the network is deemed to have converged if the selected route is unchanged. Once this condition is true, the node enters the stable phase. Finally, we again highlight the fact that the counter is used primarily as an indication of the period of time infeasible routes should remain in memory in order for the network to converge. If convergence occurs before the total number of nodes in dispute is recorded, then propagation of the counter stops.

5.2 Stable Phase

In this phase, all infeasible routes would either have expired and evicted from memory, or have been readvertised (thereby become feasible again). The reception of a more preferred route having incremented precedence value causes pivot nodes to select stable but less preferred routes. Also, we reset the variable largest router counter encountered during the detection phase.

Usage of the precedence metric has the additional benefit that nodes are aware that they are experiencing policy conflicts by observing that their selected and advertised routes have incremented precedence values. We propose propagating this knowledge upstream in the form of *route numbers*, which are a combination of router, AS and locally generated sequence number. Route numbers are carried together with the advertised routes for the purpose of troubleshooting.

5.3 A Simple Example

We next use the previous simple example (Figure 7) to illustrate the resolution process. Again, since the network and policies are symmetrical, we focus on a single node B. We begin with dispute detection; in (ii), the presence of infeasible (and expiring) route (CD:0) causes the less preferred route (D:0) to be advertised with precedence value 1. In this phase, B also begins to propagate the router counter (C_B), with an initial value of 1 (since it is part of the dispute). When this counter is propagated to A, A selects the largest counter it received (C_B), increments and advertises that. Also, pivots use the largest counters encountered to determine the timeout period for infeasible routes.

In this simple example, we note that the router counter does not get propagated beyond the immediate neighboring pivot. In the detection phase, node B advertises (BD:1), which does not need to be readvertised in the next iteration since B receives (CD:1) thereafter. As the counter is propagated together with route advertisements, this implies that no further updates to it will take place in the stable phase.

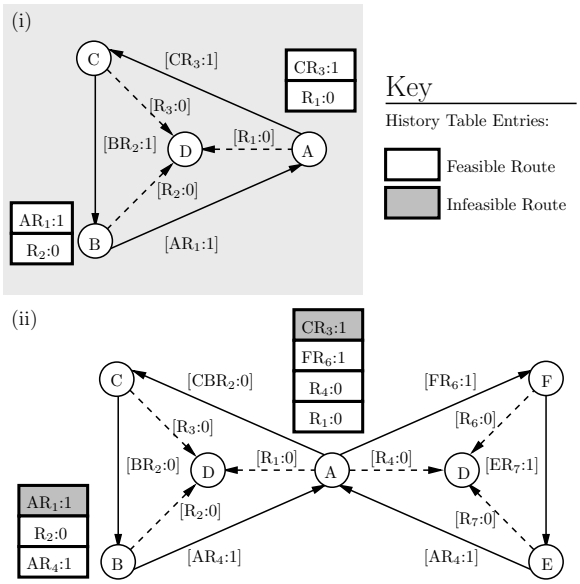


Figure 9: R_x denotes a route X terminating at the destination node D. As before, routes longer than two hops are filtered. (i) A simple dispute wheel is first resolved. (ii) Node A is involved in a second dispute. Depending on the local preference ordering at B, the first dispute may be eliminated.

5.4 A Complex Example

In Figure 9(i), we assume that the first dispute, involving nodes A, B and C, has been resolved, with the pivot nodes selecting their stable routes. A second dispute is introduced in (ii), where the disputes intersect at node A, and R_x denotes the route X to destination D. We represent destination D using separate nodes to better show the two disputes. The network is shown in the converged state, after the second dispute, involving nodes A, E and F, has been resolved. The takeaways from this example are the following:

One: Depending on the network state, the router counter sent from A can either traverse the two disputes simultaneously, or one after another. In either case, it will increase to a sufficiently high value, bounding the maximum period required to observe all routes involved in the oscillations.

Two: Depending on the local preference ordering of the nodes, the initial dispute may or may not continue to exist. In Figure 9(ii), the ordering at node B is such that AR_1 is more preferred than R_2, which in turn is more preferred than AR_4, thus the dispute involving A, B and C is eliminated. As a result, at steady state (after the expiration of AR_1) nodes B and C are able to select their most preferred routes.

5.5 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

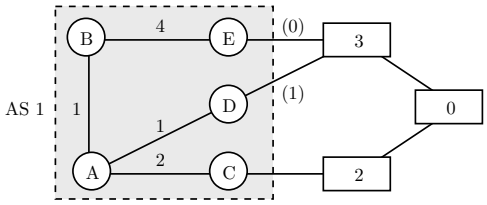


Figure 10: The MED-EVIL example from [6].

Table 2: MED Oscillation in Figure 10

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

AS 1 indicate the link's iBGP cost. Table 2 shows the sequence of routes advertised during an oscillation period.

We observe from Table 2 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. As before, we note that no additional policies are revealed.

Denoting a stored route by the 3-tuple $P{:}V{:}\{I\}$, 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 3. Note that the ignore list is not sent to neighboring nodes.

5.6 Achievement of Goals

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

1. Handling transient and permanent oscillations: We say that oscillations are permanent if they are caused by disputes that do not resolve by themselves. On the other hand, transient oscillations refer to those that disappear without usage of our solution, Figure 11 gives an example of this. Node A first learns of the route R_1, and propagates AR_1 to B. Next, A receives the advertisement for R_2, which is preferable to R_1, and subsequently advertises that to B after withdrawing AR_1. However, at B, since AR_1 is more preferred, it remains in memory before timing out and evicted. The eventual eviction results in B advertising AR_2 with precedence value 0.

Table 3: MED Oscillation Elimination

	A		B	
Step	Available	Advertised	Available	Advertised
1	D30:0:{} C20:0:{}	AD30:0	E30:0:{}	BE30:0
2	*D30':0:{BE30}* BE30:0:{} C20:0:{} D30:0:{}	AC20:1	AD30:0:{} E30:0:{}	BE30:0
3	*D30':0:{BE30}* BE30:0:{} C20:0:{} D30:0:{}	AC20:1	*AD30:0:*{} E30:0:{} AC20:1:{}	BE30:1
4	*D30':0:{BE30}* BE30:1:{} C20:0:{} D30:0:{}	AC20:1	E30:0:{} AC20:1:{}	BE30:1

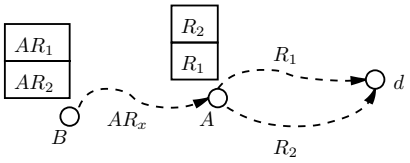


Figure 11: Routes preferable for one node may not be for upstream nodes. R_x refers to either R_1 or R_2.

Transient oscillations can also refer to those that are not due to the resolution of disputes by our solution. In Figure 9, the dispute on the right eliminates that on the left. The expiration of route AR_1 at B allows B to advertise R_2 without incrementing its precedence value. Thus, our solution does not unnecessarily penalize nodes.

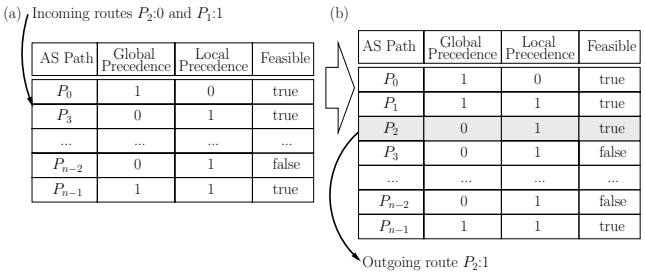
If instead network topology changes, such as link breakage, occur, adjacent nodes' states are reset. The resulting elimination of disputes (if any) is manifested in the withdrawal of more preferred routes with larger precedence values. These routes eventually expire and allow the remaining pivots to select their most preferred routes. Similarly, state can be reset whenever policy changes occur. Thus, our solution does not permanently suppress any route.

2. Minimal revealing of policies: If the input routes of a router are known, the precedence of the advertised route indicates whether the chosen route is the most preferred: if not, then its value increases. No additional information is revealed compared to BGP today: given the inputs, a route is not the most preferred if it is not advertised.

For routes stored in history, 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 indicate the routers having policy conflicts and do not contain any more information. Thus, we do not expose any additional ISP policies.

3. No requirement for global knowledge: Dispute detection operates solely on route advertisements received from neighbors, and are fully decentralized. No third party is required to gather and compute optimal routes for all ISPs. The route numbers are propagated upstream only for the purposes of troubleshooting, and do not affect nodes elsewhere, including other parts of the wheel.

4. 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 or AS number gives the right amount of visibility to assist in network troubleshooting. If a node is forced to select a less preferred route, it appends its route number to those already associated with the selected route, otherwise the existing numbers are 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.



AS Path	Global Precedence	Local Precedence	Feasible
P_0	1	0	true
P_3	0	1	true
...	...	...	...
P_{n-2}	0	1	false
P_{n-1}	1	1	true

AS Path	Global Precedence	Local Precedence	Feasible
P_0	1	0	true
P_1	1	1	true
P_2	0	1	true
P_3	0	1	false
...	...	...	...
P_{n-2}	0	1	false
P_{n-1}	1	1	true

Figure 12: (a) Before and (b) after a history table is updated. The routes with the lowest global values are first selected, after which ties are broken using the BGP decision process of today. Only feasible routes can be chosen.

```
1: let current router counter be C_c
2: if local routing has converged then
3:     for each entry in history table do
4:         if not feasible then
5:             remove entry
6: for each route R received do
7:     if route from neighbor N is filtered then
8:         set previous feasible route R_p from N infeasible
9:     else if different route received from neighbor N then
10:        set previous route R_p from N infeasible
11:        compute R's local precedence
12:        insert R into history table, set feasible
13:    else if same route received from neighbor then
14:        if previous feasible route R_p ≠ R then
15:            set R_p infeasible
16:        else
17:            update R's global precedence value
18:        set R feasible
19:    else if first route R is received from neighbor then
20:        compute R's local precedence
21:        insert R into history table, set feasible
22: select set S of eligible routes
23: select set S' with lowest global precedence, S' ⊂ S
24: select route R with lowest local precedence, R ∈ S'
25: if local precedence of R ≠ 0 then
26:     associate with R the router counter C ← C_c + 1
27: return R
```

Figure 13: Pseudo-code for updating history table and determination of the selected route for each destination.

6. ROUTER CHANGES

In this section we describe extensions to a BGP router necessary to implement our solution. We elaborate on the primary additional components, including the history table, router counter, and the adaptive convergence window, in this section.

6.1 History Table

The history table stores routes received from neighbors, as well as information necessary for dispute detection. 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. We call a route currently available and advertised by a neighboring router *feasible*. Routes that have recently been withdrawn, but which has not timed out and hence still present in the history table are called *infeasible*.

Figure 12 shows an example of a history table being updated, and Figure 13 provides the pseudo-code. Entries in the table are sorted in order of local precedence, that is, the ordering is determined using the same rules as the BGP decision process in use today. This ordering provides the local precedence value: the most locally preferred has value 0, the rest have value 1.

```
1: let incoming route be R_i
2: let associated router counter be C_i
3: let current largest encountered router counter be C_c
4: if R_i includes current AS (i.e. loop) then
5:     return
6: (Note: routes filtered due to policies are still considered.)
7: C_c ← max(C_c, C_i)
```

Figure 14: Pseudo-code for updating router counter for each destination prefix.

6.2 Router Counter

A router counter is associated with each destination prefix, and is initialized and has a lower bound value of 1. As shown in the pseudo-code (Figure 14), it is updated with each incoming route, and cleared whenever routing is deemed to have converged. The router counter is advertised together with the associated route only when the router is in the detection phase (§5.1), and when it advertises a route with incremented precedence value, that is, when it may be a pivot node.

6.3 Adaptive Convergence Window

As elaborated in §5, infeasible routes are kept in memory for a period of time in order to detect disputes. We call this period of time the *convergence window*. Assuming that one-hop route propagation delay W is similar to the Minimum Route Advertisement Interval (MRAI), the window size should be proportional to the number of nodes traversed by the router counter value (C_r), or WC_r.

The convergence window begins with a short duration (one MRAI), so that networks not containing disputes can converge relatively quickly. At the end of each convergence window, any updates to the advertised route implies that routing has not converged, and the next window is set to be the size of the current C_r. Lastly, the window size is reset after the network converges.

7. EVALUATION

We next evaluate the performance of our solution, describing our simulator, methodology as well as the performance metrics used.

7.1 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 **??** shows the main steps of the batch update process, whereas Figures 13 and 14 describe maintenance of the history and router counters respectively. We set MRAI to 30 seconds, processing delay jitter to 1 second, and link propagation delay to 10 milliseconds.

7.2 Methodolgy

To better understand the basic performance of our solution, we use simple graphs, which consist only of rim, pivot and destination nodes. 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 our solution in practice, we use an AS-level network topology constructed using routing

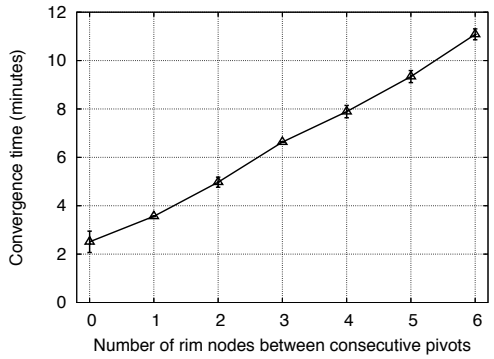


Figure 15: Simple graphs: 3-pivot network's convergence time against rim-to-pivot ratio.

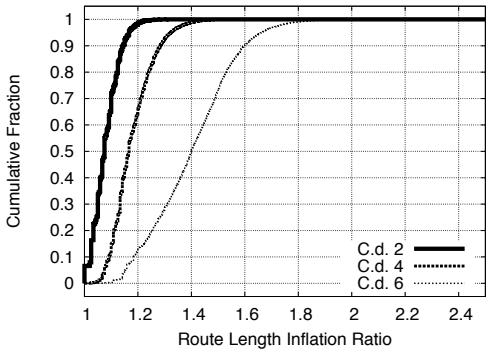


Figure 16: Increase in the maximum depth of constrained depth-first routing results in inflated routes.

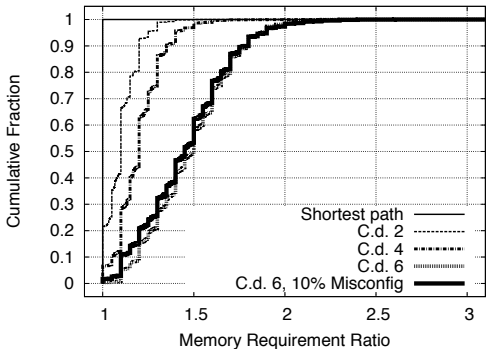


Figure 17: Deviation of path lengths from shortest increases exploration and hence more results stored.

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 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 breadth-first search 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.

7.3 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.

7.4 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 15, 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 16. 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 17, 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 due to misconfigurations, we randomly assigned next hop preferences to 10% of the nodes, verified that dispute wheels do exist (normal BGP does not converge), and that the networks converged when Precedence is used. As shown in Figure 17, we required approximately the same amount of memory as before.

Finally, we looked at the network convergence times (Figure 18). As we expected, local preferences assigned based on shortest-paths results in faster convergence. More importantly, convergence time is not significantly affected by

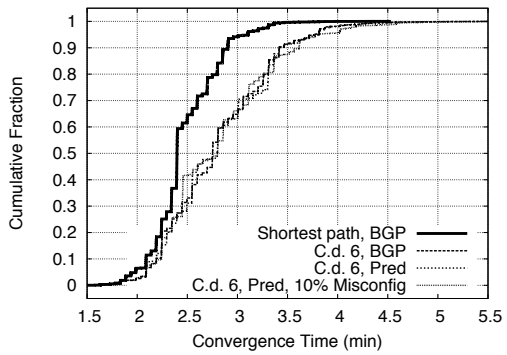


Figure 18: Usage of the precedence metric does not delay convergence, even in the presence of misconfigurations.

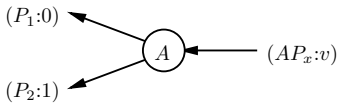


Figure 19: Detection of misbehavior can be performed by observing incoming and outgoing routes. Here, the arrows indicate direction of possible paths to destination.

usage of Precedence, nor by the presence of misconfigurations (disputes) in the network.

8. DISCUSSION

A major question that naturally arises when considering entities that act independently to maximize profits is that of misbehavior. In this section we discuss the detection of misbehavior, as well as the additional modifications to the protocol required in order to remove the constraint where an AS is represented by a single node.

8.1 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 19, 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.

(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.

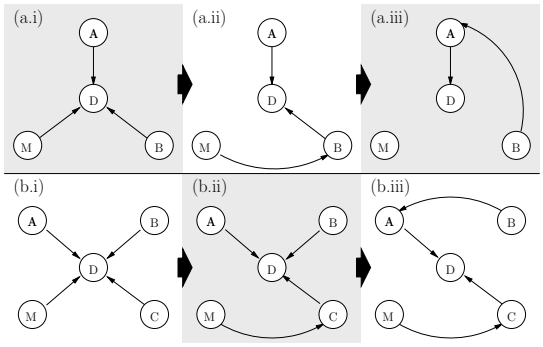


Figure 20: (a) For disputes with odd numbers of nodes, M eventually lacks a route if it initially filters the spoke one. (b) For disputes with even numbers of nodes, M does not destabilize the network.

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.

8.2 Adaptive Filtering

Misbehavior that is more difficult to detect involves *adaptive filtering*. 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.

Two scenarios can occur as illustrated in Figure 20. 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 thus also if the 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.

8.3 Misbehavior Detection

Most ASes are comprised of multiple routers and thus the usual assumption that an AS can be modeled by a single router does not hold. For instance, in Figure 21(i), router A selects, as it should, the less preferred route (R_0:0) and advertises (AR_0:1) to B, which then 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

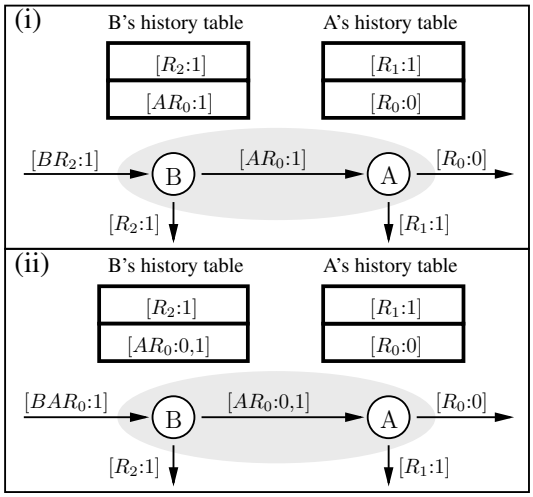


Figure 21: Tagging routes with both ingress and egress precedence values causes a multiple-router AS' behavior to be similar to that of a single router.

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 21(ii) shows the same network with the tweaked protocol. Here, correct behavior will cause A to advertise (AR_0:0), 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 §8.1 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.

9. 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.

10. 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.