

# Enforcing Convergence in Inter-Domain Routing

Jorge A. Cobb      Ravi Musunuri  
Department of Computer Science  
The University of Texas at Dallas  
Richardson, TX-75083-0688  
Email: {cobb, musunuri}@utdallas.edu

**Abstract**—The stable-paths problem is an abstraction of the basic functionality of the Internet’s BGP routing protocol. This abstraction has received considerable attention, due to the instabilities observed in BGP. In this abstraction, each node informs its neighboring nodes of its current path to the destination node. From the paths received from its neighbors, each node chooses the best path according to some locally chosen routing policy. However, since routing policies are chosen locally, conflicts may occur between nodes, resulting in unstable behavior. Current solutions either require expensive path histories, or prevent nodes from locally choosing their routing policy. In this paper, we present a solution with small overhead, and furthermore, each node has the freedom to choose any routing policy. However, to avoid instabilities, the possibility of divergence is measured using an efficient cost metric exchanged between nodes. If the cost metric indicates that divergence is occurring, steps are taken to ensure convergence.

## I. INTRODUCTION

The Internet is an inter-connected collection of Autonomous Systems (AS’ms). To route datagrams from one AS to another, each AS learns a path to every other AS. This learning is achieved by exchanging routing information between neighboring AS’ms. In particular, each AS informs its neighbors of its current path to each destination. Therefore, each AS stores in memory the full path (i.e. sequence of AS’ms) that must be traversed to reach each destination AS. The Border Gateway Protocol (BGP) [1] is the de-facto routing protocol for sharing AS path information between neighboring AS’ms.

BGP has been plagued by many forms of unstable behavior. In [2], it is shown how misconfigured routers and other software anomalies can cause BGP to generate orders of magnitude more routing message overhead than necessary. In addition, it is shown in [3] how maintaining inconsistent routing information in the memory of a router leads to the slow convergence after the failure of a route.

Our focus in this paper is a problem of even greater impact in BGP: the permanent failure to converge to a stable set of routes for the following reason. Given that each AS knows the full path to the destination, each AS has the freedom to implement routing policies. That is, each AS has a local preference of one path over another. These local preferences are often based on commercial relationships between AS’ms, or on other factors such as security. Given that the routing policies are chosen locally at each AS, it is possible that neighboring AS’ms may have conflicting routing policies. It is even possible for the conflicts between routing policies to form a circular chain of conflicts [4], [5]. When this

occurs, it is possible for BGP to diverge, that is, there are AS’ms that continuously alternate between different paths to the destination, and are perpetually unable to obtain a stable path.

Internal BGP (IBGP), i.e., the distribution of BGP routing information within an AS, also suffers from divergence problems [6], [7], [8], [9]. However, the IBGP Divergence problem is internal to each AS, while on this paper we focus on inter-AS routing.

Many schemes have been proposed to alleviate or avoid the inter-AS divergence problem of BGP. Route-flap-damping [10] reduces the rate at which oscillations may occur, but does not eliminate them. Pre-checking the conflicts in routing policies [11] between of Internet Service Providers (ISPs) eliminates divergence. However, ISPs are often unwilling to share their local routing policies with others, and it has been shown that checking the convergence of a routing policy is NP-complete [5]. Other solutions restrict the structure of the routing policies [12] or restrict the choice of paths [13] at run time. These solutions remove the freedom the routing policy freedom of the original BGP protocol. Finally, path histories [14] may be carried with each path update message. This, however, may significantly increase message and memory overhead, and furthermore, it may unnecessarily remove a path from the allowed set of paths of an AS.

In this paper, we propose a new technique to prevent the divergence problem in BGP. In our solution, nodes are free to follow their routing policy temporarily. During protocol execution, the possibility of divergence is measured using a simple cost metric exchanged between nodes. If the cost metric indicates that divergence is occurring, steps are taken to ensure convergence. However, no path is removed from the set of allowed paths of a node, minimizing the probability that a node becomes disconnected from the destination. Overhead is kept at a minimum, since only an additional integer needs to be added to BGP’s path-update messages.

The paper is organized as follows. In Section II, we outline the stable paths problem (SPP), defined in [5], as an abstraction of BGP behavior. In Section III, we present examples of SPP divergence. Related work is discussed in Section IV. Our notation to specify our protocols is given in Section V. In Section VI, we present two protocols to address the divergence problem in order of increasing complexity. Correctness proofs of our protocols are omitted due to space restrictions. Resetting our cost metric is discussed in Section VII. We conclude in

Section VIII with a summary and remarks.

## II. STABLE PATHS PROBLEM (SPP)

Griffin et. al. [4], [5], [15] defined the Stable Paths Problem (SPP) to study the divergence problem in BGP. Although an autonomous system is a collection of multiple routers, the behavior of all these routers is consistent. This allows an autonomous system to be simply represented as a single node, and the Internet to be represented as a graph  $G$ ,  $G = (V, E)$ , where the set  $V$  of nodes represents the autonomous systems, and the set  $E$  of edges represents the neighboring relationships between autonomous systems. Without loss of generality, a single destination node is assumed. We will refer to this node as *root*.

A path is simply a sequence of nodes. A path is said to be *rooted* if it is a simple path whose last node is the root. In general, the objective of each node is to find a path from itself to *root* that is consistent with the path chosen by its next-hop neighbor. That is,  $u$  may chose the path  $P$ , where  $P = \langle u, v, w, \dots, root \rangle$ , only if  $v$  currently has chosen the path  $\langle v, w, \dots, root \rangle$ . Thus, each node stores in its memory its chosen rooted path, and it advertises this path to its neighbors.

The set of paths currently chosen by all nodes is denoted by  $\pi$ , and in particular,  $\pi(u)$  denotes the path currently chosen by node  $u$ . It is possible that  $\pi(u)$  is empty, denoted by  $\langle \rangle$ , in the event that no neighbor of  $u$  has a path to *root*. The set  $\pi$  must satisfy the following conditions at all times.

*Constraint 1:* For every node  $u$ ,

- $\pi(u)$  is a rooted path whose first node is  $u$ .
- For any two consecutive nodes  $v$  and  $w$  in  $\pi(u)$ ,  $(v, w) \in E$ .

□

As mentioned earlier, each autonomous system is free to choose from the paths offered by its neighbors according to a routing policy that is locally defined at the autonomous system. We represent the combined routing policy of all nodes by the ranking relation  $\prec$  on paths. In particular, if  $P$  and  $Q$  are paths from  $u$  to *root*, then  $P \prec Q$  denotes that  $u$  prefers path  $Q$  over path  $P$ . The ranking relation  $\prec$  must satisfy the following conditions.

*Definition 1:* Relation  $\prec$  is defined over rooted paths and  $\langle \rangle$ , and satisfies the following.

- $\prec$  is transitive and irreflexive.
- For every node  $u$  and every pair of rooted paths  $P$  and  $Q$  originating at  $u$ ,

$$P \prec Q \vee Q \prec P$$

- For every rooted path  $P$ ,

$$P \prec \langle \rangle \vee \langle \rangle \prec P$$

□

Note that if  $P$  originates at  $u$  and  $P \prec \langle \rangle$ , then  $u$  will never choose  $P$ .

Below, we formally define the set of paths from which a node may choose its rooted path, and given that nodes are

greedy, the node will choose the highest ranked path from this set.

*Definition 2:* For any path  $P$ ,  $P \in \text{choices}(u, \pi)$  iff

$$P = \langle \rangle \vee (\exists v, (u, v) \in E, P = \langle u, \pi(v) \rangle)$$

□

Informally, set  $\text{choices}(u, \pi)$  contains all rooted path originating at node  $u$  that are consistent with the current paths chosen by the neighbors of  $u$ .

*Definition 3:* For any path  $P$ ,  $\text{best}(u, \pi) = P$  if and only if,

$$(\forall Q, Q \in \text{choices}(u, \pi), Q = P \vee Q \prec P)$$

□

Informally, path  $\text{best}(u, \pi)$  is the highest ranked path in  $\text{choices}(u, \pi)$ . Therefore, whenever the value of  $\text{best}(u, \pi)$  changes, node  $u$  will change its rooted path to  $\text{best}(u, \pi)$ .

## III. DIVERGENCE OF SPP INSTANCES

An *instance* of the SPP problem consists of a pair  $(G, \prec)$ , where  $G$  is the graph representing autonomous systems and  $\prec$  is the path ranking relation. Since the rank of each path is chosen arbitrarily at each node, conflicting choices at neighboring nodes may prevent nodes from maintaining a stable rooted path. That is, the path chosen by some nodes will vary continuously, even though neither  $G$  nor  $\prec$  change.

Consider the SPP instance in Figure 1.<sup>1</sup> The paths acceptable to a node (i.e. ranked higher than the empty path) are alongside the node in order of rank. Note that each node prefers longer paths over shorter paths. E.g.,  $u$  prefers the longer path  $\langle u, v, root \rangle$  over the shorter path  $\langle u, root \rangle$ . This causes the ranking of each node to be in conflict with the ranking of its next hop to *root*.

The cyclic relationship between these ranking prevents any node from obtaining a stable path to *root*. To see this, consider the following steps:

- Initially  $u$ ,  $v$ , and  $w$  choose the paths  $\langle u, v, root \rangle$ ,  $\langle v, root \rangle$ , and  $\langle w, root \rangle$ , respectively, as shown in Figure 1(a).
- Node  $v$  notices that  $w$  chose the path  $\langle w, root \rangle$ . Hence,  $v$  changes its path to  $\langle v, w, root \rangle$ . This in turn forces  $u$  to change its path to  $\langle u, root \rangle$  as shown in Figure 1(b).
- Node  $w$  notices that  $u$  chose the path  $\langle u, root \rangle$ . Hence,  $w$  changes its path to  $\langle w, u, root \rangle$ . This in turn forces  $v$  to change its path to  $\langle v, root \rangle$  as shown in Figure 1(c).
- Node  $u$  notices that  $v$  chose the path  $\langle v, root \rangle$ . Hence,  $u$  changes its path to  $\langle u, v, root \rangle$ . This in turn forces node  $w$  to change its path to  $\langle w, root \rangle$ , and the system is back to its initial state in in Figure 1(a).

Converging to a steady state is highly sensitive to the ranking of paths. For instance, in Figure 1, if the ranking of paths at  $u$  is reversed, then the system is guaranteed to converge to a steady state. As another example, consider

<sup>1</sup>This SPP instance is known as BAD GADGET in [4], [5].

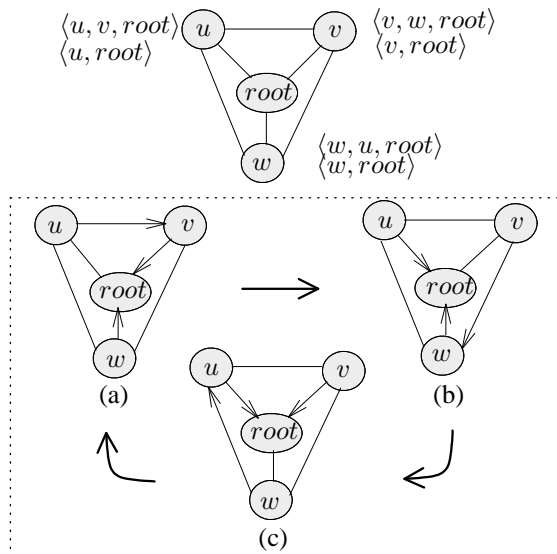


Fig. 1. Diverging SPP instance.

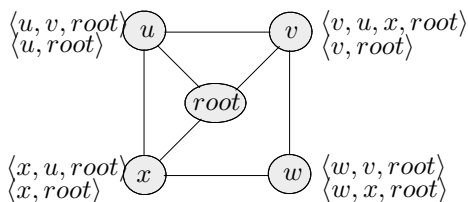


Fig. 2. SPP Instance that may diverge or converge.

Figure 2.<sup>2</sup> This SPP instance has the interesting property that, depending on timing, i.e., on the relative order in which nodes update their rooted path, it may either converge to a steady state, or continuously diverge. Again, if the ranking of paths at  $u$  is reversed, then a steady state is always reached. Due to this sensitivity to the ranking of paths, deciding if an SPP instance converges is NP-complete [5].

#### IV. RELATED WORK

Many solutions have been proposed to solve the divergence problem in BGP. We can divide divergence solutions into three different categories.

The first category avoids the divergence problem by pre-checking the conflicts in routing policies [11]. Internet Service Providers (ISPs) are expected to publish their routing policies in centralized routing registries. However, ISPs are often unwilling to share their local routing policies with others. Furthermore, as mentioned above, Griffin et al. [5] have shown that checking the convergence of a routing policy is an NP-complete problem.

The second category of solutions avoid the divergence problem by restricting the routing policies [12] or restricting the choice of paths [13].

Gao et al. [12] proposed a set guidelines for choosing routing policies based on the hierarchical structure of commercial

relationships between ISPs. This restriction ensures the routing policies are conflict-free, and thus convergence is assured. However, not all ISPs may desire to restrict their policies in this way, and the routing policy freedom of the original BGP protocol is removed.

In [13], although the routing policies are not restricted beforehand, the choice of paths at each node is restricted at run-time. This restriction is enforced at run-time via diffusing computations. Although this ensures that the system converges to a steady state, again, the routing policy freedom of the original BGP protocol is removed. In particular, the protocol prevents a sequence of path changes that does not necessarily causes the system to diverge. Furthermore, there is the extra overhead of diffusing computations.

The third category of solutions avoids the divergence problem by carrying path histories [14] with each path-update message. A possible divergence is detected by finding cycles in the path histories, and selected paths are removed to ensure convergence. This technique has several drawbacks. Path histories may significantly increase message and memory overhead. In addition, a cycle in the path history is a necessary but not sufficient condition for divergence, and hence, a divergence may be inferred even though it is still possible for the system to converge. Furthermore, a cycle in the path history may be caused by link failures and restorations and not by conflicts in the routing policies. This leads the system to unnecessarily remove a path from the list of acceptable paths at a node.<sup>3</sup> Lastly, path histories may partially reveal the routing policies of ISPs, which, as mentioned above, are preferred to be kept confidential.

As mentioned earlier, our protocols below do not restrict the routing policy that may be chosen at a node, they incur very small overhead, and do not remove any paths from the set of allowed paths at a node. However, they do restrict the behavior of a node at run time, but only in the event that divergence is inferred from the cost metric.

#### V. PROTOCOL NOTATION

Before presenting our protocols, we first give a short overview of the notation that we use in specifying network protocols<sup>4</sup>. For terseness and ease of presentation, our protocols are specified using a shared memory notation. They can be extended to a message passing model in a straightforward manner.

A protocol consists of a set of nodes  $V$  and a set of undirected edges  $E$ . We say that node  $u$  is a neighbor of node  $v$  iff  $(u, v) \in E$ . Each node is able to read, but not modify, the variables of its neighboring nodes. The behavior of each node is specified by a set of actions. Every action is of the form:

$$\langle \text{guard} \rangle \rightarrow \langle \text{command} \rangle$$

<sup>3</sup>The remove path may actually be crucial in the future if due to link failures it is the only available path.

<sup>4</sup>Our notation is loosely based on the guarded command notation of [16]

<sup>2</sup>This SPP instance is known as NAUGHTY GADGET in [4], [5].

```

node  $u$ 
begin
   $\pi(u) \neq best(u, \pi) \rightarrow \pi(u) := best(u, \pi)$ 
end

```

Fig. 3. Greedy Protocol

A guard is a boolean expression over the variables of the node and the variables of its neighboring nodes. An action is said to be enabled if its guard evaluates to true. A command is a sequence of assignment statements that assigns values to the variables of the node as a function of the variables of its neighboring nodes.

Assignment statements are of the following form:

$\langle \text{variable} \rangle := \langle \text{expression} \rangle$

where the variable is local to the node and the expression is a function of the variables of the neighboring nodes.

An execution step of a protocol consists of arbitrarily choosing an enabled action out of all actions of all processes, and executing the command of this action. An execution of a protocol consists of a maximal sequence of execution steps, that is, either the sequence is infinite, or it ends in a state where no action is enabled.

We assume all executions of a protocol are weakly fair. That is, an action whose guard is continuously true must be eventually executed.

## VI. ENFORCING CONVERGENCE

Before presenting our protocols, we specify the basic behavior of the BGP protocol using our notation. Recall that each node in BGP has the objective of obtaining the highest-ranked rooted path. This behavior is represented by the greedy protocol in Fig. 3. This is a simple protocol where each node  $u$  contains one action. The guard of the action checks if the current path of  $u$ , i.e.,  $\pi(u)$ , is different from the best path for  $u$ , i.e.,  $best(u, \pi)$ . If so, the command of the action sets  $\pi(u)$  to  $best(u, \pi)$ . As mentioned above, this protocol will diverge for certain SPP instances.

We next present our protocols. We do so in two steps. First, we examine how divergence occurs, and we enhance the greedy protocol by adding a measure of this divergence in the form of a cost associated with each node. We show that this cost grows without bound whenever the protocol diverges. We call this protocol the Divergence Detection Protocol. Finally, we place limits on the cost increase in order to ensure convergence, resulting in the Bounded Divergence Protocol.

### A. Divergence Detection Protocol

Consider again Fig. 1 and its cyclic sequence of steps. Observe that the rank of the path of each node in the cycle is periodically decreased. For example, when the state changes from Fig. 1(a) to Fig. 1(b), the rank of  $\pi(u)$  decreases. As long as the execution continues along this cycle, the rank of  $\pi(u)$  decreases periodically. Intuitively, no divergence is

```

node  $u$ 
begin
   $\pi(u) = \langle u, \pi(next(\pi(u))) \rangle \rightarrow$ 
   $cost(u) :=$ 
   $max(cost(u), cost(next(\pi(u))))$ 
   $\square$ 
   $\pi(u) \succ best(u, \pi) \wedge$ 
   $next(\pi(u)) \neq next(best(\pi, u)) \rightarrow$ 
   $\pi(u) := best(u, \pi);$ 
   $cost(u) := cost(u) + 1$ 
   $\square$ 
   $\pi(u) \prec best(u, \pi) \vee$ 
   $next(\pi(u)) = next(best(\pi, u)) \rightarrow$ 
   $\pi(u) := best(u, \pi);$ 
   $cost(u) := cost(next(\pi(u)))$ 
end

```

Fig. 4. Divergence Detection Protocol

possible if every node monotonically increases the rank of its path, because, eventually, the node would reach and keep its highest ranking path. Therefore, during divergence, the rank of the rooted path of diverging nodes must periodically decrease.

We use the above observation to allow nodes to infer that divergence is occurring. In particular, each node is assigned an integer cost. Whenever the new path of a node has a lower rank than its previous path, the cost of the node is monotonically increased. In addition to the rooted path of its neighbors, a node may read the cost of its neighbors.<sup>5</sup> As these costs increase, a node infers that divergence is occurring, and it takes remedial action by restricting its choice of paths. This is further discussed in Section VI-B.

A strict method to update the cost of  $u$  would be to prevent the cost of  $u$  from ever decreasing, and increase it by one whenever its path decreases in rank. In this way, if cyclic executions, such as those in Fig. 1, are encountered, then  $u$ 's cost is guaranteed to increase. The disadvantage of this, however, is that  $u$ 's cost does not decrease even in the event that an alternative path is found that does not lead  $u$  to diverge. Therefore, we have chosen the method below of updating the cost of  $u$  in a manner that does allow the cost of  $u$  to decrease in some instances when a non-diverging alternative path is found.

The specification of the Divergence Detection Protocol (DDP) is shown in Fig. 4. Each node  $u$  consists of three actions. The first action simply enforces that the cost of a node is never smaller than the cost of its next node along its path to *root*.

The second action is enabled when the best path for  $u$  has a rank lower than  $u$ 's chosen path  $\pi(u)$ , and  $u$  will have a new next hop node along the best path. In this case,  $\pi(u)$  is updated to the best path, and in addition, since the path of  $u$

<sup>5</sup>This would be implemented in message passing by including the cost of the node in every path update message sent.

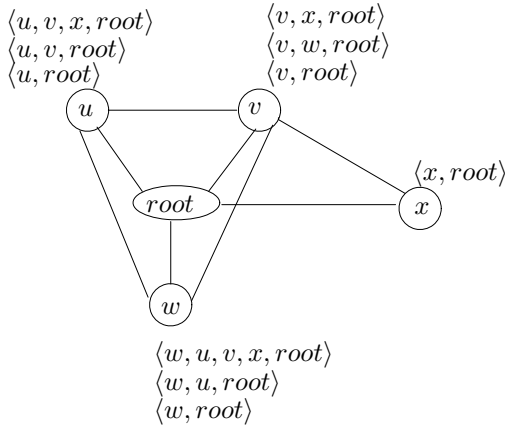


Fig. 5. SPP Instance with escape path

decreases rank,  $u$  increases its cost by one.

Next, the third action is enabled under two conditions. First, it is enabled when the best path of  $u$  is ranked higher than its current path  $\pi(u)$ . In this case,  $u$  must also update  $\pi(u)$  to the best path, but it simply sets its cost to the cost of the next node along the new best path. Note that in this case the cost of  $u$  may decrease. Second, the action is also enabled when the rank of the best path of  $u$  is lower than the rank of its current path, but in this case, the next node is the same in both paths. Here, the same assignment of values are performed as in the first case. Note again that the cost of  $u$  may decrease.

As an example, consider Fig. 5. The potential for a cyclic execution similar to that of Fig. 1 exists, since the SPP instance in Fig. 5 is a superset of the SPP instance in Fig. 1. However, note that a stable state is always reached in Fig. 5, because node  $x$  will choose path  $\langle x, \text{root} \rangle$ , and this causes all other nodes to reach  $\text{root}$  via  $x$ . Thus, if the information from  $x$  to  $v$  is slow to arrive, the system may execute the cyclic steps of Fig. 1 temporarily, but as soon as the information from  $x$  arrives to  $v$ , the system converges to a steady state.

In the above example, we would like the costs of all nodes to become zero. In this way, the costs will not hinder the choice of paths at each node. To ensure costs become zero, the third action in Fig. 4 assigns to  $\text{cost}(u)$  the same cost as the cost of the next-hop node of its new path. In this manner, if the cost of its new next-hop is low, then the cost of  $u$  itself will be low.

We must ensure, however, that even though  $\text{cost}(u)$  is allowed to decrease in the third action, if the system diverges, then  $\text{cost}(u)$  must increase monotonically in order to notify  $u$  of divergence. This actually holds, as stated formally below.

*Theorem 1:* For any node  $u$  in the DDP, if  $\pi(u)$  changes infinitely often, then  $\text{cost}(u)$  grows beyond bound.  $\square$

To quickly illustrate the above theorem, consider again Fig. 1(a). Let this be the initial state of the system, and the cost of all nodes be zero. The transition from Fig. 1(a) to Fig. 1(b) causes  $v$  to increase the rank of its path, and hence,  $\text{cost}(v)$  is set to  $\text{cost}(w)$ , i.e., it remains zero. However, it

causes the rank of  $u$ 's path to decrease, and  $\text{cost}(u)$  is set to one. The transition from Fig. 1(b) to Fig. 1(c) causes  $w$  to increase the rank of its path, and hence,  $\text{cost}(w)$  is set to  $\text{cost}(u)$ , i.e., to one. Furthermore, it causes the rank of  $v$  to decrease, and  $\text{cost}(v)$  is increased to one. Finally, the transition from Fig. 1(c) to Fig. 1(a) causes  $u$  to increase the rank of its path, and hence,  $\text{cost}(u)$  is set to  $\text{cost}(v)$ , i.e., it remains one. However, it causes the rank of  $w$ 's path to decrease, and  $\text{cost}(w)$  is set to two. Therefore, the cost of all three nodes increases, as desired.

## B. Bounded Divergence Protocol

The DDP detected divergence, but it did not modify its behavior to ensure convergence. We next present the Bounded Divergence Protocol (BDP), which ensures that the system converges to a steady state.

Given that costs increase when the system diverges, it is evident that the system should restrict its behavior when costs become large. One option is for a node to stop changing its path, even if a path with higher rank exists, whenever its own cost is beyond a certain threshold. The disadvantage of this is that if there is a path that offers an escape of the cyclic behavior, then once the node's cost reaches the threshold, the node would be unable to take the escape path.

Instead, we choose to prevent a node from choosing a new path if the cost of its new next-hop neighbor is greater than some threshold. In this case, the escape path will likely contain a low cost, and therefore, the node is free to choose the escape path, and the cyclic behavior is broken.

Figure 6 shows the specification for the BDP at node  $u$ . The specification is similar to that of DDP. For ease of presentation, the third action of the DDP has been partitioned into two actions, one for each case in which the third action of the DDP is enabled.

The first three actions are the same as in the DDP. Note that in the second and third actions, the best path of  $u$  has a lower rank than the current path  $\pi(u)$ . Note that this case only occurs when  $\pi(u)$  is no longer available, otherwise,  $\pi(u)$  would be the best path available to  $u$ . Therefore,  $u$  has no choice but to adopt the best path, since we are required to maintain a path to the destination if one is available.

On the other hand, in the fourth action, the best path of  $u$  is ranked higher than  $\pi(u)$ . Notice that in this case  $u$  has two options: take the new path or maintain its current path. We choose to take the new path only if the cost of the next-hop neighbor along the new path is less than a maximum threshold  $C$ . Otherwise,  $u$  keeps its current path. The exception to this is when  $\pi(u)$  is the empty path. In this case, since  $u$  is required to maintain a path to  $\text{root}$ , the best path is chosen irrespective of the cost of the next-hop neighbor.

The above restrictions ensure that the system reaches a steady state, as indicated below.

*Corollary 1:* Starting from any arbitrary system state (i.e., an arbitrary value of  $\pi$  and node costs), the BDP converges to a stable state within a finite number of steps.  $\square$

```

node  $u$ 

begin
   $\pi(u) = \langle u, \pi(\text{next}(\pi(u))) \rangle \rightarrow$ 
   $\text{cost}(u) :=$ 
     $\text{max}(\text{cost}(u), \text{cost}(\text{next}(\pi(u))))$ 
[]
   $\pi(u) \succ \text{best}(u, \pi) \wedge$ 
   $\text{next}(\pi(u)) \neq \text{next}(\text{best}(\pi, u)) \rightarrow$ 
   $\pi(u) := \text{best}(u, \pi);$ 
   $\text{cost}(u) := \text{cost}(u) + 1$ 
[]
   $\pi(u) \succ \text{best}(u, \pi) \wedge$ 
   $\text{next}(\pi(u)) = \text{next}(\text{best}(\pi, u)) \rightarrow$ 
   $\pi(u) := \text{best}(u, \pi);$ 
   $\text{cost}(u) := \text{cost}(\text{next}(\pi(u)))$ 
[]
   $(\pi(u) \prec \text{best}(u, \pi) \wedge$ 
   $(\text{cost}(\text{next}(\text{best}(u, \pi))) < C \vee \pi(u) = \langle \rangle) \rightarrow$ 
   $\pi(u) := \text{best}(u, \pi);$ 
   $\text{cost}(u) := \text{cost}(\text{next}(\pi(u)))$ 
end

```

Fig. 6. Bounded Divergence Protocol

## VII. RESETTING THE COST

We have shown how the cost increases up to  $C$  in the case of routing conflicts. However, we have not addressed how to decrease this cost once the routing conflicts disappear. Routing conflicts may disappear because routing policies may change, or the network topology may change.

The cost of a node will automatically reduce if it has a neighbor whose path is ranked higher than its own, and whose cost is small. However, if no such neighbor exist, we would still like to be able to reduce the cost of the node. This will allow the node to follow one more time its routing policies, and perhaps this time no conflicts will exist.

In consequence, periodically, the cost of all nodes should be reset to zero. This can be done via a distributed reset protocol [17]. The overhead required to perform such a reset is small. Namely, a minimum-hop spanning tree needs to be maintained. This tree can be maintained by each node simply storing the identity of its parent on the tree and a hop count to the root of the tree (namely, node *root*). Then, a command to reset all costs to zero is propagated from the root to the leaves via diffusing computations.

Once the costs of all nodes is zero, the nodes will proceed to change their paths according to their routing policy. If no conflicts exist, the costs will remain low. If cyclic conflicts exist, the costs will again reach the threshold  $C$ .

Note that doing a reset need not be done often. It could be done only on demand, i.e., only if there is at least one node whose cost has reached threshold  $C$ . Note that if a node reaches its threshold  $C$ , it is not urgent to reduce its

cost, since the node will still have a path to the destination, but it will simply not be able to follow the original routing policy. Therefore, this reset could be done on demand and infrequently, say, no more than once a week or once a month.

## VIII. SUMMARY AND CONCLUDING REMARKS

In this paper, we have addressed the problem of enforcing convergence in inter-domain routing. This convergence is enforced after divergence is being detected in the system due to the increase in a simple cost metric. Our solution only requires every path-update message to carry a small additional integer. Hence, our solution has very low communication and memory overhead compared to other dynamic solutions based on path histories and restrictions on path selection.

In our future work, we plan to perform simulations to evaluate the effectiveness of different values of  $C$ . In addition, we have focused on the single-node model of an AS. An AS, in practice, consists of a large collection of routers, and path information is distributed from the border routers to the interior routers using internal BGP (iBGP). iBGP also has been known to have divergence problems, and multiple solutions have been proposed. We plan to investigate how our solution to the external BGP divergence problem may interact with solutions for the iBGP divergence problem.

## REFERENCES

- [1] Y. Rekhter and T. Li, "A border gateway protocol," *IETF RFC-1771*, 1995.
- [2] C. Labovitz, G. R. Malan, and F. Jahanian, "Internet routing instability," in *Proc. of ACM SIGCOMM conference*, 1997, pp. 115–126.
- [3] D. Pei, X. Zhao, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Improving BGP convergence through consistency assertions," in *Proc. of IEEE INFOCOM conference*, vol. 2, 2002, pp. 902–911.
- [4] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "Policy disputes in path vector protocols," in *Proc. of IEEE ICNP conference*, 1999, pp. 21–30.
- [5] —, "The stable paths problem and interdomain routing," *IEEE/ACM Trans. Networking*, vol. 10, no. 2, pp. 232–243, 2002.
- [6] T. G. Griffin and G. Wilfong, "On the correctness of IBGP configuration," in *Proc. of ACM SIGCOMM conference*, 2002, pp. 17–29.
- [7] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route oscillations in ibgp with route reflection," in *Proc. of ACM SIGCOMM conference*, 2002, pp. 235–247.
- [8] T. G. Griffin and G. Wilfong, "Analysis of the MED oscillation problem in BGP," in *Proc. of IEEE ICNP conference*, 2002, pp. 90–99.
- [9] R. Musunuri, , and J. A. Cobb, "Complete solution to IBGP stability," in *Proc. of IEEE ICC conference*, vol. 2, 2004, pp. 1177 – 1181.
- [10] C. Villamizar, R. Chandra, and R. Govindan, "BGP route flap damping," *IETF RFC-2439*, 1998.
- [11] K. Varadhan, R. Govindan, and D. Estrin, "Persistent route oscillations in inter-domain routing," *Computer Networks*, vol. 32, pp. 1–16, 2000.
- [12] L. Gao and J. Rexford, "Stable internet routing without global coordination," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 681–692, 2001.
- [13] J. A. Cobb, M. G. Gouda, and R. Musunuri, "A stabilizing solution to the stable paths problem," in *Symp. on Self-Stabilizing Systems, Springer-Verlag Lecture Notes in Computer Science*, vol. 2704, 2003, pp. 169–183.
- [14] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "A safe path vector protocol," in *Proc. of INFOCOM conference*, 2000, pp. 490–499.
- [15] T. G. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proc. of ACM SIGCOMM conference*, 1999, pp. 277–288.
- [16] M. G. Gouda, *Elements of Network Protocol Design*. John Wiley & Sons, 1998.
- [17] A. Arora and M. G. Gouda, "Distributed reset," *IEEE Trans. Comput.*, vol. 43, no. 9, pp. 1026–1038, 1994.