Information Fusion xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Resolving inter-domain routing policy disputes at run-time with scalability consideration

Huaming Guo [a], Hongke Zhang [a], Chi-Yuan Chen [b], Han-Chieh Chao [b,c,*], Wei-Ming Chen [c]

[a] Next Generation Internet Research Center, Beijing Jiaotong University, PR China
[b] Department of Electrical Engineering, National Dong Hwa University, Hualien, Taiwan
[c] Institute of Computer Science & Information Engineering, National Ilan University, I-Lan, Taiwan

ARTICLE INFO

Article history:
Received 28 March 2008
Received in revised form 2 September 2009
Accepted 17 December 2009
Available online xxxx

Keywords:
Inter-domain routing
BGP
Policy disputes

ABSTRACT

Thousands of *autonomous systems* (ASes) cooperate with each other to provide global Internet connectivity. The Border Gateway Protocol (BGP) is currently the only inter-domain routing protocol deployed on the Internet. It allows ASes to select and propagate routes based on flexible and locally defined policies. However, the flexibility and freedom of policies can lead to routing instability, even policy disputes among several ASes, causing inter-domain routing oscillations. Recent studies enforce global and local constraints on policies without freedom, or require expensive memory consumption and huge numbers of message exchanges. In this paper, we propose a run-time solution that operates with small overhead, guarantees safe convergence and preserves policy freedom and privacy as much as possible. The proposed scheme uses a distributed mechanism for detecting policy disputes and does this only when policy disputes exist. ASes dynamically compresses the dispute routes for safe convergence.

1. Introduction

The Internet connects thousands of *autonomous systems* (ASes). The Border Gateway Protocol (BGP) [1] is the de facto standard for establishing connectivity between ASes currently. ASes choose a path to each destination based on their individual local routing policies. These policies are constrained by commercial agreements, performance decisions, security concerns or other considerations. Further, BGP allows ASes to apply their policies autonomously without revealing their policies to others. However, Varadhan et al. [2] showed that without coordination ASes may have conflicting BGP policies, which cause persistent route oscillations in BGP. When this instability occurs, even if there are no changes to the network topology and routing policies, BGP would endlessly exchange update messages, persistently oscillate among several route choices and permanently fail to converge to a stable path to a destination, which may significantly degrade Internet performance.

Understanding and preventing this kind of routing anomaly has been the subject of much recent study. Many solutions were proposed to guarantee stability [4–7,12–16]. Broadly speaking, these solutions can be divided into three main categories.

The first approach statically analyzes all routing policies, and decides whether they have policy conflicts that lead to instability, such as [12]. Govindan et al. provided a common language for describing the route policies to publish routing policies of all ASes, and then used a set of software to analyze the impact of these policies. However, this approach has two disadvantages. First, ASes are unwilling to widely share their routing policies. Second, deciding the safety of a routing policy is a NP-complete or NP-hard problem even if there is complete knowledge of the routing policies, as proven by Griffin [3,4]. This approach is very impractical.

The second approach needs all ASes to restrict policies according to the same rules. Gao and Rexford restricted the routing policies to an underlying business hierarchy [5,6]. They [10] divided the AS relationships between ASes into: (1) *customer–provider* (i.e., a customer pays its provider for connectivity to the rest of Internet), (2) *peer-to-peer* (i.e., a pair ASes mutually provide connectivity to the rest of the Internet for each other); and (3) *backup* (i.e., a pair ASes provide backup connectivity to the Internet for each other in the event that one AS's connection to its provider fails). The rules for selecting and exporting routes are constrained according to the hierarchy, and customer–provider cycles are not allowed. However, this commercial relationship may violate the hierarchy due to complex situations. These rules disallow the use of many routes, so all ASes may not wish to restrict their policies in this way. Furthermore, minor changes to the business relationship between ASes or misconfiguration could still lead to instability.

* Corresponding author. Address: Institute of Computer Science & Information Engineering, National Ilan University, No. 1, Sec. 1, Shenlung Road, I-Lan, Taiwan.
E-mail address: hcc@niu.edu.tw (H.-C. Chao).

1566-2535/$ - see front matter © 2009 Published by Elsevier B.V.
doi:10.1016/j.inffus.2009.12.004

The third approach can dynamically detect and suppress conflicting policies at runtime. This approach does not require a priori restrictions. Rather than defining a suitable hierarchy under a variety of marketplaces, or eliminating policy autonomy because of the potential to induce route oscillations, dynamic detect can provide more autonomy to choose routing policy. Many solutions use dynamic detection. Griffin et al. [4] proposed a solution, called Safe Path Vector Protocol (SPVP), which detects oscillations through loops in the *route histories*, that record the changes in route choices in route propagation and then removes the loop routes. However, maintaining *route histories* significantly increases the message overhead. *Route histories* also reveal partial local policies, which ASes may want to keep private. Another run-time solution used diffused computation [7]. An AS asks any other AS whose path currently traverses it if a change in path is acceptable. However, this solution is not reasonable in practice because it also needs a lot of message exchanges and requires more convergence time. Some solutions [13,14] used the frequency of adopted or abandoned routes to detect oscillations. Routes were suppressed if the frequency was larger than a predefined threshold. However, this threshold value is not easily defined. If the value is too large, the system would oscillate more. If too small, it would unnecessarily suppress the use of many routes. These solutions cannot distinguish between route flaps caused by the failure or appearance of links and route oscillations. The solution in [15] also faced this problem. Ee et al. [16] used precedence metric to detect and avoid oscillations. They tagged each route with precedence metric at the top of the normal BGP decision process and kept some routes withdrawn to lower the precedence of conflicting routes. However, this metric affected the ASes outside the disputes. ASes that prefer routes through ASes in disputes may lose autonomy.

In this paper, we propose a run-time solution that operates with small overhead, guarantees safe convergence and preserves policy freedom as much as possible. Each AS maintains route changing information to determine if it may be involved in a local dispute. In this case, it advertises a token with the route. Neighboring ASes decide whether or not to forward the token according to local policies. When it retrieves the token, there is a dispute in network, and dynamically suppresses the route for convergence and launches another token. If the dispute disappears, it cannot receive the token and release the route.

The rest of this paper is organized as follows. Section 2 presents an overview of inter-domain routing and discusses the dispute wheels problem. Section 3 defines the dispute digraph tool and proves its relation with dispute wheels. The detail of preventing dispute wheels is presented in Sections 4 and 5. We discuss the benefits in Section 6. We evaluate the solution in Section 7. Our conclusions are presented in Section 8.

2. Dispute wheels problem

The Internet is connected by thousands of distinct domains, called ASes. Routing is administratively divided into intra-domain and inter-domain. Intra-domain routing, such as Routing Information Protocol (RIP) [8] and Open Shortest Path First (OSPF) [9], are generally performance oriented. Inter-domain routing is generally policy-oriented, allowing policy-base metrics to override distance-base metrics. BGP enables network operators to flexibly express local routing policies for various business, economic, security and performance purposes, which is also the reason for the success of BGP in the past decades. However, without constraining routing policies, BGP may oscillate forever, failing to converge to a stable assignment.

Varadhan et al. first discussed persistent oscillation in BGP [2]. Because ASes tend to keep routing policies private without coordinating with other ASes, these anomalies occur without any misconfiguration and are difficult to diagnose.We describe the network as an undirected graph $G = (V, E)$, where $V = \{0, 1, 2, \dots, n\}$ is a set of nodes with each node $v \in V$ corresponding to one AS, and E is the set of edges with each edge $\{u, v\} \in E$ corresponding to a BGP peer session between ASes u and v, meaning that these ASes have physical connections and share route advertisements. A path P in G is a sequence of nodes $v_0, v_1, v_2, \dots, v_k$, $k \geqslant 0$, and for each i, $0 \leqslant i \leqslant k$, $\{v_i, v_{i+1}\} \in E$. If P traverses v, $v \in P$. If there is no single node in a path, the path is an empty path ε. If P and Q are not empty paths and the first node in Q is the same as the last node in P, then P and Q can concatenate one path PQ. Because every route is independently computed for each destination, without loss of generality, we assume that every node v wishes to establish a path to the single destination node $d \in$ V. The internal BGP (iBGP), MED attribute and other details are ignored.

The BGP routing behavior can be abstracted as: nodes receive route updates from their neighbors, indicating which destinations are reachable and by what routes. Nodes apply import policies to transform incoming route updates from neighbors. A node chooses a best route based on its local policies for each destination and then sends the best route to its neighbors according to export policies. If one node's current selected route to a given destination has changed; it sends an advertisement to its neighbors.We define the following convergence properties. Each AS $v \in V$ has a path assignment $\pi(v)$.

Consistency: Paths form a forwarding tree to the destination; if $\pi(v) = vuP$, $\pi(u) = uP$.

Stability: Each node vs path assignment $\pi(v)$ is the "best" choice based on the neighbors' choices, where "best" is determined by node vs routing choices; if $\pi(v) = v\pi(u)$, no other path $v\pi(w)$ is more preferred than $\pi(v)$ at v.

Safety: Under any initial state and any sequence of route updates, the network converges to a consistent and stable assignment.

Example 1. Here we show a very simple policy-dispute example [2,3]. In Fig. 1, ASes a, b, and c each prefer indirect paths through their neighbor AS in the count-clockwise direction over direct paths to the destination AS d. Each AS has a vertical list of available paths, the highest ranked path is at the top and the lowest ranked path is at the bottom. All three-hop paths are filtered. This configuration will oscillate forever. Fig. 2 shows the oscillation.

(i) In the initial state, a, b, and c, choose paths abd, bd, and cd respectively.

(ii) When c sends its choice cd–b, upon learning a higher ranked route b changes from its current route bd to the higher ranked route bcd. But, this forces a to change to a lower ranked route ad, because the higher ranked path abd no longer exist.

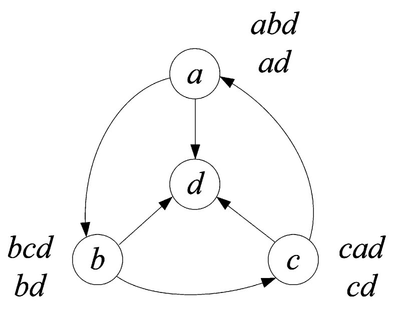

Fig. 1. A simple policy-dispute example.

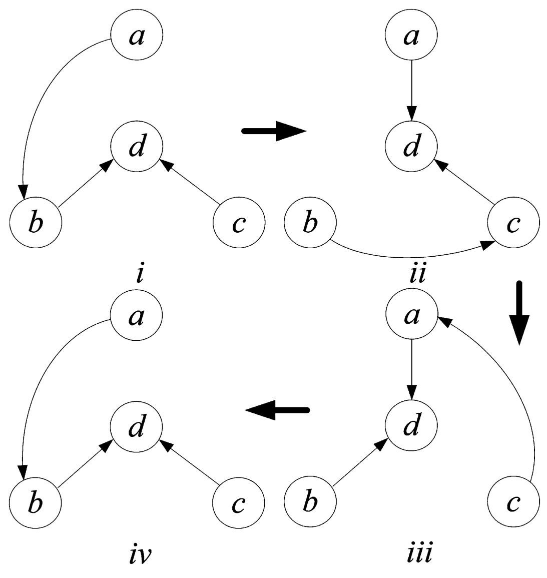

Fig. 2. A simple example of policy-dispute oscillation.

(iii) *a* then notices its choice *ad*–*c*. Similarly, *c* changes to *cad*, *b* changes to *bd*.
(iv) Finally, *b* notices its choice *bd*–*a*. And *a* reverts to *abd*, *c* reverts to *cd*. The system returns back to initial state, the sequence of route updates repeats.

That is, *a*, *b*, and *c* oscillate forever between two route choices, the system cannot reach a stable state.

2.1. Dispute wheel

Real networks have thousands of nodes (ASes), making it impractical to calculate all possible initial states and route update sequences. Griffin et al. characterized such oscillation using a smaller structure, called a dispute wheel, as shown in Fig. 3 [3]. A dispute wheel captures the interactions between the routing policies involved in an oscillation. More importantly, Griffin et al. gave us a safe assignment condition. Formally, it has the following definition.

A dispute wheel is a set of distinct nodes $u_0, u_1, \dots, u_m$, called pivots, such that the following conditions are satisfied:

1. Each pivot u_i exits two paths R_i and Q_i.
2. R_i is a spoke path from u_i to the destination d; Q_i is a rim path from u_i to the next pivot u_{i+1}.

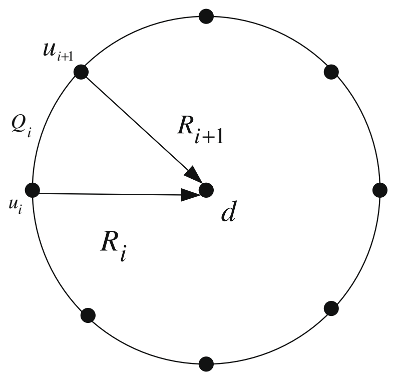

Fig. 3. Illustration of a dispute wheel.

3. Each pivot prefers the rim path $u_iQ_iu_{i+1}R_{i+1}d$ over the spoke path u_iR_id.

Note that all subscripts are to be interpreted as modulo *m* and *d* are the fixed destination nodes. We refer to non-pivot nodes along the rim paths Q_i as rim nodes.

Any policy-induced route oscillation can be characterized by a dispute wheel [3]. If a routing system does not have a dispute wheel, it is safe. However, the absence of dispute wheels is a sufficient, but not necessary, condition. The presence of a dispute wheel does not guarantee an oscillation.

3. Dispute digraphs tool

Dispute wheels are a graph constructed from the AS graph, using the same nodes and edges. It is difficult to find whether a dispute wheel exists. Currently, there is no convenient way to construct a dispute wheel. We use path relations in the AS graph to define another tool for characterizing policy disputes, called *dispute digraphs tool*. We can more easily generate a dispute wheel with dispute digraphs. We also prove that a dispute–digraph cycle corresponds to a dispute wheel. No dispute–digraph cycle is a sufficient condition to ensure routing system safety.

In AS graph *G*, we define two relations, → and → . Assume that *d* is the fixed destination node, each node $v \epsilon V$.

Subpath: $P_1 \rightarrow P_2 \exists$ a path *Q* such that $QP_1 = P_2$; P_2 has all nodes in P_1, and P_1, P_2 end at the same node.

Linear Selection: $P_1 \rightarrow P_2$.

P_1, P_2 are two different paths from the same node *v* to the destination *d*; if and only if node *v* prefers P_1 over P_2.

For any AS graph *G*, we construct a directed graph, $D = (V_D, E_D)$. The nodes V_D are the paths in an AS graph *G*. The directed edge (P_1, P_2) in E_D holds $P_1 \rightarrow P_2$ or $P_1 \rightarrow P_2$ relationship, and two relationships alternately connect. That is, the dispute digraph corresponding to the AS graph.

The definition of the two relations first appeared in [11]. They used the transitive closure of the relation → ∪ → to determinate whether two Stable Path Problem instances are equivalent. But they did not give the formal proof about the relationship between the dispute wheel and dispute digraph. In [20,21], they also defined the dispute digraph, but they defined linear selection or dispute arc on two paths that start at different nodes. Under this kind of definition, two path relations both start different disputes.

We argue that one AS cannot just use this kind of relation to determinate whether it involves a dispute only based on local information.

In dispute digraphs, each node presents a single network route rather than an AS. No node appears twice. However, one node may appear more than once in a dispute wheel. So dispute digraphs are simpler than dispute wheels.

The two relations capture the connections of route path and transition choices. Intuitively, the route paths in a dispute wheel are consistent with a cycle in a dispute graph. We therefore have the following theorem.

Theorem 1. *A routing system has a dispute wheel if and only if it has a cycle in its dispute digraph.*

Proof. Assume that the routing system has a dispute wheel. Since the rim paths of the dispute wheel form a cycle, the dispute digraph also has a cycle. This is because the spoke paths and the rim paths both belong to one of the two relations as follows. Each pivot u_i prefers the path $u_iQ_iu_{i+1}R_{i+1}d$ over u_iR_id in a dispute wheel, i.e., $u_iQ_iu_{i+1}R_{i+1}d \rightarrow u_iR_id$. That is, the linear selection relation holds. Furthermore, path u_iR_id is the subpath of $u_{i-1}Q_{i-1}u_iR_id$, i.e., $u_iR_id \rightarrow u_{i-1}Q_{i-1}u_iR_id$. All the relations in a dispute wheel imply: