Leveraging BGP Dynamics to Reverse-Engineer Routing Policies

Sridhar Machiraju
Randy H. Katz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-61
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-61.html

May 16, 2006

Leveraging BGP Dynamics to Reverse-Engineer Routing Policies

Sridhar Machiraju and Randy H. Katz
UC Berkeley
{machi,randy}@eecs.berkeley.edu

Abstract

Inter-domain routing policies are an important component of today's routing infrastructure. Knowledge about these policies can be used for better traffic engineering, detecting misconfiguration, preventing policy conflicts and also, in understanding Internet routing. However, many domains consider their policies proprietary and rarely reveal them. Hence, techniques that reverse-engineer routing policies are very useful. Existing approaches infer routing policies primarily by analyzing routing tables. In this paper, we describe how inter-domain routing dynamics, primarily the BGP convergence process, can be leveraged to infer route selection policies of domains. We discuss the results of using our proposed technique with archived BGP protocol data. We also describe the applicability of our proposed technique in achieving better traffic engineering and detecting policy conflicts.

1 Introduction

The Internet consists of multiple domains, also referred to as Autonomous Systems (ASs). Each AS connects with one or more ASs via one or more links. Routing in this multi-domain infrastructure is accomplished by a two-tier process. Interior Gateway Protocols (IGPs) are used by routers within an AS to exchange routing-related information. The inter-domain routing protocol, Border Gateway Protocol (BGP) [9], is used to exchange global routing information between ASs.

An AS uses BGP by establishing at least one BGP session between one of its routers and every neighboring AS. We refer to the two routers at the end of a BGP session as *BGP Peers*. BGP Peers exchange routing information in the form of reachable destination address prefixes and the AS-level path used to reach each of these prefixes. Since the whole path to reach a destination prefix is exchanged, BGP is a path-vector protocol. An important feature of BGP is that policies control many aspects of BGP. For instance, operators may specify import policies that would prevent certain paths from being used. Similarly, export policies are used to determine whether a path should be advertised to neighboring ASs or not. Routing policies are also used by ASs to choose the "best" path to reach a destination prefix if the destination can be reached via more than one neighbor. Specifically, BGP can be configured to assign *local preference* attributes to paths. Among all paths to a destination prefix, the path with the highest local preference is chosen by BGP. Tie-breaking rules are used to choose from paths that have the same local preference.

Routing policies affect inter-domain routing in many ways. Knowledge of these policies is useful for many reasons. First, such knowledge improves our understanding of Internet routing. Second, network operators can perform much better inter-domain traffic engineering [8]. Third, a global view of routing policies can be used to detect divergence-causing policy conflicts [7] and other misconfiguration. However, ASs consider routing policies proprietary and rarely reveal them. Hence, techniques that reverse-engineer routing policies are very useful.

In the past, many aspects of inter-domain routing have been reverse-engineered. For instance, techniques to determine the commercial relationships (which influence routing policy) between ASs have been proposed in [5, 10]. Wang et al. [11] study the import and export policies used by ASs. To our knowledge, all existing techniques use steady-state BGP, i.e., routing tables.

In this paper, we argue that BGP routing dynamics can be effectively leveraged to infer routing policies. In particular, we describe how local preferences and other route selection policies affect the *BGP convergence process*, the set of BGP messages triggered by a change in the network before all ASs converge to a new stable set of routes. Based on this description, we propose an algorithm that can extract information regarding the local preferences by observing BGP convergence processes. We use archived BGP messages from Routeviews [1] to apply our proposed algorithm and describe our results. We also discuss how our algorithm can be used pro-actively for better traffic engineering and preventing policy conflicts.

This paper is organized as follows. In section 2, we provide background material including an overview of BGP, routing policies and motivate the need to reverse-engineer routing policies. In section 3, we develop an algorithm that leverages the

BGP convergence process to infer useful information about the local preferences of ASs. In section 4, we present the results of applying our algorithm to archived BGP messages. We conclude in section 5 with a description of how our algorithm can be used pro-actively to improve routing.

2 Background

In this section, we provide background material related to our work. First, we provide an overview of BGP, the inter-domain routing protocol used in the Internet. Next, we describe the nature of BGP policies. Then, we motivate the need to reverse-engineer routing policies. Due to space constraints, we do not have a comprehensive description of all related work.

2.1 BGP Overview

The Internet consists of multiple domains or ASs. Typically, each AS belongs to a different administrative entity and has links connecting it to one or more ASs. Neighboring ASs may have more than one link between them. BGP sessions established between neighboring ASs are used to disseminate reachability information regarding destination IP address prefixes (blocks of IP addresses). The dissemination is done as follows. One or more of an ASs' neighbors may advertise reachability to a destination prefix. Each AS selects one of these neighbors to reach the prefix and further advertises this to its neighbors. To prevent routing loops, reachability advertisements also include that AS-level path used. In practice, multiple routers in an AS may speak BGP. To ensure consistency between all these routers, *internal BGP (iBGP)* sessions are established between routers of the same AS. BGP sessions between neighbors of different ASs are referred to as *external BGP (eBGP)* sessions.

One of main characteristics of BGP is that it is policy-aware. At a broad level, BGP decision-making consists of three stages, each of which can be affected by policy. First, *import policy* specifies that certain neighboring ASs should not be used to reach a destination prefix. It also assigns various attributes to paths that may be used in the later stages. After applying import policies, a destination prefix may be reachable through multiple paths. The second-stage, referred to as the *BGP path-selection process* chooses one of these paths as the path to reach the corresponding destination prefix. The path-selection process is as follows.

- Select the path with the highest *local preference (locpref)* value. This is an attribute that is specified by the AS operator.
- Select the path with the smallest number of ASs.
- Select a path learned from the Interior Gateway Protocol (IGP) over one learned from BGP.
- Since neighboring ASs may peer at multiple points, there may be multiple paths with the same next-hop AS. Select a path with smallest MED value, another path attribute.
- Select a route learned from an eBGP session over a route learned from an iBGP session.
- Select a route with the smallest IGP metric to the egress router.
- Select the route advertised by the router with the smallest ID.

In the third stage, *export policy* determines whether or not the selected path should be advertised to a neighbor AS.

2.2 Commercial Relationships

As prior work [5, 10, 11] has shown, commercial relationships between ASs are predominantly of two types:

- **Customer-Provider:** The provider AS provides connectivity to the customer AS.
- **Peer-to-peer:** The ASs provide connectivity between their respective customers.

It is known [5] that other kinds of relationships exist but are rare. These include sibling, backup relationships etc. These commercial relationships are important because they influence routing policies to a great extent. Two well-known *policy rules* are that (1) Routes from a peer/provider should not be exported to other peers/providers. (2) Routes through a customer are preferred over routes through peers/providers, i.e., the local preferences of customers are higher than peers and providers. These rules are used in [5, 10] to infer the relationships between ASs using the routes that they use. That these rules are rarely false was shown by Wang et al. [11]. They also showed that peers are often preferred over providers. They primarily used routing tables of different ASs to perform their analysis.

2.3 Why Reverse-Engineer?

As described above, prior work focuses on inferring the commercial relationships between ASs. The advantage of this approach is that it provides insights into the structure of the Internet. The disadvantage is that, by simplifying the relationships between ASs into a customer-provider, peer-to-peer etc., certain crucial details are missed which have to be inferred by other techniques. We now describe two cases where this is true - traffic engineering and detecting policy conflicts.

A well-known limitation of BGP is that it does allow an AS to easily control the flow of incoming traffic (for load balancing, backup etc.). Such inter-domain traffic engineering is usually performed [8] by using a clever BGP configuration. For instance, the operator of a multi-homed AS A might want to ensure that one of the provider links is used only as a backup. To ensure that the path using this backup link is not used much under normal circumstances, the operator would prepend the path of such BGP advertisements with A multiple times. This ensures that the length of the backup path at any AS would be larger than the primary path. Hence, if the local preferences are equal, any AS would choose the primary path over the backup. In practice, unequal local preferences could cause this the backup route to be preferred over the primary. Thus, knowledge of specific local preferences (for the prefixes of interest to A) would be invaluable to an operator in debugging such problems and devising ways to circumvent them.

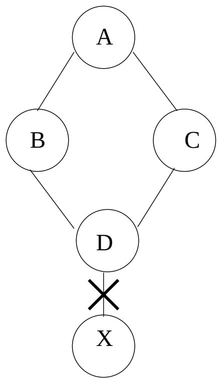
We now describe the second case where the inference of commercial relationships does not suffice. Griffin et al. [7] showed that routing policies can easily lead to divergent routing. Gao et al. [6] showed that if all AS relationships are peer-to-peer/customer-provider, ASs follow the policy rules described earlier *and* the customer-provider graph is acyclic, then such policy-induced divergence will not happen. It is generally assumed that these rules are strictly followed for every prefix. However, violation of these rules, by misconfiguration or design, could significantly affect the robustness of the Internet. Creating a global view of routing policies is an important step towards ensuring that policy-induced divergence does not occur. Voluntary disclosure of policies in public *whois* databases etc. have been ineffective. Hence, techniques that infer per-prefix local preferences are a first step towards preventing policy-induced conflicts.

3 Inferring Local Preferences

In this section, we motivate and explain our proposed algorithm to infer the local preferences used by ASs. First, we provide the basic idea behind our proposed algorithm. Then, we develop our algorithm and discuss its pros and cons. Finally, we discuss potential uses of our technique in the context of better traffic engineering and preventing policy conflicts.

3.1 Basic Principle

Consider an AS A whose neighbors B and C advertise routes $R_1 = BDX$ and $R_2 = CDX$ to a destination prefix belonging to AS X. This is shown in Figure 1. Assume that A assigns a higher local preference to R_1 than R_2. Then, according to the 7-step BGP path-selection process discussed in section 2, A would select R_1. Furthermore, AR_1 is advertised to its neighbors, if allowed by its export policy. Consider what happens if the link between D and X failed. Both routes R_1 and R_2 would be withdrawn by B and C respectively in some order. If R_1 is withdrawn before R_2, A would execute the BGP path selection process again. Since it has only 1 advertised route R_2 to the destination, A would select it. Furthermore, if its export policy does not disallow it, AR_2 would also be advertised to other neighbors. When R_2 is also withdrawn by C, A would remove R_2 and also withdraw AR_2 from its neighbors. Figure 1 lists this sequence of actions. Under an export policy that allows both AR_1 and AR_2 to be advertised to neighbors, a neighbor of AS A would receive the advertisements shown in bold. In general,



Sample Convergence Process
0.A advertises ABDX
1.D−X link fails
2.D withdraws DX from B and C
3.B withdraws BDX
4.A replaces ABDX with ACDX
5.C withdraws CDX
6.A withdraws ACDX

Figure 1: An illustration of a sample BGP convergence process when a prefix of X becomes unreachable.

Observation ObsDec: An AS A advertises paths (to its neighbors) in order of decreasing preference if no new paths are advertised to A by its neighbors and A's policy does not change.
Observation ObsInc: Conversely, an AS A advertises paths (to its neighbors) in order of increasing preference if none of its neighbors withdraws any paths during this time and A's policy does not change.

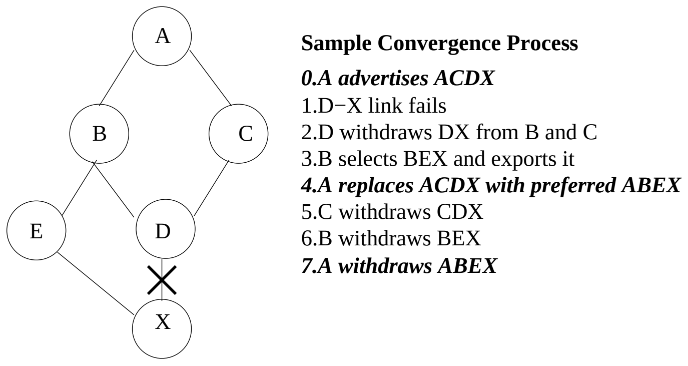
Thus, observing the AS path advertisements from AS A when the above conditions are met provides us with information on the local preferences of A for the corresponding destination prefix. Specifically, if we know that A prefers path R_1 over R_2, we can make one of the following inferences.

- **Positive-inference:** If R_1 is longer than R_2, then the local preference of R_1 *must* be greater than the local preference of R_2. This is because only the local preference can cause a longer path to be preferred over a shorter path.
- **Negative-inference:** If R_1 is not longer than R_2, then the local preference of R_1 is *not lesser* that that of R_2. If this was not true, then R_2 would have been preferred over R_1.

3.2 Proposed Algorithm

Clearly, if we had control over all the neighboring ASs of an AS A, we could easily determine whether or not the conditions of **ObsDec**/**ObsInc** are met. In practice, researchers and network operators can usually access the BGP updates generated by one or a few ASs only. Assume that we have access to BGP updates from A. We relax this assumption later in this section. Hence, to apply **ObsDec** and **ObsInc**, we need to identify times during which **ObsDec** can be applied. To do so, we use an approach similar to that used in [3, 4]. We classify updates into events; a new event is started only when the previous update was more than t_{event} time units in the past. The intuition is that, the time between events is large enough that routes for that prefix from A were stable from the end of an event to the beginning of the next. Only during an event are newer paths to reach the destination advertised to A or existing paths withdrawn. Two types of events are important to us: (1) A *PrefixDown* event is an event that results in the destination becoming unreachable from A. (2) A *PrefixUp* event is one that results in the establishment of a stable route to a previously unreachable prefix.

Though a *PrefixDown* event results in all existing routes to be withdrawn, this does not imply that the no new paths were added. In fact, during any event, BGP convergence process could result in ASs advertising short-lived paths. This is illustrated in Figure 2 where we add a new AS E connected to B and X. B might prefer the route to X through E which is prohibited (by its export policy) from being advertised to A. However, if E withdraws before D and if the route through D can be exported to A, then B would advertise a new route to A. This is a type of induced update [4] that causes incorrect inference of the location of routing instabilities [4]. Many type of induced updates are unorthodox [4] but can happen in practice.



Sample Convergence Process

0.A advertises ACDX
1.D−X link fails
2.D withdraws DX from B and C
3.B selects BEX and exports it
4.A replaces ACDX with preferred ABEX
5.C withdraws CDX
6.B withdraws BEX
7.A withdraws ABEX

Figure 2: An illustration of a BGP convergence process when AS A announces a more preferred path after a less preferred path.

BGP also uses a lot of timers for stability. For instance, the MRAI timer in BGP limits the rate of updates but not explicit withdrawal messages. Timers are also used to limit the rate of advertising routes. All of these cause temporary path exploration to occur some time after the start of the event. This is the reason why some *PrefixDown* events contain withdrawals followed by path advertisements. Hence, we restrict **ObjDec** inference to the initial stable path and the path advertised by the first update of a *PrefixDown* event only. This inference is correct as long as:

- The initial stable path is withdrawn before any induced updates. In this paper, we assume that induced updates do not occur before the first update from A. However, we are currently exploring strategies that would actually detect induced updates. For instance, in figure 2, $ABDX$ is likely to be an induced update if the path BDX was not advertised by B to any of its neighbors.
- The initial stable path is not implicitly withdrawn, i.e., the existing stable path hop is not replaced with another path. If this happens, and the replacement is advertised by A, then we notice the same next-hop and cannot infer anything about A. As we show later in this section, we use such an event to infer the local preference of an appropriate downstream AS.

Similar arguments hold good for *PrefixUp* events too except that we would use **ObjInc** only with the last short-lived update and the final stable path of the event.

```
performInference(BGPUpdateStream updates)
    $eventList = \textbf{groupIntoEvents}(updates, event_t)$;
    foreach $event$ in $eventList$
      if $\textbf{isPrefixDown}(event)$
        infer($\textbf{initialStablePath}(event)$, $\textbf{firstUpdate}(event)$);
      else if $\textbf{isPrefixUp}(event)$
        infer($\textbf{finalStablePath}(event)$, $\textbf{lastUpdate}(event)$);
      else
        $doNothing(event)$;
```

```
infer(ASPath P1, ASPath P2)
    $lastAS = \textbf{lastASInPath}(P1)$;
    $infAS = \textbf{lastCommonAS}(P1, P2)$;
    $morePref = \textbf{pathBefore}(P1, infAS)$;
    $lessPref = \textbf{pathBefore}(P2, infAS)$;
    if $\textbf{length}(morePref) > \textbf{length}(lessPref)$
      $locpref_{infAS}(morePref) > locpref_{infAS}(lessPref)$;
    else
      $locpref_{infAS}(morePref) \geq locpref_{infAS}(lessPref)$;
```

Figure 3: Pseudo-code for Inferring Local Preferences: Choose the initial (final) stable path and the first (last) update when a prefix becomes unavailable (available). Infer local preference assuming that the maximal uncommon portions of these paths are in decreasing preference for the last common AS of these paths.

We make three points. First, multiple concurrent events could cause our algorithm to fail. For instance, an overlapping link restoration and link failure may be classified as a *PrefixDown* event even though new paths may be advertised to A. Second, simultaneous changes to routing policy also might cause similar effects. Third, prior work [4] on pinpointing the cause of routing updates uses a small value of t_{event} to classify BGP updates into events. They do this since their goal is only to determine the cause of instabilities. We use a much larger value of 1 hour so that updates delayed by route flap damping are not classified as new events.

So far, we assumed that we had access to the advertisements from the AS (A in our examples above) whose local preferences we infer. In practice, we may be able to access advertisements (only) from ASs other than A. Consider Figure 1 again. Assume that A is connected to an AS V whose updates we have access to. Updates from V may reflect the convergence process of A, i.e., V advertises $VABX$, $VACX$ followed by a withdrawal. In such a case, we determine that $VABX$ and $VACX$ differ after AS A which is the *last common AS*. Hence, this implies that A advertised ABX followed by ACX and we can apply **ObsDec**. However, if timing issues etc. caused V to receive a withdraw before it advertised $VACX$, then we do not see the complete convergence process. In general, the farther V is from A, the longer into the convergence process does the first update (of that event) from A arrive. This increases the risk of induced updates causing incorrect inference. This is a natural tradeoff. We present the pseudocode of the complete algorithm in Figure 3. Note that we do not infer the absolute local preference values. They do not matter because the BGP path-selection process uses only the relative ordering.

3.3 Active Inference

We have described how to infer local preferences by passively analyzing BGP updates. Hence, very little will be known about the policies regarding a very stable prefix. However, the operator of an AS can always pro-actively withdraw/announce a prefix she owns to apply our inference algorithm. To minimize disruption to normal operations, this could be scheduled during a time of minimal activity. The advantage of this is that the owner of each prefix can infer local preferences of prefixes owned by her. In particular, a multi-homed AS could withdraw paths from all of its providers and use updates of the resulting convergence process to infer local preferences at distant ASs. Moreover, some of these advertisements can be prepended with redundant AS numbers (see section 2.3) to ensure that more *Positive-Inferences* are made. A disadvantage is that the operator may not be able to infer the local preferences of all ASs since the convergence process is a function of timings of various BGP sessions, topology etc Nevertheless, we believe that such information can be of tremendous use to operators in understanding the best way of achieving their traffic engineering goals. Such active inference can also be used by operators to ensure that policy-inflicted divergence does not occur. A way to do this would be to check that the inferred local preferences of ASs follow well-known safety rules (see [6]).

4 Results

In this section, we present the results of using our inference algorithm with archived BGP data. First, we explain how we generated the results and present some statistics related to the data used. Then, we present evidence to support some of our design decisions. A thorough validation was not possible because it is hard to obtain actual routing policies of ISPs. Finally, we make some observations regarding policies used in the Internet and the possibility of non-conforming routing policies.

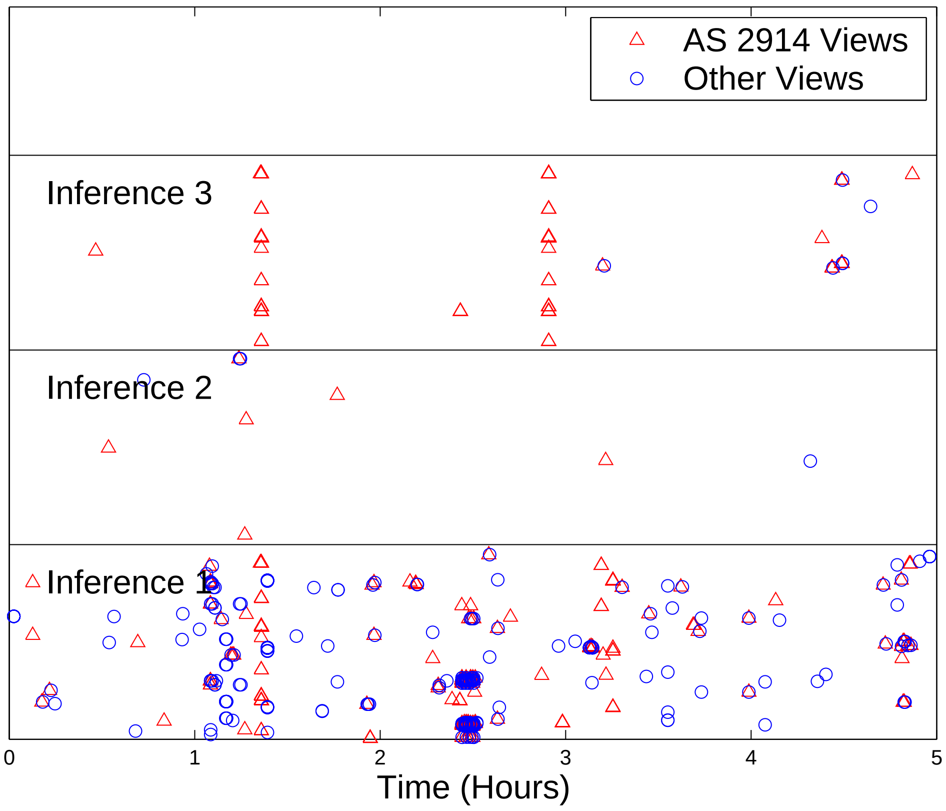


Figure 4: Plot demonstrating the consistency of inferences regarding the local preferences of AS 2914 using BGP updates from AS 2914 only and all other ASs.

4.1 Introductory Remarks

We used archives of BGP updates from the University of Oregon's Routeviews project [1]. These updates were obtained from peering sessions with about 50 peers in ASs that include tier-1 ISPs and tier-2 ISPs. The updates from any such session are referred to as a *view*. We used updates received from January 2003 to January 2005. We used 3600 seconds as t_{event} to classify updates into events. Then, we chose *PrefixUp* and *PrefixDown* events from every view and applied the algorithm described in Figure 3 to make *Positive*- and *Negative*- inferences.

To give the reader an idea of how much information can be inferred, we quote a few basic statistics. For January 2005, the recent-most month that we analyzed, we inferred local preferences of more than 40000 IP address prefixes using 1.188 million *PrefixDown* and *PrefixUp* events. The actual number of such events was larger; those which consisted of only a withdrawal and a stable path could not be used by our inference algorithm. The events used in January 2005 were inferred using updates from 39 BGP peering sessions to 33 ASs. Of these, about 740000 were *PrefixDown* events and about 450000 were *PrefixUp* events. The asymmetry between the two kinds of events was observed in every month even though the number of prefixes in BGP routing tables did not change appreciably during the month. This is likely caused because route advertisements and explicit withdrawals are treated differently [9]. The time between consecutive route advertisements for a prefix is required to be not more than an interval referred to as the *MRAI* (Minimum Route Advertisement Interval). This does not apply to explicit withdraw messages. When a prefix goes down, withdraw messages are likely to propagate faster. Hence, the convergence process is likely to cause more messages and *PrefixDown* events provide us with more information than *PrefixUp* events. We confirmed this by observing the number of messages in *PrefixDown* and *PrefixUp* events. For instance, of all events that related to prefixes of the type $64.x.y.z/a$, more than $2/3$ of the *PrefixUp* events consisted of a single route advertisement (of the final stable path). In contrast, only $1/4$ of the *PrefixDown* events consisted of a single withdrawal (of the initial stable path). Of the 1.188 million inferences that we made for January 2005, 71164 (about 6%) were *Positive-Inferences*. This varied from 6% to 8% for each of the 25 months that we analyzed.

4.2 Validation

Validating our inferences is hard because actual routing policies are rarely available. *Whois* registries are known to have policies of various ASs in the Routing Policy Specification Language (RPSL) [2]. These have been used in the past [11] to study policies. We could not perform a detailed comparison of all our inferences with this database because they are often out-of-data and incomplete. Also, local preferences have to be specified as $pref$ values in which the lower value is more preferred; This is the opposite of local preferences. Upon inspecting many of these policies, we are inclined to believe that a significant number of them have specified preferences wrongly. For instance, AS 286 uses the same numbers to describe local preferences in the comments and the $pref$ values. AS 286's configuration also assigns tier-1 ISPs the smallest $pref$ value which is unlikely.

For completeness' sake, we provide examples using AS 5511 which has a detailed up-to-data database. For destination prefix $200.82.196.0/23$, we observed the AS path $< 5511, 6505(4\text{ times}), 21826 >$ after $< 5511, 3549, 21826 >$ in a *PrefixUp* event. The registered policy does indeed say that the $pref$ of AS 6505 is 10 and 25 for AS 3549. Similarly, for prefix $193.22.84.0/24$ the AS path $< 5511, 13237, 29416, 20833(7), 31593 >$ was followed by $< 5511, 3356, 8437, 6849, 20833, 31593 >$ in a *PrefixDown* event. Again, the registered $pref$ values were consistent with the resulting inference that $AS13237$ is preferred over $AS3356$. In a third case, for prefix $216.243.234.0/23$, $< 5511, 1239, 2914, 174, 10970 >$ was followed by $< 5511, 6830, 174, 1097 >$. In this case, the registered policy was specified that AS 6830 had a higher preference only for IP addresses in $AS - AORTA$. We could not verify whether the destination prefix falls into $AS - AORTA$ or not. We did not do perform any more analysis using the registered policies because of the difficulties we encountered with the policies and the address groups. We are currently exploring ways to overcome these problems.

Consistency Across Views

The dataset we use consists of BGP updates from multiple views. During a *PrefixDown* event, we choose the initial stable path and the first update of the convergence process to use with our inference algorithm. As discussed in section 3, different views might see different paths. This could cause, for instance, the first update in a *PrefixDown* event to be different even though the last common AS is the same. To check the prevalence of this scenario, we plot figure 4. Each point refers to an inference for some prefix in January 2005. We assign each "new" inferred local preference ordering (for a prefix) a new number. We also assign a number to each unique prefix. The x-axis value is the beginning time of the event that was used for a particular inference. The y-axis value of a point from the bottom of its strip (labeled "Inference i") is the unique number of the prefix. Thus, if two different views made different inferences, we would observe two points in different strips with the same x-axis value and the same y-axis value relative to the bottom of its strip. For clarity, we plot this only for a 5-hour period. As the vertical line of red triangles show, we found that all different inferences were made by two different views of AS 2914! For instance, the inferences for $20.143.112.0/20$ were both *Negative-Inferences*, $AS1239 \geq AS701$ and $AS701 \geq AS1239$. It is likely that in reality, both 1239 and 701 may have the same local preference. IGP weights may be the actual reason behind the different behavior of the two views.

4.3 Applications

We describe a few observations we made from our inferred policies regarding traffic engineering and policy deviations.

Non-conforming Policies

We used the data from [10] to determine their inferred AS relationships. It is expected that customer routes are preferred over peer and provider routes. Also, Wang et al. [11] concluded that typically, peers are preferred over providers too. We used the AS relationships inferred in 10 instances in 2003, 2004 to detect if either of the two preference rules are being violated. We present our results in Table 1. Note that the number of policy violations does count the same policy violation seen from different views. The last column showing the number of affected prefixes (10000, on average) does indicate, however, that a significant portion of the IP address space may have some form of deviant policy. One example of a deviant policy is AS 2914 (Verio) preferring AS 3549 (Global Crossing), another tier-1 ISP, over AS 15270, a customer according to the algorithm of [10]. As future work, we intend to study if these possible policy conflicts had ever resulted in divergent routing.

Miscellaneous Observations

Our technique also showed many examples where traffic engineering by prepending redundant AS paths does not work. Consider the example in 4.2. AS 5511 prefers to use AS 6505 over AS 3549 and chooses a path with three redundant 6505's in the AS path. It is possible that in this case, traffic from AS 5511 did not constitute a large amount of traffic and hence, not a problem. But in cases where it matters, operators can use our technique pro-actively to infer such anomalous preferences and devise a well-informed traffic engineering plan. As with any network measurement study, we also saw a

Table 1: Number of Policy Violations and Abnormal Policies

Month	Prefer Provider/Peer over Customer	Prefer Provider over Peer	Number of Affected Prefixes
Jan 2003	17409	8831	10294
June 2003	20613	2426	8936
July 2003	26528	25497	15800
August 2003	25019	4567	8413
September 2003	17081	1494	6708
October 2003	19415	12028	11631
November 2003	22900	1919	9075
December 2003	24950	4661	8250
January 2004	23294	6581	9577
February 2004	18995	1858	5797

significant number of pathological scenarios. A common one was convergence processes (say *PrefixUp*) in which the first update would have no path prepending which would be replaced by the same AS path with more path prepending. For instance $< 6939, 9942, 9300, 10113 >$ was replaced 28 seconds later by an AS path that had AS 9942 thrice in it.

5 Conclusion and Future Work

We developed a novel algorithm to infer local preferences of ASs using BGP updates during a convergence process. We explained when our proposed algorithm works and when it does not. Our algorithm can be put to good use to study properties of the Internet, for better inter-domain traffic engineering and to detect policy conflicts. We are currently exploring ways to validate our algorithm in a large scale via simulations and using real data. Based on these we would improve the accuracy of our algorithm.

References

[1] `http://antc.uoregon.edu/route-views/`.

[2] C. Alaettinoglu, C. Villamizar, E. Gerich, D. Kessens, D. Meyer, T. Bates, D. Karrenberg, and M. Terpstra. RFC 2622 - Routing Policy Specification Language (RPSL), June 1999.

[3] Matthew Caesar, L. Subramanian, and Randy Katz. Towards Localizing Root Causes of BGP Dynamics. Technical Report UC Berkeley UCB/CSD-04-1302, November 2003.

[4] Anja Feldmann, Olaf Maennel, Z. Morley Mao, Arthur Berger, and Bruce Maggs. Locating Internet Routing Instabilities. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2004.

[5] Lixin Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Transactions on Networking (TON)*, 9(6), December 2001.

[6] Lixin Gao and Jennifer Rexford. Stable Internet Routing without Global Coordination. In *Proceedings of ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, June 2000.

[7] Timothy G. Griffin and Gordon Wilfong. An Analysis of BGP Convergence Properties. In *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, 1999.

[8] Bruno Quoitin, Steve Uhlig, Cristel Pelsser, Louis Swinnen, and Olivier Bonaventure. Interdomain Traffic Engineering with BGP. *IEEE Communications Magazine*, 2003.

[9] Y. Rekhter and T. Li. RFC 1771 - A Border Gateway Protocol 4(BGP-4), March 1995.

[10] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proceedings of IEEE Conference on Computer Communications (INFOCOM)*, 2002.

[11] Feng Wang and Lixin Gao. On Inferring and Characterizing Internet Routing Policies. In *Proceedings of Internet Measurement Conference (IMC)*, October 2003.