BorderGuard: Detecting Cold Potatoes from Peers

Nick Feamster
MIT Computer Science & AI Lab
feamster@csail.mit.edu

Zhuoqing Morley Mao
University of Michigan
zmao@eecs.umich.edu

Jennifer Rexford
AT&T Labs–Research
jrex@research.att.com

Abstract

Internet Service Providers often establish contractual "peering" agreements, where they agree to forward traffic to each other's customers at no cost. *Consistent route advertisement at all peering points* is a common provision in these agreements, because it gives an AS the flexibility to select egress points for the traffic (e.g., performing "hot potato" routing). Verifying "consistent export" is challenging because route advertisements are exchanged at multiple peering points and may be modified by routing policies. In this paper, we propose two algorithms to detect inconsistent routes using routing and configuration data from an AS's border routers. The first algorithm requires access to all eBGP routes advertised by a peer. Because this data is often unavailable, we propose another algorithm that detects inconsistencies using readily available data. We have applied our algorithms to the routes advertised by the peers of AT&T's commercial IP backbone. Although a peer may intentionally send inconsistent advertisements to prevent its neighbor from performing hot-potato routing, we also discuss several configuration scenarios where a peer may *inadvertently* advertise inconsistent routes, despite having consistent export policies. Finally, we explain how simple modifications to the routers could make detection of inconsistent advertisements much easier than it is today.

Categories and Subject Descriptors

C.2.6 [**Computer-Communication Networks**]: Internetworking; C.4 [**Performance of Systems**]: Measurement Techniques

General Terms

Algorithms, Management, Measurement, Performance

Keywords

BGP, anomalies, peering, inconsistent advertisement

1. Introduction

Service providers in the core of the Internet connect to each other in order to reach their respective customers. Before agreeing to "peer," two service providers sign a peering agreement that outlines the terms of their relationship. These contracts typically require the Autonomous Systems (ASes) to connect in multiple geographic locations [1]; in Figure 1, ASes A and B peer in four locations spread throughout their networks. In addition to providing redundancy, the multiple connections are meant to give an AS the flexibility to select a convenient egress point for sending traffic to the other AS. Under the common practice of *hot-potato* (or *early-exit*) routing, a router selects the "closest" egress point in terms of the intradomain path costs, in order to reduce the network resources required to carry the traffic. For example, in Figure 1, router b in AS B can direct traffic through peering point 3 rather than sending traffic a long distance across the network to one of the other egress points. Similarly, router a in AS A's network can direct traffic through peering point 0. In some cases, a network operator may override hot-potato routing to balance the traffic load.

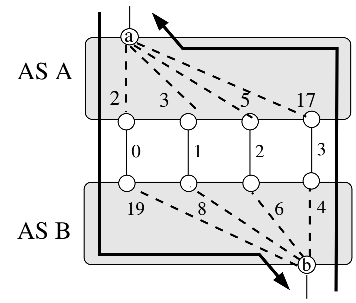


Figure 1: Hot-potato routing between peers with four peering points: Dashed lines highlight the intradomain path costs

To give operators the flexibility to select from multiple egress points, peering contracts typically require the peer to provide *consistent* routes at all interconnection points [1]. That is, an AS must make each destination reachable at every peering point via "equally good" routes. If a destination connected to router a were reachable only through peering point 0, traffic from b would have to travel over expensive long-haul links in AS B and only a short distance in AS A. In this scenario, AS A is violating its peering agreement by *forcing* AS B to do "cold potato" routing. In addition, AS A must not try to make one peering connection look less attractive to B than another (e.g., by making the AS path appear longer), unless the two ASes have agreed in advance, since this would force B to consume more resources to send traffic.

In this paper, we formulate the problem of checking the consistency of routes advertised by a peer and present a technique for detecting inconsistencies using routing and configuration data available in the receiving AS. The most closely related work is an empirical study of "path inflation" by Spring *et al.* [2], which analyzed traceroute data to infer deviations from "early exit" routing without identifying the underlying reason. In contrast, we determine whether an AS is *forced* to

1. Highest local preference
2. Lowest AS path length
3. Lowest origin type
4. Lowest MED (with same next-hop AS)
5. eBGP-learned over iBGP-learned
6. Lowest intradomain path cost to egress point
7. Lowest router ID of BGP speaker

Table 1: BGP decision process with peer-assigned attributes in bold

select a different egress point due to inconsistent route advertisements from a peer rather than *voluntarily* choosing a different egress point to satisfy its own traffic engineering goals.

An AS receives route advertisements from a peer via Border Gateway Protocol (BGP) sessions at the peering points. A BGP-speaking router sends an advertisement to notify its neighbor of a new route to the destination prefix and a withdrawal when the route is no longer available. An advertisement includes attributes, such as the list of ASes in the path, that affect the selection of the best route at each router. To be consistent, multiple routes from the same peer for the same prefix must agree in any aspects that affect the BGP decision process—AS path length, origin type, and multiple exit discriminator (MED)—as shown in bold in Table 1. Other steps in the decision process are controlled by the receiving AS. For example, a router can apply an import policy that assigns the local-preference attribute to favor one route over another, and use the intradomain path cost to select the route with the closest egress point. Although the operator can configure an import policy that resets the origin type and MED attributes to default values, the receiving AS is especially vulnerable to inconsistencies in the AS path lengths of the routes advertised by its peers.

Identifying inconsistencies should be as easy as comparing the BGP routes learned from each peer for each prefix for differences in AS path length, origin type, and MED, as discussed in Section 2. Unfortunately, acquiring a feed of *all* routes advertised by a neighboring domain is difficult in practice[1]. Instead, we consider how to detect inconsistent routes from data readily available within the local AS—an internal BGP (iBGP) feed of the "best" route for each prefix from each border router and the import policies configured on each of these routers. However, identifying inconsistencies from this data is challenging because our algorithm only has access to the "best" route for each prefix, after the routes have been manipulated by the import policies. In Section 3, we determine how much these constraints limit our ability to identify inconsistent route advertisements from peers and present an algorithm that identifies inconsistencies that force the AS to select a different egress router.

Section 4 presents the results of applying the algorithms to the routes advertised by the peers of AT&T's commercial IP backbone. We apply the first algorithm to eBGP feeds provided by one large peer and then apply the second algorithm to iBGP feeds from AT&T's border routers. Our analysis discovers many short-lived routing inconsistencies that could be explained by transient routing updates during the BGP convergence process, but we also find inconsistencies that persist for longer periods of time, suggesting either configuration mistakes or malicious behavior. In Section 5, we present several examples that illustrate how a peer might *inadvertently* advertise inconsistent routes and suggest ways the sending AS could detect potential problems in advance. Section 6 concludes the paper with a discussion of ways router vendors and network operators can make the BorderGuard problem easier to solve in the future.

[1]This would require either (1) extending today's commercial routers to provide a feed of all eBGP-learned routes, which, while definitely appealing, is not likely to happen quickly, (2) deploying packet monitors on the many high-speed links between peers to capture the BGP updates, which would be extremely expensive, or (3) asking the peer AS to provide the eBGP data feed from its own border routers, which runs the risk that the peer intentionally sends *different* information to our detection algorithm than it does to the operational routers.

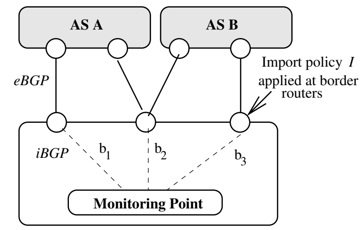


Figure 2: Monitoring inconsistent route advertisements in an AS with three peering points.

2. BorderGuard Using Direct eBGP Feeds

In this section, we formulate the BorderGuard problem and present a solution that operates on a direct feed of the eBGP-learned routes from each peer AS. Throughout the paper, our discussion focuses on a single destination prefix, since routing decisions for each prefix are independent.

A network has m peer ASes $p = 1, 2, \ldots, m$ and has n_p eBGP sessions with peer p. At any given time, the network has one (possibly null) route $r_{p,u}$ for the prefix from each peering point $u = 1, 2, \ldots, n_p$. An advertisement message on session u replaces the old value of $r_{p,u}$ with a new route; a withdrawal replaces the old value with a null route. To compare the routes, we define a function $\lambda(r_{p,u})$ that ranks a route based on the first five steps of the BGP decision process in Table 1—up to, but not including, the "hot potato" step that chooses the route with the closest egress point[2]. A lower value of $\lambda(r)$ implies a less attractive route (e.g., a route with a longer AS path length); a null route has the lowest possible value. We consider a peer as *inconsistent* if $\lambda(r_{p,u}) \neq \lambda(r_{p,v})$ for some $u, v \in [1, n_p]$.

Our algorithm applies this check to streams of eBGP data from a given peer. Upon receiving an update message on session u, the algorithm compares $\lambda(r_{p,u})$ to the values $\lambda(r_{p,v})$ for $v \in [1, n_p]$ and reports any mismatches. In the next section, we present a second algorithm that operates on streams of the "best" BGP route from each border router in the local AS.

3. BorderGuard Using Indirect iBGP Feeds

In this section, we describe the algorithm that detects inconsistent route advertisements from peers, using only data that are directly available to that AS. We first define this new problem and explain the challenges for inferring the characteristics of eBGP advertisements from iBGP data and routing policy. We state the conditions that must be true in order for this inference to be possible. We then present an algorithm that accurately determines whether a peer advertises consistent routes at all peering points as long as these conditions are satisfied.

3.1 Problem Formulation

An AS has k border routers, each of which may have zero or more sessions to each of the AS's peers. We also define a function $Routers(p)$ that returns the set of n_p routers in the AS that peer with p. Each border router i applies an import policy, I_i, to the routes that it receives via eBGP and selects a single best route b_i for a destination.

[2]Since the local-preference attribute is local to an AS, an eBGP-learned route does not have a local preference. Also, all eBGP-learned routes would receive the same treatment in step 5 in the BGP decision process. As such, only the AS path length, origin type, and MED affect the comparison between two eBGP-learned routes. Steps 1 and 5 are important in Section 3, though, to compare the "best" routes seen in different *iBGP* data feeds.

In practice, I is actually configured and applied on a per-session basis, rather than a per-router basis, but we abstract this detail to simplify notation. Each router i then distributes the route b_i to other routers in the AS via iBGP. An AS can get access to the routes $b_1, b_2, \ldots, b_k$ using iBGP sessions to a route monitor, as shown in Figure 2; many ASes already deploy such a monitor. The values of $Routers(p)$ and I_i are readily available from the router configuration data.

Access to the only the *best* routes limits an AS's ability to directly determine whether a peer advertised a route at some router (as well as the characteristics of the advertised route): the alternate routes at the border routers are not available. To determine the properties of the complete set of routes that any peer advertises, we must devise an algorithm that takes the set of best routes as input and infers properties about the routes from a peer that are not in that set.

Our inference algorithm applies the following insight: *the route b_i that router i selects must be at least as good as all other routes learned at router i, according to the first five steps of BGP decision process*. Using this insight, we can often make the following assertion: if a peer p advertises routes $r_{p,u}$ and $r_{p,v}$ to two distinct border routers and the router that learns $r_{p,u}$ selects it as the best route but the router that learns $r_{p,v}$ selects a route that is *worse* than $r_{p,u}$ according to the first five steps of the BGP decision process, then $\lambda(r_{p,u}) > \lambda(r_{p,v})$ (i.e., peer p advertised inconsistent routes). In many cases, we can make assertions about $\lambda(r_{p,v})$, *even though the monitoring point never sees* $r_{p,v}$, based on the fact that $r_{p,v}$ is missing from the set of best routes. In the next section, we describe the assumptions necessary to make this determination and also explain the cases where our algorithm cannot make accurate inferences.

3.2 Limitations on Inferring Violations

Access to only the import policies and iBGP routes from border routers presents several limitations and challenges for inferring inconsistent route advertisements.

Import policies change route attributes. Routes that the iBGP monitor sees as inconsistent may in fact be caused by the import policy locally at border routers, and routes that appear consistent *with each other* at the iBGP monitor do not ensure that a peer is sending consistent route advertisements. The monitor only observes b_i, but to detect inconsistencies in routes as sent by the peer that advertised that route ($peer(b_i)$), we must be able to determine the route that the peer actually advertised before import policy transformation (i.e., $I_i^{-1}(b_i)$). To ensure that, given b_i and $p = peer(b_i)$, the algorithm can determine the corresponding $r_{p,u}$ (i.e., the route that the peer initially sent), we require the following condition:

CONDITION 1 (INVERTIBLE IMPORT POLICY). *For all $i \in [1, k]$, I_i^{-1} is computable. That is, it is possible to recover the route that $peer(b_i)$ initially advertised by applying $I_i^{-1}(b_i)$.*

Import policies often overwrite certain route attributes (e.g., MED) on routes learned from peers, unless the AS has agreed in advance to accept them. Overwriting a route attribute is not invertible, so this operation violates Condition 1. Fortunately, the inability to determine these route attributes does not matter in a practical setting, because the AS can *force* these attributes to be consistent by overwriting the attributes in the same way (e.g., a common practice is to set MED values to 0 on all routes learned from a peer). The inference algorithm is most useful when a peer is sending inconsistent routes *in a way that import policy cannot rectify* (e.g., inconsistent AS path lengths or missing route advertisements).

Import policy can make consistent routes appear inconsistent. The algorithm must infer properties of a route that it might not see. Because we are interested in determining the properties of such a route *before* a router applies import policy, we must assume that the import policy *at a single router* treats all routes r for which $\lambda(r)$ is equal in exactly the same way. That is, the import policy at each router should not treat two routes that are consistent in a way that would make them inconsistent.

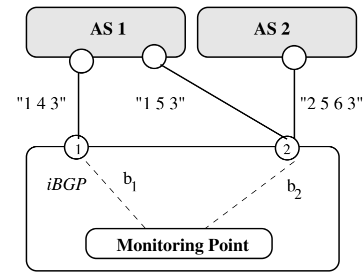


Figure 3: Example illustrating different AS paths (with the same AS path *length*) from the same peer.

CONDITION 2 (CONSISTENT TREATMENT OF CONSISTENT ROUTES.). *If $\lambda(r_{p,i}) = \lambda(r_{p,j})$ then $\lambda(I_i(r_{p,i})) = \lambda(I_i(r_{p,j}))$.*

The inference algorithm should be able to infer how route $r_{p,j}$ would have been treated by our import policy at i if that route were "consistent" with $r_{p,i}$. Otherwise, it is impossible to tell whether the AS's import policy caused the consistency violation or whether the inconsistency was caused by a peer.

Figure 3 explains how a violation of this assumption can cause ambiguity. In this case, the AS's peer p advertises a route $r_{p,1}$ with AS path "1 4 3" at one border router and a route $r_{p,2}$ with AS path "1 5 3" at a second border router; assume that all other route attributes are the same. Note that these routes are *consistent*: $\lambda(r_{p,1}) = \lambda(r_{p,2})$, because the AS path *lengths* are the same. If router 2 applied a policy that, for example, assigned a lower local preference to routes with AS path "1 5 3", then router 1 could conceivably select a route $r_{q,1}$ from another peer q. We would like to be able to say that $\lambda(r_{p,1})$ must be worse than $\lambda(r_{p,2})$ (i.e., that the routes are inconsistent), but we cannot do so: it is impossible to distinguish between the case where p sends route "1 5 3" and router 1 selects a route from q and the case where p sends a route with a longer path length to router 1 (or does not send any route).

Unfortunately, this assumption is occasionally violated. For these peers and sessions, we cannot detect inconsistent advertisements from iBGP messages alone. Nevertheless, we were still able to perform our analysis on the vast majority of peers; we discuss our analysis further in Section 4.

Inability to distinguish inconsistent routes from a missing route. Because it has direct access to eBGP messages, the algorithm in Section 2 is able to distinguish between two separate cases of inconsistent advertisements: (1) when a peer sends routes with inconsistent attributes to one or more peering points and (2) when a peer fails to send *any* route for a prefix to one or more peering points. With access to only the best routes from each router, however, the inference algorithm cannot determine whether a border router did not select a route from peer p because the route from p looked "worse" than other routes learned at that router or because p did not advertise any route at all to that router. Because the *effect* of either of these inconsistencies is the same—in either case, the AS may be forced to do "cold potato" routing—it is not crucial that the inference algorithm be able to distinguish between these two cases.

Arbitrary path selection tiebreaking. The BGP decision process may break ties between two routes r_1 and r_2 for which $\lambda(r_1) = \lambda(r_2)$ arbitrarily (e.g., based on the router ID of the router from which the route was learned or on which route was learned first). As a result, the inference algorithm may not be able to detect whether a given peer p advertised consistent routes to a destination if $\lambda(b_i) = \lambda(b_j)$ but b_i and

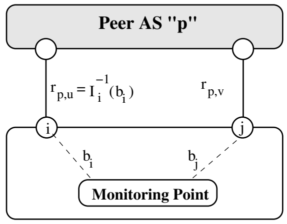


Figure 4: Applying the BorderGuard consistency assertion

b_j are learned from two different peers, p and q. For example, suppose that $\lambda(r_{p,u}) = \lambda(r_{q,u}) = \lambda(r_{p,v})$, but the tiebreaking stage at router i selects the route from peer q. In this case, the inference algorithm cannot determine whether the routes advertised by p are consistent, because $\lambda(b_i) = \lambda(b_j)$: the route from router i is not strictly worse, so a consistent $r_{p,i}$ could have existed, but it is impossible for the AS to invert this based on b_i and b_j alone.

Arbitrary tiebreaking of equally-good eBGP-learned routes at a given border router may occur frequently[3] and it prevents the inference algorithm from determining whether a peer advertised a consistent route to that router. Fortunately, this scenario can only arise if one peer advertises a route to that router *that is equally good as the other peer's advertisements*. If tiebreaking prevents inference at a given router, another equally good route must exist at that router, and "cold potato" routing will not occur anyhow: the routers in the AS that would have chosen a consistent route from that peer at that router will instead use the alternate route (rather than sending traffic to another border router), since the route they learn from that peering is as good as the consistent route would have been.

3.3 BorderGuard Consistency Assertion

Given the two assumptions from the previous section and access to both the iBGP feeds from the border routers, $(b_1 \ldots b_k)$, and the import policies, $(I_1 \ldots I_k)$, an AS can now determine whether its peer p is sending inconsistent advertisements at different peering points by testing the following assertion:

for each border router i $\quad (i = 1 \ldots k)$
for each router $j \in Routers(peer(b_i))$
$\lambda(b_j) \geq \lambda(I_j(I_i^{-1}(b_i)))$

If this condition is violated, then peer $p = peer(b_i)$ has failed to send a consistent advertisement to router j. Figure 4 explains the intuition behind this result. Ultimately, for each router i that selects a best route $r_{p,u}$, the AS must verify the following condition on routes learned from peer p:

$$\lambda(r_{p,u}) \quad = \quad \lambda(r_{p,v})$$

given only b_i and b_j. We can compute $r_{p,j}$ using Condition 1 to invert the import policy at router i on b_i:

$$\lambda(I_u^{-1}(b_i)) \quad = \quad \lambda(r_{p,v})$$

Finally, we can apply Condition 2 to obtain:

$$\lambda(I_j(I_i^{-1}(b_i))) \quad = \quad \lambda(I_j(r_{p,v}))$$

This condition must be true if $peer(b_i)$ is sending consistent advertisements (i.e., $\lambda(r_{p,u}) = \lambda(r_{p,v})$), based on our observation of b_u, *even through the monitoring point may not observe* $I_j(r_{p,v})$. If the monitoring point receives a b_j such that $\lambda(b_j)$ is strictly less than $\lambda(I_j(I_i^{-1}(b_i)))$ (i.e., the ranking of a consistent route from $peer(b_i)$ after router j applies import policy), then $\lambda(r_{p,u}) \neq \lambda(r_{p,v})$. That is, either peer p did not advertise a route to router j, or the attributes $r_{p,v}$ were strictly worse than those of $r_{p,u}$.

Testing this assertion in a live network is straightforward. Both the set of k border routers and the set of routers that peer with a peer p, $Routers(p)$, are readily available from the router configuration. I_j and I_i^{-1} can also be determined from the import policies defined in the router configurations. $peer(b)$ for any best route is also easy to compute: it is simply the first AS in the AS path attribute of the route. Starting with a table dump of the routes, the monitor can directly test the assertion for every b_i for all prefixes; in steady state, detection is more lightweight: whenever any best route b_i changes, the algorithm can simply test the assertion for that best route, rather than re-executing the check for all $(b_1 \ldots b_k)$.

4. Measurement Results

In this section, we apply the algorithms from Sections 2 and 3 to the routing and configuration data of AT&T's commercial IP backbone. We analyze both the eBGP data from one of AT&T's peers—a large tier-1 ISP—and the iBGP data from the border routers in AT&T's network (AS 7018) that connect to peers over the period of May 1-8, 2004. We verified that AT&T's import policies and peering sessions did not change during this period, and that no resets occurred on the BGP sessions to the route monitors.

4.1 Direct eBGP Feeds from One Peer

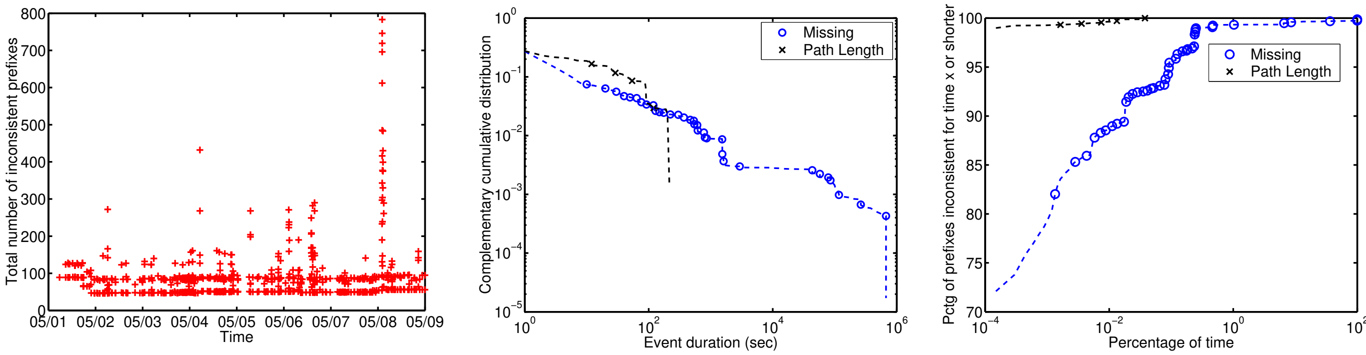
We examine eBGP feeds from an AS with about half a dozen peering points with AT&T. The data were obtained directly from the peer's routers in the same PoPs as the eBGP sessions with AT&T, and sometimes from the same router that peers with AT&T. The route monitor receiving these eBGP feeds is treated like a "customer" receiving a complete routing table. To simulate the routes received by a peer like AT&T, we use community strings to distinguish customer routes from peer routes. Route advertisements that would not be advertised to a peer are treated as withdrawals, since a BGP session with AT&T would not advertise these routes. We identify two types of inconsistency in the eBGP feeds: missing prefixes and differing AS path lengths.

Figure 5(a) shows a time series of the total number of inconsistent prefixes over the eight-day period of the study. Fewer than five prefixes have inconsistent AS path lengths at any given time, and most inconsistencies involve missing advertisements at one or more peering points. Figure 5(b) shows the complementary cumulative distribution of the duration of the inconsistencies. Most inconsistencies last less than two minutes, suggesting that they are caused by transient events such as routing protocol convergence. Figure 5(c) shows the overall duration of inconsistencies for the prefixes advertised by this peer. The graph shows that 70% of the prefixes were *never* inconsistent, and more than 97% were inconsistent less than 0.23% of the time. Still, a few inconsistencies due to missing advertisements persisted for hours or even days. A small number of prefixes were inconsistent for the entire duration of the study, perhaps due to configuration mistakes.

4.2 Indirect iBGP Feeds from Border Routers

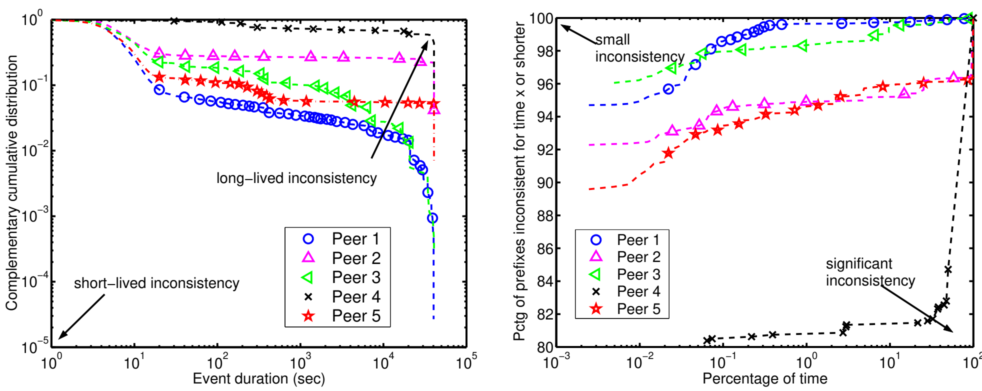
We apply our algorithm to iBGP updates received at a monitor that is configured as a route-reflector client to the AT&T border routers that connect to peers. Our analysis excluded a small number of peers where the import policies did not satisfy Condition 2 in Section 3.2. About half of the inconsistencies discovered for the peer in Section 4.1 were also discovered by the iBGP analysis; the other half of the inconsistencies were obscured by arbitrary tiebreaking at the router or because

[3] It is not uncommon for routes to the same destination from multiple peers to be "equally good" in terms of local preference, AS path length, MED, and origin type. For example, an enterprise might multihome to two or more of an AS's peers; both peers will advertise routes to that customer with the same path length. In these cases, border routers will break ties arbitrarily.



(a) Inconsistency over time (b) Inconsistent event duration distribution (c) Inconsistency across prefixes

Figure 5: eBGP data analysis of a large tier-1 ISP peering with AT&T



(a) Inconsistent event duration distribution (b) Inconsistency duration across prefixes

Figure 6: iBGP data analysis of several peers with AT&T

AT&T chose a route through a customer rather than a peer. Overall, two-thirds of AT&T's peers *never* had more than five inconsistent prefixes at time.

Our analysis in Figure 6 focuses on five of the remaining peers; Peer 3 corresponds to the same peer analyzed in Section 4.1. At any given time, at most a few hundred prefixes have inconsistent advertisements. Figure 6(a) shows the distribution of inconsistency duration for five peers, excluding the large number of events that persist for less than one second due to transient routing changes. The peers exhibit varying degrees of inconsistency. Peer 4, for instance, has significantly longer inconsistency events; in fact, this peer advertises more than 100 prefixes inconsistently for the entire duration of our study. Figure 6(b) shows the distribution of time for which the prefixes each peer advertised are inconsistent. About 20% of the prefixes advertised by Peer 4 are inconsistent more than 30% of the time. For the other peers, only 10% of prefixes advertised from any other were *ever* advertised inconsistently, and more than 90% of the prefixes were consistent at least 99% of the time.

To quantify the impact of routing inconsistencies, we analyzed the traffic destined to inconsistent prefixes using Netflow data collected from the border routers. We focused on the ten most inconsistent prefixes per peer and all prefixes that were inconsistent for the entire one-day period. The inconsistencies corresponded to less than 1% of the prefixes and less than 0.5% of the traffic leaving AT&T via the peering links. Although the inconsistencies involve small amounts of traffic, some can cause significant traffic diversions: one neighbor ISP failed to advertise 30 prefixes at five separate locations for the entire duration of the trace. In our future work, we plan to analyze the traffic directed to specific peers (such as Peer 4) in more detail and analyze longer traces.

5. How Bad Routes Can Come From Good Peers

Although a peer may intentionally violate the "consistent export" requirement, inconsistencies may be inadvertent. For example, a peer might mistakenly have minor differences in its export policies, such as filtering small subnets at one location and not another. However, applying the *same* export policy at each peering point does *not* guarantee consistent advertisements. In this section, we present three cases where an AS might not advertise consistent routes to its peer, even though the AS applies consistent export policies. Because we see neither the missing route nor the configuration of the neighboring AS, we cannot determine what caused the inconsistency (or even whether it was accidental), but there are at least three plausible explanations for unintentional inconsistencies:

Missing iBGP session: Each router in an AS selects a single best route for each prefix from the routes learned via iBGP and eBGP neighbors. In the simplest scenario, the peer AS has a "full mesh" iBGP configuration with a BGP session between each pair of routers. However, a configuration mistake may lead to a missing iBGP session, as shown by the dashed line between routers 1 and 4 in Figure 7(a). As a result, router 3 receives a BGP route to d but router 4 does not, leading the peer to advertise the prefix at one peering point but not the other.

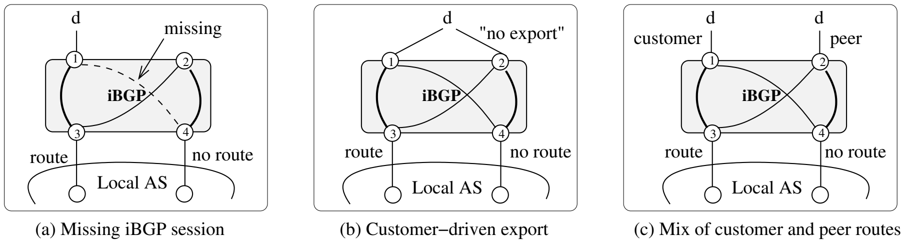


Figure 7: Peer AS configurations that lead to inconsistent route export, despite consistent export policy. Router 3 has a small intradomain path cost to router 1, and router 4 has a small intradomain path cost to router 2.

A similar configuration mistake could also cause the peer to advertise routes with different AS path lengths, if one router learns a short route and another learns a longer route.

Although it might appear that a missing iBGP session is a pathological case of misconfiguration that would be quickly caught by a network operator, it turns out that missing iBGP sessions are fairly common, and can go unnoticed for some time. For example, in Figure 7(a), the destination remains reachable, so an operator might not immediately notice that router 4 does not have complete routing information. Additionally, larger ASes often use more complicated iBGP topologies involving route reflection [3]; in these cases, ensuring a fully connected iBGP topology is more subtle than ensuring a full mesh. Recent work that analyzes errors in BGP configuration has discovered that missing iBGP sessions occur reasonably often [4].

Customer-driven export: Many ASes allow a customer to tag a BGP route with *community* attributes that influence the handling of the route [5, 6]. For example, a customer might be allowed to use the "no export" community [5] to instruct the provider not to export the route to neighboring ASes (e.g., to control its incoming traffic, the customer might advertise a subnet of a larger prefix to its immediate provider but not require that subnet to be propagated further). If the customer connects to the provider in multiple locations, one route might have this tag and another might not, as shown in Figure 7(b). The two customer routes look "equally good," leading routers 3 and 4 to select the closest egress point (routers 1 and 2, respectively). Even if the two routers apply the same export policy, router 3 would export the route but router 4 would not. Similarly, an AS might allow its customers to assign a community that triggers "AS prepending" when a route is exported, which could lead the AS to export routes with different AS path lengths.

Mix of customer and peer routes: An AS may learn routes for a prefix from multiple neighboring ASes. In Figure 7(c), router 1 learns a route from a customer and router 2 learns a route from a peer. Suppose the routes have the same AS path length and that the import policies assign the same local preference to both routes. Then, routers 3 and 4 would receive two "equally good" routes (i.e., with the same local preference and AS path length). Each router would select the route with the closest egress point, leading router 3 to select a customer-learned route and router 4 to select a peer-learned route. However, an AS typically does not export a route learned from one peer to another [7]. Even if routers 3 and 4 apply exactly the same export policy (i.e., "export only customer routes"), router 3 would export a route to d but router 4 would not, leading the local AS to receive a route to the prefix at one peering point and not the other. We recently discovered that this very problem was discussed on the North American Network Operators Group (NANOG) mailing list seven years ago [8].

Designing tools for detecting these kinds of configuration errors and policy conflicts would be very useful for preventing unintentional violations of the "consistent export" requirement.

6. Conclusion and Future Work

Contractual peering agreements often require that two ASes advertise consistent routes at all peering points. Today, ASes can use the algorithm that we propose in Section 3 to detect inconsistent route advertisements from neighboring ASes, as long as the AS's import policies satisfy the conditions we proposed. We note that, for load balancing purposes, import policies that set local preference values based on AS path *length*, rather than on specific ASes in the path, allow the inference algorithm in Section 3 to be applied. Although import policies based on AS path length usually provide sufficient flexibility for performing traffic engineering [9], import policies based on AS path length are occasionally insufficient. In cases where an AS must use these types of import policies, detection of inconsistent route advertisements requires complete access to all of the eBGP routes advertised from that peer; router vendors should add support for monitoring all eBGP-learned routes learned by an AS's border routers.

The algorithms we propose can also be used in conjunction with router configuration and iBGP data to validate previous studies on cold potato routing (e.g., [2]). Cold potato routing must be caused by either local import policy or inconsistent route advertisements. With access to an AS's router configurations and iBGP routing data, we can verify these empirical measurements by examining an AS's import policies and applying our proposed algorithm for detecting inconsistent route advertisements. We intend to explore this further in our future work.

Acknowledgments

We thank Jaeyeon Jung and the anonymous reviewers for helpful comments on a draft of this paper.

7. References

[1] "AOL Transit Data Network: Settlement-Free Interconnection Policy," `http://www.atdn.net/settlement_free_int.shtml`.

[2] Neil Spring, Ratul Mahajan, and Thomas Anderson, "Quantifying the causes of path inflation," in *Proc. ACM SIGCOMM*, August 2003.

[3] T. Bates, R. Chandra, and E. Chen, *BGP Route Reflection - An Alternative to Full Mesh IBGP*, April 2000, RFC 2796.

[4] Nick Feamster and Hari Balakrishnan, "Verifying the correctness of wide-area Internet routing," Tech. Rep. MIT-LCS-TR-948, Massachusetts Institute of Technology, May 2004.

[5] R. Chandra, P. Traina, and T. Li, "BGP communities attribute," Request for Comments 1997, August 1996.

[6] E. Chen and T. Bates, "An application of the BGP community attribute in multi-home routing," Request for Comments 1998, August 1996.

[7] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Trans. Networking*, vol. 9, no. 6, pp. 681–692, December 2001.

[8] Randy Bush, "Consistent Policy != Consistent Announcements," `http://www.merit.edu/mail.archives/nanog/1997-03/msg00250.html`, March 1997.

[9] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford, "Techniques for interdomain traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 5, October 2003.