Address-based Route Reflection

Ruichuan Chen† Aman Shaikh§ Jia Wang§ Paul Francis†
†Max Planck Institute for Software Systems (MPI-SWS), Germany
§AT&T Labs – Research, Florham Park, NJ, USA
{rchen, francis}@mpi-sws.org, {ashaikh, jiawang}@research.att.com

ABSTRACT

BGP Route Reflectors (RR), which are commonly used to help scale Internal BGP (iBGP), can produce oscillations, forwarding loops, and path inefficiencies. ISPs avoid these pitfalls through careful topology design, RR placement, and link-metric assignment. This paper presents Address-Based Route Reflection (ABRR): the first iBGP solution that completely solves all oscillation and looping problems, has no path inefficiencies, and puts no constraints on RR placement. ABRR does this by emulating the semantics of full-mesh iBGP, and thereby adopting the correctness and path efficiency properties of full-mesh iBGP. Both traditional Topology-Based Route Reflection (TBRR) and ABRR take a divide-and-conquer approach. While TBRR scales by making each RR responsible for all prefixes from some fraction of routers, ABRR scales by making each RR responsible for some fraction of prefixes from all routers. We have implemented a fully functional ABRR prototype. Using BGP data from a Tier-1 ISP, our analytical and implementation results show that ABRR's scaling and convergence properties compare positively with traditional TBRR.

1. INTRODUCTION

The original model for Internal BGP (iBGP) was full mesh: all routers in an Autonomous System (AS) peer with all others [8]. This produces a situation where all routers in the AS learn the best paths of all other routers. This allows routers to make best-path decisions consistent with each other (barring configuration errors), and prevents iBGP-initiated instabilities and inefficient paths.

Full-mesh iBGP scales poorly: every router is required to obtain and store state for every other router. To alleviate this scaling problem, the IETF standardized two mechanisms: Route Reflectors (RR) [8] and Confederations [36]. Both of these take a divide-and-conquer approach: they partition the set of AS routers into smaller groups, and do best-path selection first within each group and then across groups. In the case of Confederations, the AS is split into multiple sub-ASes, with each sub-AS operating more-or-less like an AS with its own independent IGP (Interior Gateway Protocol) and internal full-mesh or RR-based iBGP. In the case of RRs, the internal peering structure is hierarchical, with client routers peering only with their parent RRs, and RRs in turn peering full-mesh with each other. This structure can have multiple layers of hierarchy.

RRs and Confederations compromise iBGP consistency: different routers learn different routes from iBGP. This lack of consistency has been shown to produce oscillations, forwarding loops, and path inefficiencies [7, 21, 22, 26, 34]. As the RR solution is very commonly deployed, these problems have been extensively studied for RRs, and a number of solutions have been proposed [7, 10, 18, 22, 25, 26, 32, 34, 38, 40]. All known solutions, however, impose constraints of one sort or another as compared to full-mesh iBGP.

The most common approach, and the one adopted by industry, is to engineer around the problems through careful topology configuration [22, 26]: one or a pair of RRs is placed in most if not every major PoP (Point of Presence), with clients in the same PoP or nearby smaller PoPs. The RRs run full-mesh iBGP with each other. ISPs set IGP metrics so that intra-PoP distances are always shorter than inter-PoP distances. Altogether, this effectively constrains the iBGP topology to match the physical topology. The result is that both MED-based and topology-based oscillations are avoided [26], though this arrangement inevitably restricts the RR placement. This approach also avoids most path inefficiencies: if needed, selected iBGP peering between clients within a PoP, or the announcement of additional routes, could remove the remaining path inefficiencies. Of course, all things being equal, it would be better if the problems associated with RRs and Confederations could be solved fundamentally at the protocol level rather than through careful network design and management.

This paper presents what is to our knowledge the first iBGP solution that:

1. is decentralized in that it divides work among RRs,
2. solves all MED-based and topology-based oscillations, and forwarding loops,

ACM CoNEXT 2011, December 6–9 2011, Tokyo, Japan.

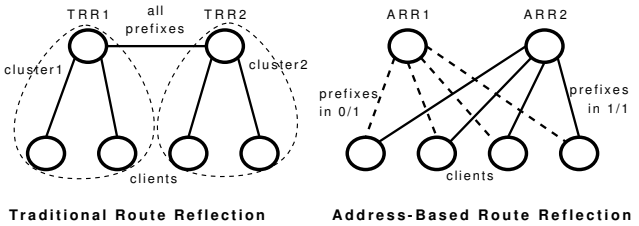


Figure 1: With traditional Route Reflection, routers are assigned to RRs. With Address-Based Route Reflection, address ranges are assigned to RRs. The lines denote iBGP peering sessions.

3. has no iBGP-induced path inefficiencies, and

4. achieves all of the above with no restrictions on RR placement.

This solution, called Address-Based Route Reflection (ABRR), is based on a simple key insight:

> *The BGP best-path decision for any given prefix is independent of that of any other prefix.*

By contrast, the BGP best-path decision for any given prefix at a given router is highly dependent on information provided by other routers. This suggests a much cleaner way to divide-and-conquer iBGP. Rather than divide on a router basis, we can divide on an address basis. This idea is illustrated in Figure 1.

Traditional "Topology-Based" Route Reflection (TBRR) is shown on the left. TBRR route reflectors (TRR) are associated with different routers, forming groups called clusters. iBGP peerings are required between clients and TRRs in a cluster, and also between TRRs. ABRR is shown on the right. Here, route reflectors (ARR) are associated with different address ranges. All clients iBGP-peer with all ARRs, but only advertise prefixes that fall within the address range of the ARR. Each ARR, in its role as an ARR, computes best routes only for some fraction of all prefixes. For each such prefix, however, the ARR has the same knowledge as with full-mesh, because it peers with all other routers and therefore directly learns all advertised routes. Robustness is achieved by simply deploying multiple ARRs for each address range: no coordination between redundant ARRs is required. ABRR can operate with no new BGP message formats, though it does require multi-path capability as defined in the add-paths draft [39].

Our overall approach is to emulate the semantics of full-mesh iBGP. Full-mesh iBGP has efficient paths and does not suffer from oscillations. By emulating full-mesh iBGP, ABRR adopts these characteristics. This emulation is accomplished using two techniques. First, compared with TBRR, ABRR reduces the number of iBGP hops between border routers from three to two. This leads to a crucial property that, from the perspective of any given prefix, ABRR is effectively a centralized approach. Second, ARRs advertise multiple routes, namely the set of routes that tie for best on AS-level path decision metrics. Altogether, this emulation allows routers to learn what they would have learned in full-mesh iBGP. The end result is that ABRR achieves semantics equal to full-mesh iBGP, the scalability and convergence properties comparable to traditional route reflection, and the freedom to place RRs anywhere.

The value of RR placement freedom is more than just academic. Increasingly many ISPs are moving away from hierarchical edge-core topologies, and towards flat full-mesh tunneled topologies, where every border router has a virtual link (usually MPLS) with every other [31]. The need for flat topologies is driven not just by Virtual Private Network (VPN) services, but also by features like traffic engineering and fast re-route. Indeed, the Tier-1 ISP we measured now uses pure control-plane RRs instead of control- and data-plane RRs. While in principle control-plane RRs could be placed anywhere, the ISP is forced to place them near their clients in order to avoid long paths and correctness problems. ABRR frees up both the placement of RRs and the number of RRs. Altogether, this paper makes the following contributions:

- It presents what is to our knowledge the first iBGP solution that removes all limitations on RR placement for both correctness and path-efficiency, while still dividing work among RRs. (§2)

- It analyzes the performance of ABRR and TBRR, and finds that ABRR compares positively with TBRR in terms of storage and convergence time, for both RRs and clients. (§3)

- It gives the implementation results of fully functional ABRR and TBRR using BGP data from a Tier-1 ISP. This validates the scalability of ABRR, and further shows that ABRR requires fewer iBGP updates for both RRs and clients. (§4)

2. DESIGN OF ABRR

2.1 ABRR Protocol

In many respects, ABRR is similar to Topology-Based Route Reflection (TBRR). As with TBRR, routers are Route Reflectors (ARR) or clients of ARRs. Clients are the sources and sinks of iBGP updates, and ARRs reflect updates between clients.

The critical difference between TBRR and ABRR is in how the work of processing updates is partitioned. In TBRR, a Route Reflector (TRR) is responsible for all routes from some fraction of routers (its clients). In ABRR, an ARR is responsible for some fraction of

Table 1: Protocol comparison between TBRR and ABRR. (Note that an ABRR client may also be an ARR, so ARR ↔ Client messages may be conceptually internal to a router.)

TBRR	ABRR
TRR → Client 1. Best routes received from other TRRs 2. Best routes received from clients (not returned to sender) 3. Best routes received from eBGP neighbors 4. Best routes locally originated	**ARR → Client** (for routes in AP only) 1. Best AS-level routes (not returned to sender)
TRR → TRR 1. Best routes received from clients 2. Best routes received from eBGP neighbors 3. Best routes locally originated	**ARR → ARR** Not applicable
TRR → eBGP Neighbor 1. All best routes (not returned to sender)	**ARR → eBGP Neighbor** Not applicable
Client → TRR 1. Best routes received from eBGP neighbors 2. Best routes locally originated	**Client → ARR** (for routes in AP only) 1. Best routes received from eBGP neighbors 2. Best routes locally originated
Client → eBGP Neighbor 1. All best routes (not returned to sender)	**Client → eBGP Neighbor** 1. All best routes (not returned to sender)

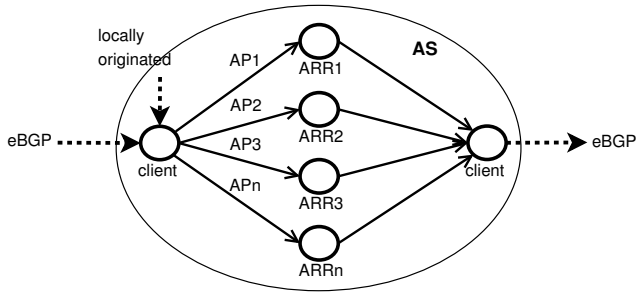


Figure 2: Flow of updates in ABRR. Links represent iBGP peering sessions. The terms ARR and client refer to router roles: the same physical router may be both an ARR and a client.

routes from all routers. In this paper, we assume that address ranges are used to determine which routes any given ARR is responsible for. For instance, ARR_1 could be responsible for 0/8, ARR_2 for 1/8, etc. Each such range is called an Address Partition (AP). An AP can include multiple prefixes, and different APs can overlap. Since prefixes and BGP updates are not evenly distributed across the whole address space, the network administrator can select APs that result in a roughly equal number of routable prefixes and/or BGP updates per AP.

The flow of updates through the AS is shown in Figure 2 (with details in Table 1). Clients run the best-path decision as normal. If a best route is not learned from iBGP, the client advertises it to whichever ARRs are responsible for the destination prefix in the route. If a prefix spans multiple APs, then the associated route is advertised to the ARRs for all such APs. As with full-mesh and TBRR, clients never advertise iBGP-learned routes over iBGP. Clients advertise their best routes to their eBGP neighbors.

The terms client and ARR refer to functional roles in the router. In other words, when we refer to a "client" or "ARR", implicitly we are referring to the client or ARR function within the router. The client function is to be the source and sink of iBGP updates. The ARR function is to reflect received iBGP updates. Any data-plane router must be a client (i.e., has the client function) for every AP. Any router may be an ARR (i.e., has the ARR function) for one or more APs. A data-plane router that is an ARR is also a client for itself, i.e., it has both ARR and client functionality, and logically passes updates between these functional components. If an ARR is a pure control-plane device, then it is an ARR without being a client.

In this paper, when we refer to an ARR, we implicitly mean with respect to a given AP. For instance, imagine two routers R_1 and R_2 that are both ARRs (implied here is that they are ARRs for the same AP). When router R_1 receives an eBGP update for a destination prefix within the AP, if that prefix is its best path it forwards the update to R_2. In so doing, it is acting in its role as a client, and so we can say "client R_1 forwards the update to ARR R_2". Logically the client R_1 also forwards the update to the ARR function within R_1.

The only time two routers are iBGP peers (i.e., have an iBGP peering session) is when one router is a client and the other is an ARR. ARRs do not peer with each other, and clients do not peer with each other. Each

Table 2: BGP best-path selection steps from RFC4271. ARRs compute multiple best AS-level routes using steps 1∼4 only.

#	Decision Process
1	Highest Local Preference
2	Shortest AS Path
3	Lowest Origin Type (IGP→EGP→Incomplete)
4	Lowest Multi-Exit Discriminator (MED)
5	eBGP-learned over iBGP-learned
6	Lowest IGP Metric
7	Lowest Router ID
8	Lowest Peer Address

ARR has an iBGP peering session with every client (we explain in §3.3 why this is not a scaling issue for modern routers). Typically we expect to see no more than 10 or 15 APs, and potentially fewer. This is because the scaling benefits of ABRR reach diminishing returns beyond that point (see §3.2). Assuming two ARRs per AP for redundancy, each client peers with no more than 20 to 30 ARRs.

Each client advertises its eBGP-learned or locally-originated best routes to all ARRs responsible for the AP containing the route (see Table 1). In this fashion, all ARRs learn, for all routes in their AP, everything they would have learned in full-mesh iBGP. All ARRs (for a given AP) receive the same set of iBGP routes.

From its complete set of learned routes, each ARR selects and advertises multiple routes for each prefix *directly* to all clients. In this manner, compared with TBRR, ABRR reduces the number of iBGP hops between border routers from three to two. This leads to a crucial property that, for any given prefix, ABRR is effectively a centralized approach. ARRs are not expected to execute BGP policies. Rather, policies are deployed at clients, as they would be in a full-mesh scenario.

Specifically, each ARR selects all routes that remain after steps one through four of the best-path selection process as described in RFC4271 [33] (see Table 2). We refer to these as the *best AS-level routes*, because they represent all the routes that tie for best in terms of AS-level criteria. Commercial routers typically have additional steps interspersed between these four steps. For instance, Cisco routers have a "largest weight" step before step 1, and a "locally originated" step after step 1. These steps are not considered in selecting the multiple routes. The best AS-level routes are advertised to all clients, with the exception that routes are not advertised to the client from which they were learned (see Table 1). The routes are not advertised to other ARRs since this information would be redundant. The best AS-level routes can be encoded using the existing add-paths mechanism in [39].

The effect of adding these additional best AS-level routes, combined with ABRR's effectively per-prefix centralized approach, is that clients learn the same information they would have learned from full-mesh iBGP. This leads ABRR to adopt the behavioral characteristics of full-mesh iBGP, i.e., it has no oscillations, and no path inefficiencies.[1]

2.2 ABRR Emulates Full-mesh Semantics

To see why ABRR emulates the semantics of full-mesh, we can compare the operation of a router in full-mesh iBGP with that of a client in ABRR. Define an "iBGP-learned" route as a route learned from iBGP, and an "other-learned" route as one learned from eBGP or that is locally originated.

The first thing to note is that an ABRR client transmits into iBGP the same information that a full-mesh BGP router transmits into iBGP. Specifically, if the best route for the client/router is other-learned, then it will advertise that into iBGP. Otherwise, if the best route is iBGP-learned, it advertises nothing into iBGP (or withdraws any previous route). Of course, for the client, "advertise into iBGP" means sending the route to ARRs, whereas for the full-mesh router, it means sending the route to all other routers.

Suppose that a router has an other-learned route r that is its best route among all of its other-learned routes (for a given prefix).

Assume first that r is also that router's best route across all routes (for a given prefix) available to the AS. In other words, even if the router knew of all other routes available to the AS, it would still choose this other-learned route r as its best route. For both full-mesh and ABRR, r will be advertised on iBGP. This is because there are no better routes across the AS, so no matter what routes are received or not received by the router, r will be its best route, and it will therefore advertise on iBGP. In the case of full-mesh, r will go to all other routers. In the case of ABRR, r will go to the ARR, which in turn must necessarily forward it to all clients. This is because r is by definition one of the ARR's best AS-level routes as well. If it were not, that would mean that some other route q would have been selected over r in steps 1∼4 of the selection process, and this route q would have been forwarded to the router. But if this is the case, then q would also be a better route from the router's point of view, thus violating our original assumption.

Next, assume that r is not that router's best route across all routes available to the AS. Rather some other

[1]Note that, in full-mesh iBGP, there is a problem that a border router which ignores the MED attribute might prevent other routers from learning low-MED routes. Nevertheless, this can be easily fixed by appropriate configuration.

route q, which is iBGP-learned, is its best route. The above paragraph shows that q should be advertised to the router in both full-mesh and ABRR. Therefore, for both full-mesh and ABRR, the router will not advertise the route r on iBGP.

Note that it is possible that at a certain time t, the router is not aware of the better route q, and so does advertise route r on iBGP. In the case of full-mesh, r will go to all other routers. In the case of ABRR, there are two possibilities. If the ARR does not know of q, then it will forward route r to all other routers. If, however, the ARR does know of q, it will not forward r. In this case, the actual routes received by routers in full-mesh and ABRR differ. Indeed, if in full-mesh some other routers also do not yet know of q, they may temporarily accept r as their best route. Eventually, however, all routers will hear of q, and the router will withdraw r, so in the steady state the same routes are delivered for both full-mesh and ABRR.

2.3 ABRR Has No Routing Anomalies

2.3.1 No iBGP Oscillations

Note first that our use of best AS-level routes per se is not novel: Basu *et al.* [7] used them to prevent MED-based oscillations. There are some differences, however. In Basu's scheme, all routers advertise all best AS-level routes, while in ABRR only ARRs advertise all best AS-level routes — clients only advertise their best routes (if not iBGP-learned). Nevertheless, the use of best AS-level routes *partially* helps ABRR emulate full-mesh iBGP, and so for this reason also contributes to preventing MED-based oscillations in ABRR.

Note however that simply propagating best AS-level routes does *not* prevent topology-based oscillations [18]. What's more, these oscillations can only occur between RRs [19]. Since with ABRR iBGP routes pass through only one RR, ABRR has no topology-based oscillations. This means, changing the division of labor among RRs from topology-based to address-based makes ABRR a logically centralized approach for any given prefix, which solves the topology-based oscillations.

2.3.2 No Forwarding Loops

In order to know where to send iBGP updates, routers must know which other routers are ARRs and clients. As with TBRR, in ABRR iBGP messages do not loop when routers are configured correctly. Clients never forward iBGP-learned routes on iBGP. ARRs only send iBGP-learned routes to clients, not other ARRs.

Of course, routers are not always configured correctly. Even without mis-configuration per se, router configuration will be transiently out of sync when multiple routers are being updated. Both TBRR and ABRR may have loops if router configuration is inconsistent. In the case of ABRR, imagine a situation where there are three routers, A, B, and C. All three believe that they themselves are ARRs, but the others are clients. In the absence of a mechanism to prevent it, a BGP update in this configuration could hypothetically loop around A to B to C to A, because each would think it is receiving an update from a client, and so deliver it to the other supposed client. Of course, a single looping update would be recognized as old news and not continue beyond the first round, but multiple different updates for the same prefix could potentially chase each other around the loop indefinitely.

Either loop-detection mechanism used by route reflectors today, the Cluster List or the Originator ID [8], can be used to break loops in ABRR. It's worth noting, however, that these are overkill: since an ARR should never forward a route to another ARR, all that is needed to break the loop is a single bit indicating that the update has been reflected by an ARR. In our implementation, we use this approach implemented as an extended community attribute.

2.3.3 No Path Inefficiencies

ABRR has no iBGP-induced path inefficiencies. The reasoning is simple: ABRR emulates full-mesh, and full-mesh has no path inefficiencies (i.e., barring configuration errors, it selects the best next-hop according to IGP metrics or BGP policies). Because of this, the physical placement of ARRs within the ISP is irrelevant, at least for the purpose of path selection. Placement is of course relevant for robustness. An ARR should not be placed where single link or router failures can partition the ARR from the rest of the network. Furthermore, an ISP would want redundant ARRs, each placed in geographically diverse locations so as to minimize dependencies on the same links or routers. Two or three ARRs per AP, however, should be sufficient.

2.4 Transition to ABRR

It is highly desirable for ISPs to be able to transition from TBRR to ABRR without interrupting service. This can be done in an incremental and controlled fashion as follows. Routers must be able to run both TBRR and ABRR, but only accept routes from one or the other, based on configuration. This allows an ISP to cutover to ABRR one AP, or even fraction of an AP, at a time. Specifically, the ISP deploys ABRR for a single AP, and routers run both TBRR and ABRR, but initially accept routes only from TBRR. After the ISP verifies that ABRR is running correctly, it pushes out a configuration causing all routers to accept routes from ABRR for that AP (while still accepting routes from TBRR for APs not yet transitioned). This process repeats for each successive AP until all routes are learned from ABRR. Finally, TBRR can be turned off.

3. PERFORMANCE ANALYSIS

This section, along with Appendix A, provides a pen-and-paper analysis of the performance of ABRR and TBRR for both RRs and clients. The subsequent section details comparative measurements based on a fully functional implementation of ABRR and the `Quagga` [5] implementation of TBRR, using BGP data from a Tier-1 ISP. These measurements validate our analysis.

As suggested in a significant amount of previous research [9, 12, 17, 23, 24, 27, 28], to compare ABRR and TBRR, we are interested in three metrics: the number of routes that need to be stored, the number of iBGP peering sessions required, and the iBGP convergence time. If ABRR performs comparably to TBRR, then we can conclude that ABRR is overall better because of its superior correctness and deployment flexibility.

3.1 Best AS-Level Routes

As described in §2.1, ARRs transmit all best AS-level routes to clients. The scalability of ABRR therefore depends directly on how many best AS-level routes per prefix there are. There are two main causes for multiple best AS-level routes. The first cause is multiple best AS paths to remote ASes. The second cause is multiple peering points between neighboring ASes. We use BGP data collected on Dec 20, 2010 from one AS out of several ASes owned by a Tier-1 ISP to derive the number of best AS-level routes per prefix, and apply the result in our analysis. Note that using data from a Tier-1 ISP gives a conservative estimate, first because a Tier-1 ISP tends to peer with many large peer ASes, resulting in many disparate views and therefore many AS paths; and second because a Tier-1 ISP tends to have many peering points with each peer AS.

Our Tier-1 ISP data contains about 416K unique prefixes, roughly 76% of which come from peer ASes, and the rest are locally originated or from customer ASes. There are more than 1000 BGP routers, less than 10% of which are *peering routers* collectively peering with 25 peer ASes. On average, the number of peering points with each peer AS is about 8. The majority of other routers are *access routers* connected to customer ASes.

Figure 3 shows the average number of best AS-level routes per prefix given the number of peer ASes. We derive this number by selecting the peers of the Tier-1 AS at random, and then with these selected peers, measuring the number of best AS-level routes per prefix. We plot two curves. The "Peer ASes Only" curve considers only routes learned from these selected peer ASes, while the "All Sources" curve considers routes learned from various sources, not only including randomly selected peer ASes (as in "Peer ASes Only"), but always including all the customers and static routes. The result demonstrates that, for a Tier-1 ISP, path diversity comes primarily from peer ASes. For lower-tier ISPs,

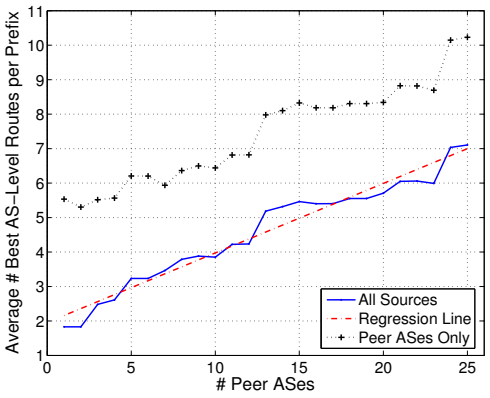


Figure 3: Best AS-level routes per prefix

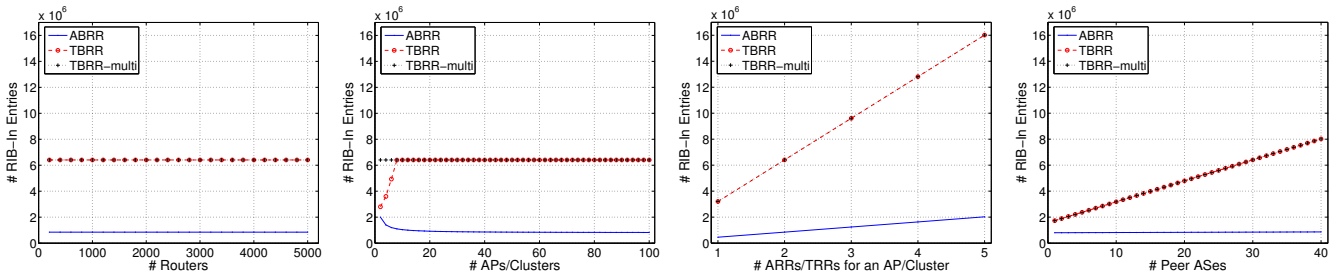
we would expect to see little path diversity from peers, since lower-tier ISPs generally only exchange static and customer routes with peers [20]. The primary source of path diversity for lower-tier ISPs would be their transit providers. Generally, lower-tier ISPs have only a handful of transit providers, and so we would expect their average number of best AS-level routes per prefix to be towards the left side of the graph in Figure 3.

We fit a regression line to the "All Sources" curve, denoted as $\mathcal{F}(\#PASs)$, where PAS denotes Peer AS. Therefore, the average number of best AS-level routes per prefix $\#BAL = \mathcal{F}(\#PASs)$, which will be used in the remainder of our analysis.

3.2 RIB-In and RIB-Out Analysis for RRs

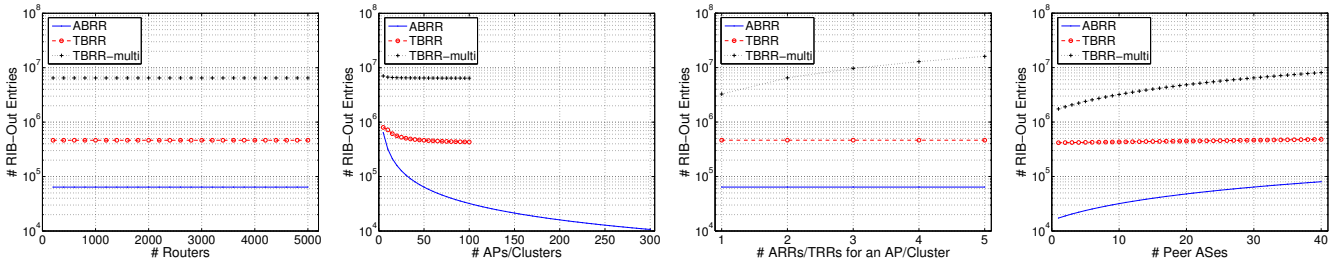
Our definition of RIB-In corresponds to the conceptual Adj-RIB-In defined by BGP [33], consisting of the routes reported by every BGP neighbor. Our definition of RIB-Out consists of the routes reported to BGP neighbors, and thus corresponds to the conceptual Adj-RIB-Out defined by BGP [33]. In our analysis, we derive ARR's and (traditional) TRR's RIB-In and RIB-Out sizes, respectively, under a few practical simplifying assumptions. Appendix A describes the analytical procedure and expressions in detail.

Traditional TBRR assumes that TRRs advertise a single best path. In reality, there is an increasing interest among ISPs to advertise additional paths. A comparison between ABRR and single-path TBRR is fair in the sense that single-path TBRR is predominantly how ISPs deploy TBRR today. The comparison is unfair, however, in the sense that ABRR provides multiple paths that may be exploited for traffic engineering and fast re-route. If an ISP wished to exploit multiple paths, then a fairer comparison would be between ABRR and a multi-path version of TBRR. For this reason, we provide a second analysis of TBRR, one where each TRR maintains and advertises all best AS-level routes. The analytical procedure and expressions for



(a) (b) (c) (d)

Figure 4: # RIB-In Entries of an ARR/TRR. Default values are 2000 routers, 50 APs/Clusters, 2 ARRs/TRRs per AP/Cluster, and 30 peer ASes. Note that, in (a), (c), and (d), the plots for TBRR and TBRR-multi are identical.



(a) (b) (c) (d)

Figure 5: # RIB-Out Entries of an ARR/TRR. Default values are 2000 routers, 50 APs/Clusters, 2 ARRs/TRRs per AP/Cluster, and 30 peer ASes.

multi-path TRR can be found in Appendix A.3.

Figures 4 and 5 plot the analytical results for various RRs' RIB-In size and RIB-Out size, respectively. These figures show the scaling effect of four parameters: the number of routers, the number of APs/Clusters, the number of ARRs/TRRs per AP/Cluster, and the number of peer ASes. The intent here is not to give exact numbers for some specific ISP, but to demonstrate the general effect of these parameters, and the relative differences between ABRR and TBRR (single- and multi-path). The default setting of these parameters, taken to model a typical large-scale AS [35], are 2000 routers, 30 peer ASes, and 50 APs/clusters each managed by 2 ARRs/TRRs. Each figure varies one of the parameters while holding the others constant at their default values. We assume 400K unique prefixes.

The primary takeaway from these figures is that, for virtually *all* parameter settings, ABRR has substantially smaller memory requirement than TBRR.

From Figure 4, we see that the parameter that most influences the number of RIB-In entries in ABRR is the number of ARRs per AP, i.e., the "redundancy factor". This is because routers (ARRs and clients alike) must maintain a route per ARR. Fortunately, two or three ARRs per AP are sufficiently redundant, and so memory growth here is quite limited.

It is interesting to see the effect that the number of APs has on both RIB-In and RIB-Out sizes (Figures 4b and 5b respectively). The benefit to RIB-In size from increasing the number of APs quickly reaches diminishing returns. The number of RIB-In entries is dominated by the need for an ARR, in its role as a client, to maintain the default-free zone (DFZ) routing table.

In the case of RIB-Out, however, it is a different story. Since ARRs only need to advertise prefixes in their AP to all clients, the RIB-Out size can be steadily reduced by increasing the number of APs. While the possible number of APs is limited only by the number of routers in the network, the number of clusters is generally limited by the number of major PoPs. In Figure 5b, this fact is emphasized by truncating the curves for TBRR at 100 clusters.

On the other hand, increasing the number of APs also increases the number of ARRs, and therefore the number of iBGP peering sessions that clients must maintain. This effect is not seen in TBRR, since clients only iBGP-peer with the TRRs in their cluster, and therefore the number of clusters does not matter for clients. This is further discussed in §3.3 and §3.4.

3.3 Peering Sessions of RRs

In ABRR, every ARR has an iBGP session with every

other router in the AS. By contrast, in TBRR, every TRR has iBGP sessions with only its clients and other TRRs. To make this concrete, in the Tier-1 AS we measured, the TRR with the most sessions has about 200, and the average is about 100. Each ARR in this network would require over 1000 sessions.

What does this mean in terms of processing and bandwidth? Figure 5 shows that the ARR's RIB-Out is roughly an order of magnitude smaller than the TRR's RIB-Out. This means that the number of BGP updates that must be generated is correspondingly an order of magnitude smaller (verified in Figure 7). On the other hand, the ARR has perhaps an order of magnitude more iBGP peers. This means that, once generated, a BGP update may be transmitted more times. From a processing point of view, generating an update is far more expensive than receiving or transmitting one [6]. Indeed BGP peer groups, where sets of peers can be grouped into identical policy sets, exploit this characteristic [3]. In other words, ABRR trades-off a substantial gain in processing for a modest loss in bandwidth.

Putting aside BGP-level processing, there is also a cost to simply maintain TCP and BGP sessions — session states, TCP control blocks, keep-alive messages, etc. While non-trivial, it is well-known how to scale TCP sessions. TCP offload engines like [2] can handle tens of thousands of simultaneous connections through techniques like buffer reuse [13]. Nowadays, the business need for many BGP sessions has steadily increased, and modern routers like the Cisco ASR1000 series can handle several thousand BGP sessions, and have been tested to 8000 sessions each with the full routing table [6]. This is well within the session requirement of an ARR deployed in even the largest ISPs. In addition, [11] implies that general-purpose computers on the control-plane can also scale to a large number of peering sessions. We note that ARR's boot time may increase somewhat because of the large number of sessions that need to be established. However, because ARRs generally are redundant, and are not in the data path, boot time is not a critical issue.

An increased number of sessions also increases the configuration requirement. ISPs, however, can and often do automate BGP session configuration in their network management systems. Alternatively, BGP peer discovery could in principle be automated through IGP, for instance as has been proposed in the IETF [30].

3.4 Analysis for Clients

Although ABRR clients receive all best AS-level routes from ARRs, they only need to store the best routes. This is because, should there be a change in the set of best AS-level routes, the ARRs will convey all such routes to the clients with each update. Of course, ABRR clients can choose to store multiple routes for the purposes of traffic engineering or fast re-route. Apples-to-apples, we can regard the RIB-In and RIB-Out sizes for ABRR and TBRR clients to be comparable. Because of multiple routes in each update, the cost of processing a single received update is more for ABRR clients; however, on the other hand, due to race conditions as updates propagate through TRRs, TBRR clients will receive more updates than ABRR clients (see §4.2).

Regarding the number of peering sessions, Figures 4b and 5b show that only 10 or 15 APs are needed for ARRs to achieve substantially smaller RIBs than TRRs. This means that, in reasonable configurations with 2 ARRs per AP, ABRR clients should have no more than 20 or 30 iBGP peering sessions, as compared to two for TBRR clients. This is well within the capabilities of the existing installed base. Overall, the scalability of ABRR and TBRR clients are similar.

3.5 iBGP Convergence Time

The primary contributor to convergence time is the Minimum Route Advertisement Interval (MRAI) timer, which determines the minimum interval between the sending of BGP updates to a given peer for a common set of destinations [33]. For iBGP, the suggested default is 5 seconds, a value that is used by many major router vendors. Since ABRR reduces the number of iBGP hops between border routers from three (or occasionally more) in TBRR to two, the impact of MRAI delay can be considerably reduced by ABRR.

Not all updates are subject to the MRAI delay, and an ISP can choose to set its value lower. In cases where MRAI delay does not come into play, ABRR has pros and cons, though minor either way. On the pro side is the fact that there is one less iBGP hop in ABRR. Furthermore, an ARR requires much fewer iBGP updates (see §4.2), thus less processing delay. On the con side, the RR hop in ABRR may be far across the network, whereas with TBRR, clients tend to be close to their RRs. In addition, an ARR must transmit a given update to many clients, thus incurring some transmission delay for the clients that receive the update. Nevertheless, these delays are quite small (milliseconds), and should therefore not be a factor in deciding for or against ABRR. In any event, there is no reason to believe that ABRR would significantly increase iBGP convergence time. Indeed, in some cases (e.g., with MRAI) ABRR can significantly reduce it.

3.6 Analysis Summary

The main takeaway from this section is that ABRR and TBRR scalability are broadly comparable. ARRs require substantially less memory than TRRs, and process correspondingly fewer BGP updates. ARRs have a large number of iBGP sessions, but in practice this is not an issue with either modern routers or general-

purpose computers acting as control-plane RRs. Moreover, client scalability differs little between ABRR and TBRR. Likewise, convergence time is also comparable or in some cases significantly improved by ABRR.

4. IMPLEMENTATION RESULTS

We implemented ABRR with less than 2000 lines of C code on the version 0.99.15 code base of `Quagga` [5], a routing software package with support for OSPF and BGP among others. We deployed ABRR on our testbed running Linux 2.6.32. On the testbed, we additionally developed a simple pseudo BGP speaker, called route regenerator, which uses the MRT-format routing trace to direct BGP feeds towards our implementation. This testbed allowed us to thoroughly exercise our implementation, do apples-to-apples comparisons of ABRR with TBRR, and verify the analytical results from §3.

Although we ran various experiments to measure the performance, the results we report in this section come from modeling the iBGP topology of the Tier-1 AS as used in §3. It has more than 1000 BGP routers spread over 27 clusters with 2 TRRs per cluster. This topology exceeds our testbed's capability, so we limit our experiments to all peering routers and their corresponding TRRs, which are spread over 13 clusters. The majority of prefixes (about 315K) and best AS-level routes (10.2 per prefix) come from peer ASes, so this model captures the main scaling challenges. For ABRR, we configure between 1 and 32 APs, each with 2 ARRs. The address range size for each AP is the same.

The BGP update traces for our experiments were taken from all peering routers starting on Dec. 20, 2010, and extending two weeks. The traces were obtained from the Tier-1 ISP's BGP monitor, which collects and timestamps BGP updates from the peering routers in realtime. We start our trace by taking a snapshot of the peering routers' RIBs, and generating a series of BGP announcements from our route regenerators to the appropriate RRs in order to initialize the state. We then feed the subsequent two-week BGP updates to the RRs. These updates are delivered in the order they were received by BGP monitor. We don't emulate the link latency between routers. In other words, the absolute timing of neither the original BGP updates nor update handling at original routers is preserved. To partially gauge the effect of this lack of timing fidelity, we ran pairs of experiments where the route regenerators fed updates at different rates. In one case, they were sent as fast as possible (roughly 20x faster than realtime). In another case, they were sent in realtime. The resulting numbers of updates differed by no more than 3%.

4.1 RIB-In and RIB-Out Sizes

Figure 6 gives the min, max, and average RIB-In and RIB-Out sizes across all RRs after the initial snapshot.

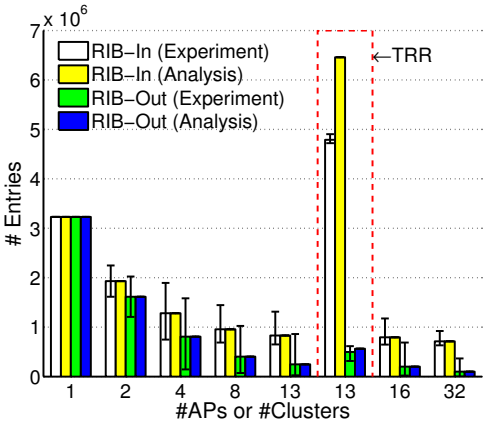


Figure 6: RIB-In and RIB-Out sizes of an ARR/TRR. Note that, the histograms within the red dashed rectangle are for TRR with #Clusters=13, and all others are for ARR with various #APs.

It also gives the expected sizes from the analysis. The result shows that the average experimental number of RIB-In and RIB-Out entries for ARR matches the analysis exactly. The min and max, however, differ significantly from the average: as much as 50% in some cases. This is because our experiment uses the uniform address ranges each with different numbers of routable prefixes. ISPs, however, can easily control this variance by selecting address ranges that have the appropriate percentage of prefixes.

The average experimental number of RIB entries for TRR in Figure 6 does not match the analysis exactly. For RIB-In, the analysis overestimates by 34.9%, and for RIB-Out, by 13.4%. This is because our analysis makes the simplifying assumptions that peering routers are uniformly distributed among clusters, and best AS-level routes are uniformly distributed among peering routers (see Appendix A.2). These maximize TRR's RIB-In and RIB-Out sizes. In the Tier-1 AS we measured, the total number of best AS-level routes (over all prefixes) learned by individual peering routers varies from 0 to 206,635. This also explains the min/max variance. Nevertheless, as expected, the RIB-In and RIB-Out for ARRs are substantially smaller than for TRRs.

4.2 Number of Updates

In this section, we first consider the number of updates received, generated, and transmitted by ARRs and TRRs. Generated updates are updates to the RIB-Out. The distinction between generated and transmitted is important because it requires far more processing to generate an update than to transmit one. We also consider the number of updates received by clients.

Figure 7 gives the number of updates received and generated by the RRs during two weeks. The result

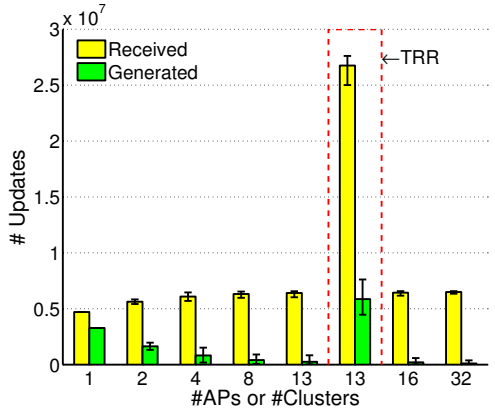


Figure 7: Number of updates received and generated by an ARR/TRR over two weeks. Note that, the histograms within the red dashed rectangle are for TRR with #Clusters=13, and all others are for ARR with various #APs.

shows that the number of received updates by ARRs increases with the number of APs. However, the rate of increase flattens out. Figure 7 also indicates that the number of updates generated by ARRs shrinks dramatically with the number of APs, and in general is a small fraction of the number for TRRs. This is in small part due to the fact that the ARR RIB-Out is about one-half the size of the TRR RIB-Out (see Figure 6), and in large part due to the fact that, in ABRR a change of route only goes to its two ARRs, while in TBRR a change of route occurs at possibly many TRRs.

To determine the total number of transmitted updates, we compute the number of transmissions that would have been required to send updates to all clients. For this, we emulate the Tier-1 AS's full iBGP topology with all 27 clusters for TBRR, and a corresponding ABRR topology with 27 APs. The result is that, on average, each TRR transmits 2.5 times more updates: 310 updates/second versus 125 updates/second for ARR. However, for ABRR, each of these updates has on average 10.2 (best AS-level) routes per prefix, and is therefore roughly 10 times longer than its TBRR counterpart. Hence, from the point of view of transmission bandwidth, an ARR transmits roughly four times more bytes. For route reflectors, ABRR effectively trades-off a modest loss in bandwidth for a substantial gain in processing for both received and generated updates.

Regarding the number of updates received by clients, we surprisingly find that ABRR clients receive roughly 30% fewer updates than TBRR clients[2]. This is particularly non-intuitive given that ARRs send updates to clients once any best AS-level routes changed, while TRRs send updates only when the best routes changed, and so presumably ABRR clients would receive more updates. The reason for ABRR's better performance here is *race conditions* in TBRR. It is a frequent occurrence (observed both in the original Tier-1 ISP traces and in our experiment) that updates for the same routing event, which are processed at roughly the same time at different clients in different clusters, are nevertheless processed by their respective TRRs at different times (by 100's of ms to several seconds in the Tier-1 ISP traces, and 100's of ms in our experiment). The end result is that a given TRR will receive a series of updates for the same routing event from different TRRs. Some of these updates will provide a better route than a previously received one. Each time this happens, the TRR will transmit the new update to its clients. A similar phenomenon [14] has been separately observed on the propagation of unnecessary BGP withdraws.

By contrast, when a set of ABRR clients receive updates at roughly the same time for a given routing event, they will forward the updates directly to the same ARR. Due to the timing of update processing at ARRs, the ARR will normally have received most or all of these updates by the time it actually processes them, and therefore, this ARR only needs to transmit the resulting single update to its clients.

[2]This is after adjusting for the fact that, in the Tier-1 AS's iBGP topology, about 20% of TBRR clients belong to two clusters, and therefore receive twice as many updates.

5. RELATED WORK

The problems associated with traditional route reflection — MED-based oscillations, topology-based oscillations, forwarding loops, and path inefficiencies — were early reported in [4, 15] and then extensively studied [7, 21, 22, 26, 34].

In the past decade, a lot of effort has been devoted to solving these routing anomalies in traditional route reflection. Much of this work is focused on how to design and configure the network topology to eliminate these anomalies. In [22], Griffin and Wilfong described the sufficient conditions for correct (two-level) iBGP configuration: the RRs always prefer the routes learned from clients over those learned from non-clients, and each shortest path between any two routers should be a valid "signaling path". However, [38] suggested that these two conditions are too restrictive for designing correct iBGP topologies. Moreover, in [18], Flavel and Roughan showed that, in multi-level hierarchies, configuring IGP metrics such that downstream routers are closer than any others does not prevent oscillations. In [38], Vutukuru *et al.* described a graph separator to choose the RRs and iBGP sessions by recursively partitioning the network into multiple connected components while guaranteeing the correctness. In [32], Rawat and Shayman modeled an AS using the IGP connectivity graph and iBGP peering graph, and constructed an oscillation/loop-proof iBGP configuration. With these

configuration guidelines, network operators could design an optimized network, though not all of the guidelines are strictly followed by ISPs. In [16], Feamster and Balakrishnan developed a router configuration checker to help network operators find faults in BGP configurations using static analysis. ABRR overcomes the need for "topology-engineered" solutions, and in particular the need to restrict the placement of RRs.

Other approaches aim to solve route reflection problems at the protocol level. In [7], each router advertises all best AS-level routes to all iBGP neighbors. This approach solves MED-based oscillations, but not the topology-based oscillations [18]. Similar to [7], ABRR also utilizes the best AS-level routes. ABRR, however, solves all oscillation problems. In [18], all oscillations are solved by introducing a new step in the BGP best-path selection process: minimizing logical iBGP hops before minimizing physical IGP metrics. This introduces the problem of inefficient paths, which can only be solved through restricted placement of RRs so that the iBGP topology is close to or the same as the physical topology. Unlike [18], ABRR has no restrictions on RR placement even with respect to path efficiency.

The Routing Control Platform (RCP) [11] takes a very different approach to the iBGP problem. Rather than full-mesh, routing updates are sent to a centralized control-plane BGP speaker (the RCP) which makes best-path decisions and feeds these back to routers. These decisions consider the physical topology of the network so that there are no path inefficiencies. This work showed that modern general-purpose computers scale adequately to handle the load, and that the RCP can be replicated for robustness. The extension [37, 29] of the basic RCP idea goes even further, centralizing eBGP peering and all routing policies into a small overlay, thus scaling the iBGP control plane and simplifying policy configuration. We find this RCP argument in many respects to be compelling. However, RCP is a substantial departure from how ISPs operate today; in contrast, ABRR can be executed within the existing router installed base. An ISP may reasonably worry, for instance, that while RCP can handle today's routing requirements, a new requirement like, say, per-voice-call traffic engineering, might overwhelm the RCP.

6. CONCLUSION

This paper presents Address-Based Route Reflection (ABRR) that, for the first time, solves all oscillation problems and finds efficient paths while placing no constraints on RR placement. ABRR requires no new BGP message formats, though it does require the capability of add-paths [39].

Using BGP data from a Tier-1 ISP, our performance analysis and implementation results demonstrate that ABRR's scaling and convergence properties compare positively with traditional topology-based route reflection (TBRR). Overall, combined with ABRR's superior correctness and deployment flexibility, we conclude that ABRR is substantially better than TBRR.

7. REFERENCES

[1] AT&T Global IP Network Peering Policy. http://www.corp.att.com/peering/.

[2] Benefits of SSL Acceleration and TCP Connection Offload Using the Cisco Application Control Engine Module. http://www.cisco.com/web/SG/learning/mindef/files/ACE_TCP_Offload.pdf.

[3] Cisco BGP Peer Groups. http://www.cisco.com/warp/public/459/29.pdf.

[4] Cisco Field Notice: Endless BGP Convergence Problem in Cisco IOS Software Releases. http://www.cisco.com/en/US/ts/fn/100/fn12942.html.

[5] Quagga Software Routing Suite. http://www.quagga.net/.

[6] Private communications with R. Raszuk from Cisco, Oct. 2010.

[7] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. T. Wilfong. Route oscillations in I-BGP with route reflection. In *SIGCOMM*, 2002.

[8] T. Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456, Apr. 2006.

[9] T. Bu, L. Gao, and D. F. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45(1):45–54, 2004.

[10] M.-O. Buob, S. Uhlig, and M. Meulle. Designing Optimal iBGP Route-Reflection Topologies. In *Networking*, 2008.

[11] M. Caesar, D. F. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. E. van der Merwe. Design and Implementation of a Routing Control Platform. In *NSDI*, 2005.

[12] D.-F. Chang, R. Govindan, and J. S. Heidemann. An empirical study of router response to large BGP routing table load. In *Internet Measurement Workshop*, 2002.

[13] A. Cohen and R. Cohen. A Dynamic Approach for Efficient TCP Buffer Allocation. *IEEE Trans. Computers*, 51(3):303–312, 2002.

[14] V. V. den Schrieck, P. François, C. Pelsser, and O. Bonaventure. Preventing the Unnecessary Propagation of BGP Withdraws. In *Networking*, 2009.

[15] R. Dube. A comparison of scaling techniques for BGP. *Computer Communication Review*, 29(3):44–46, 1999.

[16] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *NSDI*, 2005.

[17] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel. HAIR: Hierarchical Architecture for Internet Routing. In *ReArch*, 2009.

[18] A. Flavel and M. Roughan. Stable and flexible iBGP. In *SIGCOMM*, 2009.

[19] A. Flavel, M. Roughan, N. Bean, and A. Shaikh. Where's Waldo? Practical Searches for Stability in iBGP. In *ICNP*, 2008.

[20] L. Gao. On Inferring Autonomous System Relationships in the Internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745, 2001.

[21] T. Griffin and G. T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *ICNP*, 2002.

[22] T. Griffin and G. T. Wilfong. On the correctness of IBGP configuration. In *SIGCOMM*, 2002.

[23] E. Karpilovsky and J. Rexford. Using forgetful routing to control BGP table size. In *CoNEXT*, 2006.

[24] D. Krioukov, K. C. Claffy, K. Fall, and A. Brady. On Compact Routing for the Internet. *Computer Communication Review*, 37(3):41–52, 2007.

[25] P. Marques, R. Fernando, E. Chen, P. Mohapatra, and H. Gredler. Advertisement of the best external route in BGP. IETF Internet-Draft draft-ietf-idr-best-external-04, Apr. 2011.

[26] D. McPherson, V. Gill, D. Walton, and A. Retana. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. RFC 3345, Aug. 2002.

[27] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 address allocation and the BGP routing table evolution. *Computer Communication Review*, 35(1):71–80, 2004.

[28] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, Sept. 2007.

[29] I. Oprescu, M. Meulle, S. Uhlig, C. Pelsser, O. Maennel, and P. Owezarski. oBGP: An Overlay for a Scalable iBGP Control Plane. In *Networking*, 2011.

[30] R. Raszuk. OSPF Extensions for BGP Peer Discovery. IETF Internet-Draft draft-raszuk-ospf-bgp-peer-discovery-00, June

2003.
[31] R. Raszuk. To Add-Paths or not to Add-Paths. NANOG48, Feb. 2010.
[32] A. Rawat and M. A. Shayman. Preventing persistent oscillations and loops in IBGP configuration with route reflection. *Computer Networks*, 50(18):3642–3665, 2006.
[33] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Jan. 2006.
[34] J. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, 2005.
[35] N. T. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *SIGCOMM*, 2002.
[36] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 5065, Aug. 2007.
[37] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. E. van der Merwe. Wresting Control from BGP: Scalable Fine-Grained Route Control. In *USENIX Annual Technical Conference*, 2007.
[38] M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *INFOCOM*, 2006.
[39] D. Walton, E. Chen, A. Retana, and J. Scudder. Advertisement of Multiple Paths in BGP. IETF Internet-Draft draft-ietf-idr-add-paths-06, Sept. 2011.
[40] D. Walton, A. Retana, E. Chen, and J. Scudder. BGP Persistent Route Oscillation Solutions. IETF Internet-Draft draft-walton-bgp-route-oscillation-stop-04, Mar. 2011.

APPENDIX

A. RIB ANALYSIS FOR RRS

In our analysis, we wish to derive RIB-In and RIB-Out sizes of RRs given the following input parameters:

- $\#Prefixes$: the total number of prefixes
- $\#APs$ or $\#Clusters$: the total number of APs or clusters
- $\#ARRs$ or $\#TRRs$: the total number of ARRs or TRRs
- $\#PASs$: the total number of peer ASes (from which we derive $\#BAL$ as already stated in §3.1)

A.1 ABRR

Define the term *managed routes* as those routes an ARR maintains in its role as an ARR (learned from clients), and *unmanaged routes* as those routes an ARR maintains in its role as a client (learned from ARRs for other APs). The total RIB-In size $\mathcal{S}_{\text{RIB-In}_{\text{ARR}}}$ is then the sum of the total due to managed routes $\mathcal{S}^m_{\text{RIB-In}_{\text{ARR}}}$ and the total due to unmanaged routes $\mathcal{S}^u_{\text{RIB-In}_{\text{ARR}}}$:

$$\mathcal{S}_{\text{RIB-In}_{\text{ARR}}} = \mathcal{S}^m_{\text{RIB-In}_{\text{ARR}}} + \mathcal{S}^u_{\text{RIB-In}_{\text{ARR}}}$$

We make the simplifying assumption that an ARR (or TRR for that matter) is not a border router. This keeps the math simple and allows us to isolate ARR (TRR) performance from border router performance. We also make the simplifying assumption that each ARR only manages one AP, and that prefixes are evenly distributed among APs. It is well-known that the prefixes are actually not evenly distributed. This is, however, completely controllable by having an ISP select APs that have roughly equal numbers of prefixes. Therefore, for managed routes, each AP contains $\frac{\#Prefixes}{\#APs}$ prefixes on average, and each prefix contributes $\#BAL$ best AS-level routes:

$$\mathcal{S}^m_{\text{RIB-In}_{\text{ARR}}} = \#BAL \times \frac{\#Prefixes}{\#APs}$$

The number of redundant ARRs for any given AP is $\frac{\#ARRs}{\#APs}$. For unmanaged routes, while each ARR in its role as a client receives all best AS-level routes for each prefix not within its own AP, it maintains only one best route to each redundant ARR for each such prefix. This is because, should there be a change in the set of best AS-level routes for those unmanaged prefixes, the corresponding ARRs will transmit all such routes again. Therefore,

$$\mathcal{S}^u_{\text{RIB-In}_{\text{ARR}}} = \frac{\#ARRs}{\#APs} \times \#Prefixes \times \left(1 - \frac{1}{\#APs}\right)$$

Regarding RIB-Out size, ARRs advertise the best AS-level routes from among their managed routes to all clients (excluding other ARRs for the same AP), with the exception that they do not transmit routes to clients from which they received the routes. We assume that ARRs have configured a single peer group. This is reasonable because all best AS-level routes are transmitted to all clients, and so there is no reason to distinguish between clients at least on the basis of topology. This produces a RIB-Out of:

$$\mathcal{S}_{\text{RIB-Out}_{\text{ARR}}} = \mathcal{S}^m_{\text{RIB-In}_{\text{ARR}}}$$

A.2 Traditional TBRR

As with the ABRR analysis, we define *managed routes* as those routes that a TRR learns from its clients, and *unmanaged routes* as those learned from other TRRs. The total RIB-In size is then:

$$\mathcal{S}_{\text{RIB-In}_{\text{TRR}}} = \mathcal{S}^m_{\text{RIB-In}_{\text{TRR}}} + \mathcal{S}^u_{\text{RIB-In}_{\text{TRR}}}$$

Clients advertise a route if it is the best route and it is eBGP-learned or locally originated, and withdraw or not advertise it otherwise. There are $\#BAL$ best AS-level routes per prefix across the entire AS, and so within each cluster there are on average $\frac{\#BAL}{\#Clusters}$ best AS-level routes advertised. The TRRs in a cluster must maintain these routes, and so the RIB-In size due to managed routes is:

$$\mathcal{S}^m_{\text{RIB-In}_{\text{TRR}}} = \frac{\#BAL}{\#Clusters} \times \#Prefixes$$

The RIB-In entries for unmanaged routes are of course the entries advertised by other TRRs. In normal TBRR operation, each TRR advertises the best route from its cluster to other TRRs. This is because clusters are normally configured such that intra-cluster routes are preferred over inter-cluster routes [22, 26]. A TRR will therefore advertise one route per prefix no matter how many best AS-level routes (if not zero) are advertised by its clients. The total number of routes advertised by all TRRs, then, depends on the distribution of peering points among clusters, and the distribution of best AS-level routes among peering points. While real ISPs do not exhibit the uniform distribution of peering points and best AS-level routes, they would be close to the uniform distribution. This is because ISPs are motivated to provide short paths to exit points, and therefore distribute those exit points evenly across their topologies. For instance, AT&T peering policy mandates that multiple peering points be geographically diverse [1].

A TRR will advertise to other TRRs one route per prefix if the number of best AS-level routes is equal to or greater than the number of clusters, and on average will advertise $\frac{\#BAL}{\#Clusters}$ routes otherwise. Given this, we define a function $\mathcal{G}$ to characterize the total number of routes a TRR will advertise to another TRR as:

$$\mathcal{G}(.) = \begin{cases} \frac{\#BAL}{\#Clusters} \times \#Prefixes & \text{if } \#BAL < \#Clusters \\ \#Prefixes & \text{if } \#BAL \geq \#Clusters \end{cases}$$

A TRR will receive $\mathcal{G}(.)$ routes from every other TRR. Therefore, the size of a TRR's RIB-In from unmanaged routes is:

$$\mathcal{S}^u_{\text{RIB-In}_{\text{TRR}}} = \mathcal{G}(.) \times (\#TRRs - 1)$$

Regarding TRR's RIB-Out size, we assume that TRRs have configured two peer groups, one for clients and another for other TRRs. This is, for instance, the configuration used by the Tier-1 AS we measured. We can divide the TRR's advertised routes into two types. First are the routes that it advertises to other TRRs, the number of which is captured in the function $\mathcal{G}(.)$. These are advertised to all other TRRs, as well as to all clients except the client from which the route was learned. Second are the remaining $(\#Prefixes - \mathcal{G}(.))$ best routes which are learned from other TRRs and only advertised to all clients. Therefore,

$$\mathcal{S}_{\text{RIB-Out}_{\text{TRR}}} = \mathcal{G}(.) \times 2 + (\#Prefixes - \mathcal{G}(.)) \times 1$$

A.3 Multi-Path TBRR

The analysis of RIB-In and RIB-Out sizes here is quite similar to that for traditional single-path TBRR (see §A.2), with the exception that the first term in the function $\mathcal{G}(.)$ is always used because we do not cap the number of routes advertised by a TRR to one per prefix. Without further explanation, the analysis of RIB-In and RIB-Out sizes for multi-path TRR is as follows:

$$\begin{aligned}
\mathcal{S}^m_{\text{RIB-In}_{\text{multi}}} &= \mathcal{S}^m_{\text{RIB-In}_{\text{TRR}}} = \frac{\#BAL}{\#Clusters} \times \#Prefixes \\
\mathcal{S}^u_{\text{RIB-In}_{\text{multi}}} &= \mathcal{S}^m_{\text{RIB-In}_{\text{multi}}} \times (\#TRRs - 1) \\
\mathcal{S}_{\text{RIB-In}_{\text{multi}}} &= \mathcal{S}^m_{\text{RIB-In}_{\text{multi}}} + \mathcal{S}^u_{\text{RIB-In}_{\text{multi}}} \\
\mathcal{S}_{\text{RIB-Out}_{\text{multi}}} &= \mathcal{S}^m_{\text{RIB-In}_{\text{multi}}} \times 2 + \mathcal{S}^u_{\text{RIB-In}_{\text{multi}}} \times 1
\end{aligned}$$