

Address-based Route Reflection

Ruichuan Chen, Paul Francis

Max Planck Institute for Software Systems

Kaiserslautern-Saarbruecken, Germany

{rchen, francis}@mpi-sws.org

Abstract

BGP Route Reflectors (RR), which are commonly used to help scale Internal BGP (iBGP), can produce instabilities, forwarding loops, and path inefficiencies. ISPs mitigate these problems 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, puts no constraints on RR placement, and scales comparably to traditional Topology-Based Route Reflection (TBRR). 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 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 for all routers. This paper describes ABRR, and provides static performance analysis and implementation results that support these claims.

1 Introduction

The baseline configuration for Internal BGP (iBGP) is full mesh: all routers in an Autonomous System (AS) peer with all others [10]. This produces a situation where all routers in the AS learn the best path of all other routers. This allows routers to make best-path decisions consistent with each other, 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) [10] and Confederations [37]. 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 indepen-

dent 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.

R Rs and Confederations compromise iBGP consistency: different routers learn different routes from iBGP. This lack of consistency has been shown to produce instabilities, forwarding loops, and path inefficiencies [9, 22, 23, 28, 35]. As the RR solution is very commonly deployed, these problems have been extensively studied for RRs, and a number of partial or complete solutions have been proposed [9, 13, 20, 23, 27, 28, 33, 35, 39, 41]. 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 by constructing the topology so that they don't arise [23, 28]. Industry has largely converged on a style of RR placement that avoids these problems. One or a pair of RRs is placed in most if not every PoP (Point of Presence). The other routers in the PoP are clients of the RRs in this PoP. The RRs provide the only reachability to other PoPs. The RRs run full-mesh iBGP with each other, often over L2 paths provided for instance by MPLS. ISPs set IGP metrics so that intra-PoP distances are always shorter than inter-PoP distances. Altogether, this effectively constrains the physical topology to match the control topology. The result is that both topology-based and MED-based oscillations are avoided [28]. This arrangement avoids most path inefficiencies: if needed, selected iBGP peering between clients within a PoP, or the announcement of additional routes, can 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. To date, however, no such fundamental solution has been proposed that doesn't continue to restrict RR placement. In particular, [20] is to our knowledge the only previously known non-centralized solution that solves *both* topology-based and MED-based oscillations for *any* RR placement. This approach, however, requires that, in making the best-path selection, minimizing logical iBGP hops be given precedence over minimizing physical IGP metric. To avoid the path inefficiency that this produces, RRs must still be placed more-or-less as they are today in order to maintain the close coupling between the logical iBGP topology and the physical topology. To be clear, [20] does free up RR placement from the perspective of correctness. In this sense, [20] is more forgiving to configuration errors that might otherwise inadvertently produce correctness problems in a traditional RR deployment. Nevertheless, [20] does not free up RR placement from the perspective of path efficiency, which is obviously an important consideration for ISPs.

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

1. is non-centralized in that it divides work among RRs,
2. scales comparably to traditional Route Reflection (herein called TBRR, for Topology-Based Route Reflection),
3. solves all iBGP correctness problems, including topology-based oscillations, MED-based oscillations, and loops,

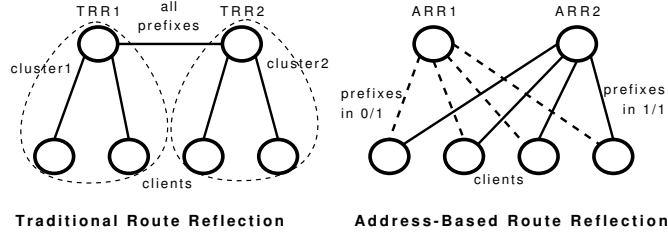


Figure 1: With traditional Route Reflection, routers are assigned to RRs. With Address-Based Route Reflection, address ranges are assigned to RRs. The lines denote iBGP peering sessions.

4. has no path inefficiencies, and
5. 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 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 is shown on the left. Route reflectors (TRR) are associated with different routers, forming groups called clusters. iBGP peerings are required between clients and TRRs in a cluster, and between TRRs (full-mesh). ABRR is shown on the right. Here, route reflectors (ARR) are associated with different address ranges. All client routers iBGP-peer with all ARR, but only advertise prefixes that fall within the address range of the ARR. Each ARR, in its role as an ARR, computes best-paths 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 iBGP as defined in the add-paths draft [40]. Changes to BGP implementations are minimal: we implemented ABRR with less than 2000 lines of new code in the Quagga BGP [6].

Compared to TBRR, ABRR reduces the number of iBGP hops between edge routers from three to two. This leads to the following property:

From the perspective of any given prefix, ABRR is a centralized approach.

In other words, by merely changing the division of labor among RRs from router-based to address-based, we change iBGP route distribution from a distributed approach to a centralized approach without sacrificing scalability. This property is crucial, because a centralized approach is much easier to solve. [21] shows that topology-based oscillations can only occur between RRs, so this problem simply disappears with ABRR.

Our overall approach is to emulate the semantics of full-mesh iBGP:

Full-mesh iBGP has efficient paths and does not suffer from oscillations, so by emulating full-mesh iBGP ABRR adopts these characteristics.

This emulation is accomplished by having ARRs advertise multiple routes — namely, the set of routes that tie for best on AS-level path decision metrics. This 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, scalability 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 motivated to move away from hierarchical edge-core topologies and towards “flat” full-mesh tunneled topologies, where every edge router has a virtual link (usually MPLS) with every other [32]. The need for flat topologies is driven not just by Virtual Private Network (VPN) services, but also by features like traffic engineering and fast-reroute. In hierarchical topologies, there is a convenient alignment between the goals of scaling the IGP (for instance through OSPF areas) and of safely placing RRs. In flat topologies, this convenience disappears. Router vendors are already competing in terms of providing larger flat IGPs, and some ISPs run parallel networks: flat for VPN services, hierarchical for global routing services [32]. It seems entirely possible that ISPs would appreciate the freedom of placing RRs anywhere without having to worry about, for instance, the effect of a flat IGP on RR correctness or path efficiency. This placement could be within existing routers, or pure control plane RRs based on general-purpose computers (either a small number of big machines, or a large number of small machines). Whether this turns out to be of value or not remains to be seen. ABRR, at least, adds RR placement freedom to the set of options available to ISPs. Altogether, this paper makes the following contributions:

- It presents what is to our knowledge the first non-centralized iBGP solution that removes all limitations on RR placement for both *iBGP correctness* and *iBGP path-efficiency*.
- It provides a static performance analysis for ABRR and TBRR, and compares their results.
- It gives the evaluation results of an apples-to-apples comparison of the fully functional ABRR implementation with TBRR and full-mesh iBGP.

In spite of these contributions, these results must be considered preliminary because we don’t yet have dynamic analyses based on measurements from real ISPs. Nevertheless, we are cautiously optimistic that ABRR is a positive alternative to TBRR.

Outline. The next section gives the design of ABRR, explains why it emulates the semantics of full-mesh iBGP, and discusses various practical issues. Section 3 presents a comparative analysis of the scaling characteristics of ABRR and TBRR. Section 4 describes our implementation of ABRR, and compares its performance with standard implementations of full-mesh and TBRR. We give an overview of related work in Section 5, and conclude in Section 6.

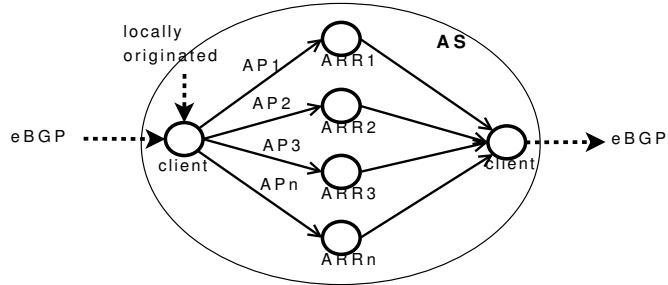


Figure 2: Links represent iBGP sessions. The terms ARR and client refer to router roles: the same physical router may be both a client and an ARR.

2 Design

In most respects, ABRR is similar to traditional Topology-Based Route Reflection (TBRR). As with TBRR, routers are either Route Reflectors (ARR) or clients of ARR. Clients are the source and sinks of iBGP updates, and ARR 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 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 may be responsible for $0/8$, ARR_2 for $1/8$, and so on. Each such range is called an Address Partition (AP), and different APs can have overlaps. Strictly speaking some other criteria may be used, as long as the criteria is consistent for all clients. An example might be “all routes that originate from ASes 0 - 200”.

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 ARR are responsible for the destination prefix in the route. If a prefix spans multiple APs, then the associated route is advertised to the ARR for all such APs. As with full-mesh and TBRR, clients never advertise iBGP-learned routes over iBGP. Clients advertise their best paths to their eBGP neighbors.

The terms ARR and client 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 messages. The ARR function is to reflect received iBGP messages. 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 — it has both ARR and client functionality, and logically passes messages between these functional components. If an ARR is a pure control-plane device (i.e., does not forward IP packets), 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 example, imagine two routers R1 and R2 that are both ARRs (implied here is that they are ARRs for the same AP). When router

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

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

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

R1 receives an eBGP update for a destination prefix within the AP, if that prefix is its best path it forwards the update to R2. In so doing it is acting in its role as a client, and so we can say “client R1 forwards the update to ARR R2”. Logically the client R1 also forwards the update to the ARR function within R1.

The only time two routers are iBGP peers (i.e., have an iBGP session) is when one router is a client and the other is an ARR. Define a *pure client* as a router that is not an ARR for any prefix. Two pure clients are not iBGP peers. Likewise define a pure control-plane ARR (i.e., one that has no client function) as a *pure ARR*. Two pure ARRs are not iBGP peers. In a deployment where a relatively small fraction of routers serve as ARRs, most routers are pure clients, and each pure client has a relatively small number of iBGP peers. By contrast in a deployment where most data-plane routers are ARRs, the number of iBGP sessions approaches that of full-mesh. Each session, however, carries fewer routes than with full-mesh.

Each client advertises its best eBGP-learned or locally-originated route to all ARRs responsible for the AP containing the route (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 to all clients. Specifically it selects all routes that remain after steps one through four of the best-path selection steps in Table 2 from RFC4271 [34]. 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 (Table 1). The routes are not advertised to other ARRs since this information would be redundant. The best AS-level routes may be encoded using the existing add-paths mechanism in [40].

The effect of adding these additional best AS-level routes in ABRR is that the semantics of full-mesh iBGP are

emulated at least in the steady state. Because of this, ABRR adopts the behavioral characteristics of full-mesh iBGP: that is, it has no oscillations, and no path inefficiencies.

2.1 ABRR Emulates the Semantics of Full-mesh

To see why ABRR emulates the semantics of full-mesh, we can compare the operation with 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 path for the client/router is other-learned, then it will advertise that into iBGP. Otherwise, if the best path 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 ARR, 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 will in turn 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 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.

2.2 ABRR Has No Routing Anomalies

Note that our use of best AS-level routes is not novel: Basu *et al.* [9] used them to prevent MED-based oscillations (but cannot prevent topology-based oscillations [20]). There are some differences, however. With Basu, all routers advertise all best AS-level routes, while in ABRR only ARRs advertise all best AS-level routes — clients only advertise their best route (if not iBGP-learned). Nevertheless, the use of best AS-level routes allows ABRR to emulate full-mesh iBGP, and so for this reason also contributes to preventing MED-based oscillations in ABRR. Without this, ABRR would have MED-based oscillations. Indeed in an early version of ABRR that did not use best AS-level routes, our RouteViews-driven experimental testbed exhibited MED-based oscillations.

The use of best AS-level routes in ABRR, however, also prevents path inefficiencies. The reasoning is simple: ABRR emulates full-mesh, and full-mesh has no path inefficiencies, therefore ABRR has no path inefficiencies. 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. Of course, 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 should be sufficient.

The fact that full-mesh has no topology-based oscillations also means that ABRR has no topology-based oscillations. This conclusion is separately supported by Flavel *et al.* [21], which shows that topology-based oscillations can only occur between RRs. Since with ABRR iBGP routes pass through only a single RR, ABRR has no topology-based oscillations.

2.3 Preventing iBGP Message 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-received updates on iBGP. ARRs only send iBGP-received 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. In addition, B thinks A is an ARR, C thinks B is an ARR, and A thinks C is an ARR. However, A thinks B is a client, B thinks C is a client, and C thinks that A is a client. A BGP update in this configuration would loop around A to B to C to A. Of course, a single looping update would be recognized as old news and not continue beyond the first round, but two different updates for the same prefix could chase each other around the loop indefinitely.

Either loop-detection mechanism used by route reflectors today, the Cluster List or the Originator ID [10], 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.

2.4 Additional Configuration

One of the reasons that ISPs like TBRR is that it simplifies iBGP configuration. When a new client is added, only the client and its RRs need to be configured for the iBGP peering relationship. By contrast, in ABRR adding a new client requires that iBGP peering between the client and all ARRs be configured.

From a technical standpoint, auto-configuring iBGP peers is not complicated. It could be done through a centralized management system, or through the IGP, for instance as has been proposed in the IETF [31]. This protocol could easily be extended to allow routers to not only announce their presence as a BGP peer within OSPF, but also to announce for which APs they are an ARR. While we recognize that this is an additional burden on the ISP, and therefore an obstacle to acceptance by ISPs, we nevertheless don't feel that this represents a difficult technical challenge per se.

2.5 Policies

BGP implementations have a rich set of policy controls. Many policies are best done at border routers, for instance filtering customer prefixes or tagging customer routes incoming, or setting MED values outgoing. In the case of TBRR, other policies, such as prefix aggregation, may be executed at the TRR. For instance, [11] even suggests that TRRs would be a good place to manage traffic engineering.

Since ABRR emulates full-mesh, any policies that can be implemented in full-mesh iBGP can be implemented in ABRR. In terms of executing policies in the RR, however, ABRR is more restrictive than TBRR. An ARR (acting in its role as an ARR) can only reliably execute policies that act on a single prefix. Policies that act on multiple prefixes may or may not work at an individual ARR, depending on whether the prefixes fall outside the ARR's set of APs. These would have to be implemented at border routers (clients) instead. For instance, an ARR would not be able to execute aggregation policies on prefixes that fall outside of the ARR's APs. As another example, the advertise-map policy feature allows a router to apply actions on routes to a given prefix based on the presence or absence of routes to other prefixes (though note that the use of this policy feature is already limited to outgoing eBGP messages).

Speaking theoretically, there could be policies that can be implemented in ABRR that cannot be easily implemented in TBRR. These would be policies that apply to a specific pair of border routers. For instance, in full-mesh, a conceivable policy would be for a border router to tag routes one way when sent to one border router, and another way when sent to another. A policy like this could be executed at an ARR, since it knows which updates are sent to and received from which clients.

There could also be policies that can be implemented in TRRs but cannot be as easily implemented in ARRs. These would be any policies that are defined for a cluster (or, equivalently in many cases, a PoP) as a whole.

To implement these policies in an ARR would require defining which clients belong to which policy groups. An example of this might be prefix filtering for all customers at a given PoP. A policy like this might be implemented using a peer group defined across all clients. Note that we are not aware of how often such policies are defined in practice, or how difficult it would be to migrate the policies to the clients.

To conclude, compared to TBRR, ABRR doesn't impose any policy restrictions from a functional point of view. ABRR may, however, change the way in which ISPs deploy policies. More research is required to determine how substantial these changes might be.

3 Static Performance Analysis

This section provides a comparative static analysis of ABRR and TBRR. The goal here is not to show how a specific ISP with specific routers would scale, but rather to identify the major components of performance, and to show how the approaches behave comparatively with respect to these components. A large body of previous work has indicated that the scaling problem of iBGP largely relates to RIB size [12, 15, 19, 24, 25, 29, 30], so our analysis focuses on two parameters, the number of entries in the RIB, and the number of entries in the RIB-Out. Our definition of RIB consists of the best route reported by every BGP neighbor. This RIB reflects all entries that realistically need to be stored by BGP. Strictly speaking, a router may choose to remove some RIB entries, at the expense of requesting a refresh from BGP neighbors when the best route becomes unavailable [24], but we don't assume this.

Our definition of RIB-Out consists of all entries reported to each BGP neighbor, and thus corresponds to the conceptual Adj-RIB-Out defined by BGP [34]. In practice, routers typically do not keep a "full" separate RIB-Out data structure, but rather efficiently maintain the necessary information with for instance pointers to the RIB. As such, the RIB-Out does not typically consume much in the way of memory. Rather, we include it in our analysis because the number of entries in the RIB-Out provides a rough comparative indication of message transmission and processing overhead for outgoing updates — every entry in the RIB-Out will have been transmitted to the associated BGP neighbor.

3.1 Best AS-Level Routes

As described in Section 2, ARRs transmit all best AS-level routes to clients. The scalability of ABRR therefore depends directly on how many best AS-level routes there are. There are two main causes for multiple best AS-level routes. The first cause is multiple best AS paths to remote ASes. To estimate this, we measured the number of best AS-level routes from the `route-views2`'s IPv4 RIB dump which consists of full BGP tables for 42 eBGP neighbors connected to 38 distinct ASes, collected on Dec 3, 2009 at 12:00am UTC [7]. These are roughly equivalent to the AS paths that would be seen by a transit ISP from its peers, or by a non-transit ISP from its transit providers. Figure 3 gives the average number of best AS-level routes per prefix for between 1 and 38 randomly selected distinct neighbor ASes, from `route-views2`.

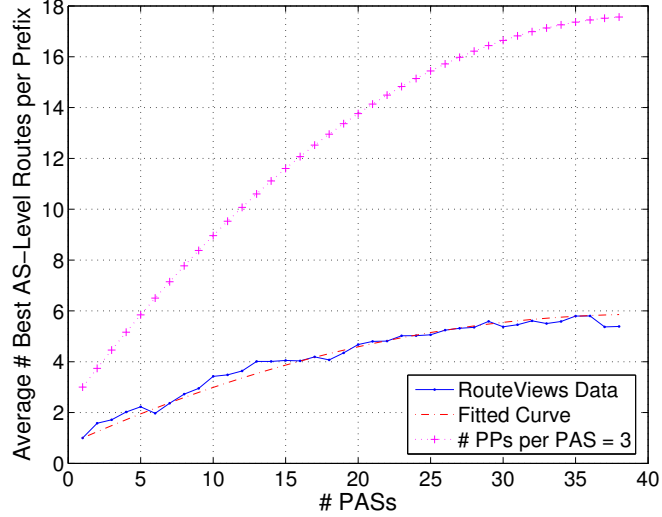


Figure 3: Best AS-Level Routes per Prefix

The number of best AS-level routes increases with the number of different “views” into the Internet. However, the rate of increase flattens out, presumably as the true total number of best AS-level paths is approached. Using least squares, we fitted a curve to these values (Figure 3), denoted as $\mathcal{F}(\#PASs)$, which is used in the remainder of our analysis. Here, *PAS* denotes Peer/Provider AS.

The second cause for multiple best AS-level routes is multiple peering points between neighbor ASes. Specifically, we compute the average number of best AS-level routes per prefix as $\#BAL = \mathcal{F}(\#PASs) \times \#PPs$, where $\#PPs$ is the average number of peering points between a given AS and its neighbor PAS. To estimate $\#PPs$, we used the iPlane [26] dataset covering 14849 ASes collected on June 27, 2010 [5]. From this, we computed the average number of peering points with each neighbor AS as 1.501. Unfortunately this average includes neighbor ASes that are customers where often there is a single peering point (45.21% of AS pairs from our dataset), and so underestimates the number of peering points with neighbor PASs. Noting that AT&T’s peering policy mandates that there be at least three peering points [1], we adopt “ $\#PPs$ per PAS = 3” as a conservative estimate. Figure 3 also shows $\#BAL$ for “ $\#PPs$ per PAS = 3”.

3.2 RIB and RIB-Out Analysis

In our analysis, we wish to derive RIB size and RIB-Out size 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 ARR/TRRs
- *#PASs*: the total number of peer/provider ASes (from which we derive $\#BAL$ as already stated)

Table 3: Prefix Density Distribution

AP	# Prefixes	AP	# Prefixes
0.0.0.0/3	6952	128.0.0.0/3	16469
32.0.0.0/3	17531	160.0.0.0/3	25768
64.0.0.0/3	81717	192.0.0.0/3	137871
96.0.0.0/3	29137	224.0.0.0/3	0

- $\#N$: the total number of routers
- $\#Clients$: $(\#N - \#ARRs)$ or $(\#N - \#TRRs)$

ABRR Analysis. 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 size $\mathcal{S}_{\text{RIB}_{\text{ARR}}}$ is then the sum of the total due to managed routes $\mathcal{S}_{\text{RIB}_{\text{ARR}}}^m$ and the total due to unmanaged routes $\mathcal{S}_{\text{RIB}_{\text{ARR}}}^u$:

$$\mathcal{S}_{\text{RIB}_{\text{ARR}}} = \mathcal{S}_{\text{RIB}_{\text{ARR}}}^m + \mathcal{S}_{\text{RIB}_{\text{ARR}}}^u$$

We make the simplifying assumption that an ARR (or TRR for that matter) is never 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. Note that, based on RouteViews trace, the prefixes are actually not evenly distributed, as shown in Table 3¹. This is completely controllable. In reality, an ISP could select APs that have roughly equal numbers of routable prefixes. Therefore, for managed routes, each AP contains $\frac{\#Prefixes}{\#APs}$ prefixes on average, and each prefix contributes $\#BAL$ best AS-level routes:

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

The number of redundant ARRs for any given AP is $\frac{\#ARRs}{\#APs}$. For its unmanaged routes, each ARR in its role as a client maintains a route to each redundant ARR for each prefix not within its own AP:

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

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 route. This produces the size of RIB-Out:

¹224.0.0.0/3 is reserved for multicast, limited broadcast, and future use [4].

$$\mathcal{S}_{\text{RIB-Out}_{\text{ARR}}} = \mathcal{S}_{\text{RIB}_{\text{ARR}}}^m \times \left(\#N - \frac{\#\text{ARRs}}{\#\text{APs}} - 1 \right)$$

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

$$\mathcal{S}_{\text{RIB}_{\text{TRR}}} = \mathcal{S}_{\text{RIB}_{\text{TRR}}}^m + \mathcal{S}_{\text{RIB}_{\text{TRR}}}^u$$

The behavior of clients in TBRR is similar to that of clients in ABRR in that the clients will 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 number of RIB entries due to managed routes is:

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

The RIB 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 [23, 28]. A TRR will therefore advertise one route per prefix no matter how many best AS-level routes are advertised by its clients. The total number of routes advertised by all TRRs, then, depends on the distribution of peering points (PPs) among clusters, and the distribution of best AS-level routes among PPs. For instance, to take one extreme, if all PPs are in a single cluster, then the TRRs for that cluster will advertise one route per prefix, and all other TRRs will withdraw or not advertise any routes. On the other extreme, if PPs are uniformly distributed among the clusters, and best AS-level routes are uniformly distributed among PPs, then unless there are more best AS-level routes than clusters, all best AS-level routes will be advertised collectively by TRRs. In between these two extremes would be a random distribution of PPs and best AS-level routes. Here, for a given prefix, some clusters may have no best AS-level routes to advertise, and other clusters may have multiple best AS-level routes, only one of which is advertised.

While real ISPs of course exhibit neither pure uniform nor pure random distribution of PPs and best AS-level routes, we suspect that they are closer to uniform distribution than to random distribution. This is because ISPs are motivated to provide short paths to exit points, and therefore distribute those exit points evenly across their topologies. Indeed, AT&T's peering policy mandates that the multiple peering points be geographically diverse [1]. We therefore make the simplifying assumption that PPs are uniformly distributed among the clusters, and best AS-level routes are uniformly distributed among PPs. Note that this maximizes the number of unmanaged RIB entries, and so from the narrow perspective of RIB size alone makes TBRR look worse than it would under a different assumption. Nevertheless we believe that this is the trade-off ISPs would make, given that the larger RIB size results in shorter intra-ISP paths.

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 other TRRs as:

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

For clarity, we abbreviate $\mathcal{G}(\#BAL, \#Clusters, \#Prefixes)$ as $\mathcal{G}(\cdot)$. A TRR will receive $\mathcal{G}(\cdot)$ routes from every other TRR. Therefore, the size of a TRR's RIB from unmanaged routes is:

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

Regarding TRR's RIB-Out size $\mathcal{S}_{\text{RIB-Out}_{\text{TRR}}}$, 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 $\mathcal{G}(\cdot)$. These are advertised to all other TRRs, as well as to all clients except the client from which the route was learned. This is captured in the first term in the equation below. Second are the remaining $(\#Prefixes - \mathcal{G}(\cdot))$ best routes which are learned from other TRRs and advertised to all clients. These are captured in the second term in the equation below.

$$\begin{aligned} \mathcal{S}_{\text{RIB-Out}_{\text{TRR}}} &= \mathcal{G}(\cdot) \times \left(\#TRRs + \frac{\#Clients}{\#Clusters} - 2 \right) \\ &+ (\#Prefixes - \mathcal{G}(\cdot)) \times \frac{\#Clients}{\#Clusters} \end{aligned}$$

Additional Paths in TBRR. The above analysis for TBRR assumes that TRRs advertise a single best path. In reality, there is an increasing interest among ISPs to advertise additional paths. For instance, it is a common feature in routers that clients advertise their best external path whether or not that path is the best path [27]. This improves failure response time as well as reduces the chances of RIB-based oscillations. Besides this, mechanisms for advertising additional paths in BGP are being advanced in the IETF [40], though industry has by no means settled on how best to use these additional paths [32].

A comparison between ABRR and single-path TBRR is fair in the sense that single-path TBRR is predominantly how ISPs deploy TBRR today (with the possible exception of best external path). The comparison is unfair, however, in the sense that ABRR provides multiple paths that may (or may not) 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 multi-path version of TBRR using add-paths capability [40]. For this reason, we provide a second analysis of TBRR, one that like ABRR maintains and advertises all best AS-level routes, called T-BAL mechanism.

The analysis of RIB and RIB-Out size in this case is quite similar to that for single-path TBRR above, with the exception that the first term in the function $\mathcal{G}(\cdot)$ 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 for RIB and RIB-Out size for multi-path TBRR (T-BAL) is as follows:

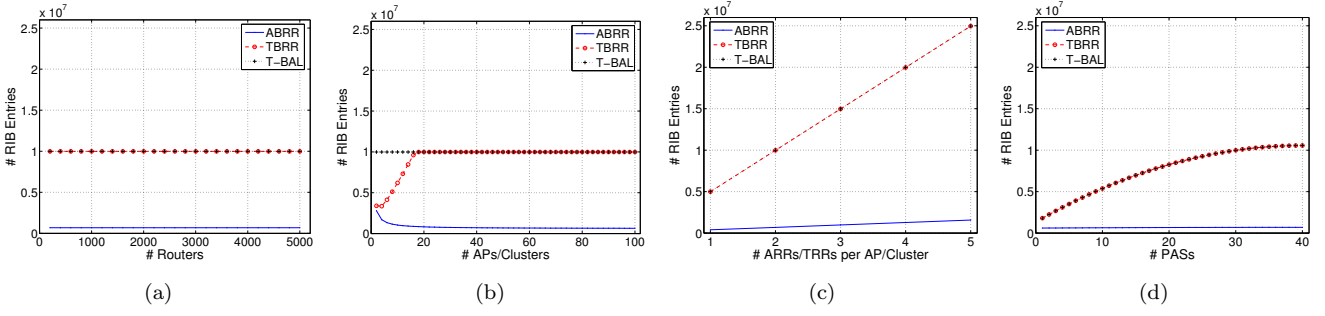


Figure 4: # RIB Entries of ARRs/TRRs (serves primarily as an indicator of memory load). Default values are 2000 routers, 50 APs/Clusters, 2 ARRs/TRRs per AP/Cluster, 30 PASs, and 3 PPs per PAS.

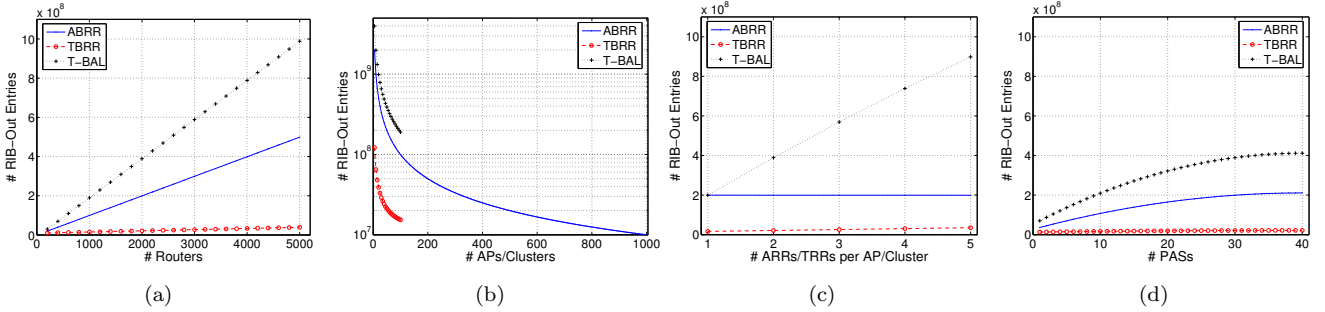


Figure 5: # RIB-Out Entries of ARRs/TRRs (serves primarily as an indicator of outgoing update bandwidth and processing load). Default values are 2000 routers, 50 APs/Clusters, 2 ARRs/TRRs per AP/Cluster, 30 PASs, and 3 PPs per PAS. Note that Figure (b) is log scale.

$$\begin{aligned}
 \mathcal{S}_{\text{RIB-T-BAL}}^m &= \mathcal{S}_{\text{RIB-TRR}}^m = \frac{\#BAL \times \#Prefixes}{\#Clusters} \\
 \mathcal{S}_{\text{RIB-T-BAL}}^u &= \mathcal{S}_{\text{RIB-T-BAL}}^m \times (\#TRRs - 1) \\
 \mathcal{S}_{\text{RIB-T-BAL}} &= \mathcal{S}_{\text{RIB-T-BAL}}^m + \mathcal{S}_{\text{RIB-T-BAL}}^u \\
 \mathcal{S}_{\text{RIB-Out-T-BAL}} &= \mathcal{S}_{\text{RIB-T-BAL}}^m \times \left(\#TRRs + \frac{\#Clients}{\#Clusters} - 2 \right) \\
 &\quad + \mathcal{S}_{\text{RIB-T-BAL}}^u \times \frac{\#Clients}{\#Clusters}
 \end{aligned}$$

Analytical Results. Figures 4 and 5 plot the above equations for RIB 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 PASs. The default setting of these parameters, taken to model a typical large-scale AS [36], are 2000 routers, 30 PASs (3 PPs per PAS), and 50 APs/clusters each is managed by 2 ARRs/TRRs. Each figure varies one of the parameters while holding the others constant at their default values. We assume 300,000 unique prefixes [2].

The primary takeaways from these figures are as follows. In terms of memory (i.e., RIB size), ABRR is more efficient than TBRR. In terms of message and processing load (i.e., RIB-Out size), normal single-path TBRR is generally more efficient than ABRR, but if ABRR is configured with enough APs, then it matches TBRR’s efficiency (see discussion below). For multi-path TBRR (T-BAL), however, ABRR is more efficient in all respects.

From Figure 4, we see that the parameter that most influences the number of RIB entries in ABRR is the number of ARRs per AP — 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 and RIB-Out size (Figures 4b and 5b respectively). The benefit on RIB size from increasing the number of APs quickly reaches diminishing returns. The number of RIB entries is dominated by the need for ARRs 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 other routers, the size of the RIB-Out, and therefore the corresponding processing load, can be steadily reduced by increasing the number of APs. This is emphasized through the log-scale for Figure 5b. 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 PoPs. In Figure 5b, this fact is emphasized by truncating the curves for TBRR at 100 clusters. Overall, a large ISP could for this reason reduce RIB-Out on ARRs well below that of TRRs, and certainly well below that of full-mesh, which lacks any mechanism for reducing RIB-Out.

On the other hand, increasing the number of APs also increases the number of ARRs, and therefore the number of iBGP sessions that client routers must maintain. This effect is not seen in TBRR, since clients only peer with the TRRs in their cluster, and therefore the number of clusters does not matter for clients. More iBGP sessions increase load on routers as well as increase configuration burden. Regarding the former, routers today have limitations on the number of iBGP sessions they can maintain. These limitations could well exist, however, because TBRR eliminates the need for router vendors to engineer for a large number of iBGP sessions. It should be the case that maintaining a large number of (low volume) sessions is technically not that difficult. Regarding the latter, the configuration burden could be mitigated by the automatic iBGP configuration mechanism described in Section 2.4, but only to an extent. The fact that this technology has not been broadly adopted could also simply reflect the fact that TBRR minimizes the need for it.

4 Implementation and Evaluation

This section describes our implementation of ABRR, and gives experimental results comparing ABRR against TBRR and full-mesh for a 30-router testbed using RouteViews traces.

4.1 Experimental Setup

We implemented ABRR with less than 2000 lines of C code on the version 0.99.16 code base of Quagga [6], an open source routing software package with support for OSPF and BGP among others. We deployed ABRR on a testbed consisting of 30 machines running Linux 2.6.30. Though relatively small, this testbed nevertheless allowed us to thoroughly exercise our implementation, do apples-to-apples comparisons of ABRR with both TBRR and full-mesh, and verify the analytical results from section 3.

Routing updates for our testbed were derived from two RouteViews [7] IPv4 RIB dumps² (`route-views2` and `route-views4`, respectively) on Dec 3, 2009 at 12:00am UTC. The `route-views2` dump produced 315,445 unique RIB entries from 42 eBGP neighbors connected to 38 distinct ASes, while the `route-views4` dump produced 306,450 unique RIB entries from 7 eBGP neighbors connected to 7 distinct ASes. We developed a simple pseudo BGP speaker, called *route regenerator*, which uses the RouteViews RIB dumps to generate live BGP feeds. It is important to note that the route regenerator is not a full BGP speaker. It does not contain the necessary FSM, and never sends withdraws. Therefore in our experiments, there is always a RIB entry for each prefix sent by an eBGP neighbor.

In ABRR, APs are the unit of division. In TBRR, clusters are the unit of division. Therefore, to compare ABRR and TBRR apples-to-apples, we assign the same number of APs in the ABRR experiment as clusters in the corresponding TBRR experiment. Likewise, both APs and clusters can have redundant RRs. Again, to make our comparisons apples-to-apples, we assign the same number of ARRs/TRRs to APs/clusters.

In our experiments, we independently vary the number of APs/clusters (default 4), the number of ARRs/TRRs per AP/cluster (default 2), and the number of peer/provider border routers³ (PBRs) in the AS⁴ (default 7). This allows us to see the impact of each of these factors on scalability. Within an experiment, each AP has the same-size address range, and clusters have the same number of routers. PBRs peer with one eBGP neighbor each.

The `route-views4` dump was used for the experiments where the numbers of APs/Clusters and ARRs/TRRs per AP/Cluster were measured. This allowed us to have relatively fewer PBRs, and therefore more routers to use as ARRs and TRRs. The `route-views2` was used for the experiments where the number of PBRs was varied, because it allows us to work with a larger number of PBRs.

As in Section 3, we use RIB and RIB-Out size as our performance metrics. The RIB measurements reflect the case where only one entry per peer per prefix is stored.

4.2 Experimental Results

The first thing to notice about Figures 6, 7, and 8 is that the measurements for ARR/TRR RIB and RIB-Out sizes (all of the (a) and (c) sub-figures) match very well against our analytical numbers in Figures 4 and 5.

²We filtered out the IPv6 part of the dump files used in our experiments.

³“Peer/provider border routers (PBRs)” denote the border routers in an AS which connect to peer/provider ASes (PASs).

⁴Note that, in the evaluation we vary the number of PBRs in the AS, instead of the number of PASs used in the analysis.

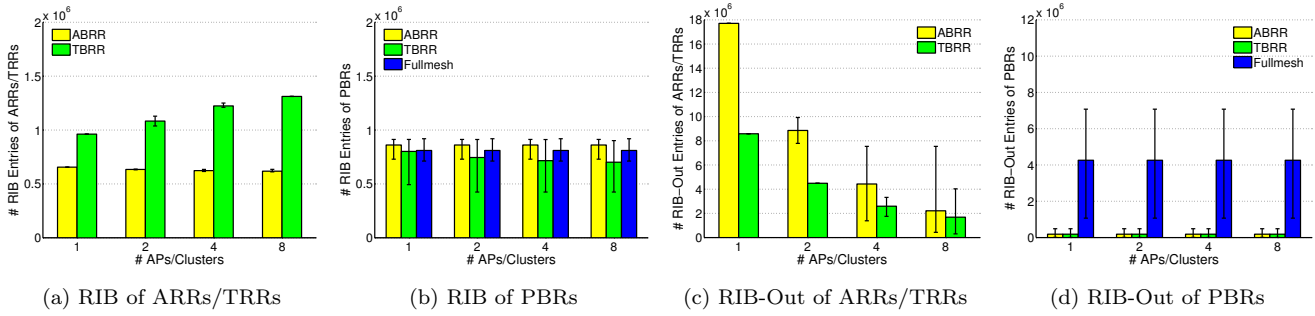


Figure 6: Comparison under Different Numbers of APs/Clusters (7 PBRs and 2 ARR/TRRs per AP/Cluster)

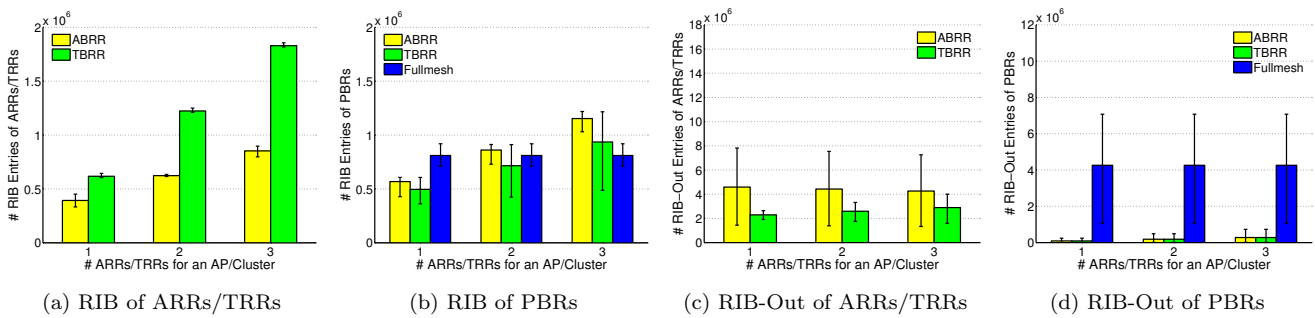


Figure 7: Comparison under Different Numbers of ARR/TRRs for an AP/Cluster (4 APs/Clusters and 7 PBRs)

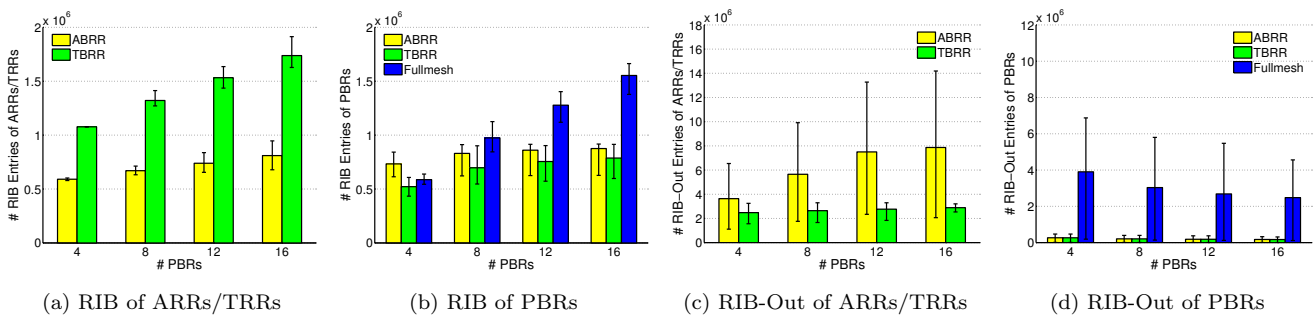


Figure 8: Comparison under Different Numbers of PBRs (4 APs/Clusters and 2 ARR/TRRs per AP/Cluster)

Therefore, the conclusions reached in Section 3 apply directly here. Namely, the performance of ARR and TRR is roughly comparable. The RIB of ARR is smaller than that of TRR, which indicates that ARR scale better in terms of memory consumption. The RIB-Out of ARR is larger than that of TRR, which in principal means that TRR scale better in terms of message and processing load; however, this difference can be eliminated with increased number of APs.

One thing that our measurement numbers capture that the analysis doesn't is variation between individual routers. Note that the major bars represent averages, while the "error bars" represent the max and min values across all routers. Overall, there are two factors that contribute to variation in RIB size. One is the variation in the number of routable prefixes per AP. This is seen in the large max-min bars in Figures 6c, 7c and 8c. Note that this variation is completely controllable. For our experiments, we selected equal-sized address ranges in terms of number of addresses. In reality, an ISP could select APs that have roughly equal numbers of routable prefixes. The second contributor to variation is the difference in the number of routes advertised by each peer. This can best be seen in Figures 6d, 7d, and 8d, where the max-min bars for the full-mesh case are quite large.

The second thing that the measurement results capture that the analysis doesn't is the RIB and RIB-Out of PBRs. For PBRs in ABRR and TBRR, the RIB is coming from the ARR/TRR and from the external peer. In Figure 7b, we see that the RIB size of PBRs increases because the number of ARR and TRR increases. Actually, in Figure 6b, the RIB size for TBRR PBRs shrinks slightly with an increase in the number of clusters. What is happening here is that since the number of PBRs across the whole network is held steady, the number of PBRs inside a cluster is decreasing, and the chance that a PBR's eBGP-learned route is the best route increases. This means that more best routes will be through oneself, and therefore there will be more withdraws received from the TRR, which leads to the lower RIB size of TBRR PBRs.

For full-mesh, we see from Figure 8b that the number of PBRs has a significant effect on the RIB size. This is because as the number of available routes from other PBRs increases, the number of best routes through iBGP increases, therefore the size of the RIB increases. By the same token, the RIB-Out for full-mesh PBRs decreases (Figure 8d), because the PBR withdraws more and more routes as it finds better alternatives via iBGP. Note that the default number of PBRs in the experiments of Figures 6 and 7 is 7. From Figure 8b we see that the case of 8 PBRs puts the RIB size of full-mesh PBRs at very near the RIB size of the ABRR and TBRR PBRs. Thus it is a coincidence from our default parameter selection that the RIB size all PBRs in Figure 6b is roughly the same.

The RIB-Out size of ABRR and TBRR PBRs (Figure 6d) is significantly less than the RIB-Out size of ARR and TRR (Figure 6c). This is a simple reflection of the fact that ARR advertise routes to all clients (for its APs), and TRR transmit routes to all other TRR and their clients. By contrast, PBRs only transmit routes to ARR and TRR. By the same token, the RIB-Out size of full-mesh PBRs is much bigger than either ABRR or TBRR PBRs (Figure 6d), and also significantly bigger than the RIB-Out size of ARR and TRR in the case of 8 APs/clusters (Figure 6c). This is a simple reflection of the fact that full-mesh PBRs transmit routes to all other routers including PBRs.

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 [3, 16] and then extensively studied [9, 22, 23, 28, 35].

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 [23], 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, [39] suggested that these two conditions are too restrictive for designing correct iBGP topologies. Moreover, in [20], 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 [39], 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 [33], 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 [17], 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 limit placement of RRs.

Other approaches aim to solve route reflection problems at the protocol level. In [9], each router advertises all best AS-level routes to all of its iBGP neighbors. This approach solves MED-based oscillations, but not topology-based oscillations [20]. Similar to [9], ABRR also advertises the best AS-level routes. ABRR, however, solves all oscillation problems. In [20], all oscillation problems are solved by introducing a new step in the BGP best-path decision process: comparing the number of iBGP hops. 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 [20], ABRR has no restrictions on RR placement even with respect to path efficiency.

The Route Control Platform (RCP) [14, 18] takes a very different approach to the iBGP problem. Rather than full-mesh, routing updates are sent to a single centralized control-plane BGP speaker (the RCP) which makes best-path decisions and feeds these back to routers. These decisions take into account 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 of the basic RCP idea [38] goes even further, centralizing eBGP peering and all policy at the RCP, thus simplifying policy configuration and reducing policy-based errors. We find this argument in most respects to be quite compelling. However, RCP is a substantial departure from how ISPs operate today; contrastively, ABRR could 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.

Virtual Aggregation (ViAggre) aims to improve utilization of hardware memory in routers by shrinking the forwarding information base (FIB) [8]. It does this by splitting the address space into virtual prefixes, and allowing different routers to be responsible for routing for different virtual prefixes. ABRR is reminiscent of ViAggre in that it uses address partitions to divide and conquer scaling issues.

6 Conclusion

This paper presents Address-Based Route Reflection (ABRR), a variant on traditional topology-based route reflection (TBRR) that solves all known oscillation problems and finds efficient paths while placing no constraints on RR placement. Through static analysis, and a small-scale testbed that validates our analysis, this paper provides preliminary evidence that ABRR's scalability is comparable to that of TBRR. The comparison is rough: broadly stated, ABRR scales better in terms of memory, while TBRR scales better in terms of message and processing load. However, ABRR deployment has more degrees of freedom due in large part to no constraints on RR placement. This flexibility can often be exploited to decrease ABRR message and processing load, but at the expense of additional iBGP peering sessions. Finally, a comparison of ABRR and TBRR is in any event apples-to-oranges in that ABRR by default provides more paths — as many paths as full-mesh iBGP provides. When these paths are provided by TBRR (i.e., through the add-paths mechanism [40]), ABRR scales better across the board.

In order to provide definitive evidence (for or against) of ABRR's scalability compared to TBRR, we must measure the static and dynamic performance of ABRR and TBRR for real ISPs. Towards this end, as future work, we would like to do the following: 1) Obtain ISP BGP tables and router topologies, and use them to do detailed static analysis of RIB size for a variety of ABRR deployments. 2) Obtain BGP traces from these ISPs, and use them as workflows on a larger-scale deployment of our implementation to measure the system dynamics and message/processing overhead. 3) Obtain BGP policy configuration from these ISPs, and determine how and to what extent this has to be changed to accommodate ABRR.

References

- [1] AT&T Global IP Network Peering Policy. <http://www.corp.att.com/peering/>.
- [2] BGP Routing Table Analysis Reports. <http://bgp.potaroo.net/>.
- [3] Cisco Field Notice: Endless BGP Convergence Problem in Cisco IOS Software Releases. <http://www.cisco.com/en/US/ts/fn/100/fn12942.html>.
- [4] IANA IPv4 Address Space Registry. <http://www.iana.org/assignments/ipv4-address-space/ipv4-address-space.xml>.
- [5] iPlane Datasets. <http://iplane.cs.washington.edu/data.html>.
- [6] Quagga Software Routing Suite. <http://www.quagga.net/>.

- [7] Route Views Project Page. <http://www.routeviews.org/>.
- [8] H. Ballani, P. Francis, T. Cao, and J. Wang. Making Routers Last Longer with ViAggre. In *NSDI*, 2009.
- [9] 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*, pages 235–247, 2002.
- [10] T. Bates, E. Chen, and R. Chandra. BGP Route Reflection: An Alternative to Full Mesh Internal BGP (IBGP). RFC 4456, Apr. 2006.
- [11] O. Bonaventure, S. Uhlig, and B. Quoitin. The case for more versatile BGP Route Reflectors. IETF Internet-Draft draft-bonaventure-bgp-route-reflectors, July 2004.
- [12] T. Bu, L. Gao, and D. F. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45(1):45–54, 2004.
- [13] M.-O. Buob, S. Uhlig, and M. Meulle. Designing Optimal iBGP Route-Reflection Topologies. In *Networking*, pages 542–553, 2008.
- [14] 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.
- [15] 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*, pages 203–208, 2002.
- [16] R. Dube. A comparison of scaling techniques for BGP. *Computer Communication Review*, 29(3):44–46, 1999.
- [17] N. Feamster and H. Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *NSDI*, 2005.
- [18] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe. The Case for Separating Routing from Routers. In *FDNA*, 2004.
- [19] A. Feldmann, L. Cittadini, W. Mühlbauer, R. Bush, and O. Maennel. HAIR: Hierarchical Architecture for Internet Routing. In *ReArch*, 2009.
- [20] A. Flavel and M. Roughan. Stable and flexible iBGP. In *SIGCOMM*, pages 183–194, 2009.
- [21] A. Flavel, M. Roughan, N. Bean, and A. Shaikh. Where’s Waldo? Practical Searches for Stability in iBGP. In *ICNP*, 2008.
- [22] T. Griffin and G. T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *ICNP*, pages 90–99, 2002.
- [23] T. Griffin and G. T. Wilfong. On the correctness of IBGP configuration. In *SIGCOMM*, pages 17–29, 2002.
- [24] E. Karpilovsky and J. Rexford. Using forgetful routing to control BGP table size. In *CoNEXT*, 2006.
- [25] D. Krioukov, K. C. Claffy, K. Fall, and A. Brady. On Compact Routing for the Internet. *Computer Communication Review*, 37(3):41–52, 2007.
- [26] H. V. Madhyastha, T. Isdal, M. Piatek, C. Dixon, T. E. Anderson, A. Krishnamurthy, and A. Venkataramani. iPlane: An Information Plane for Distributed Services. In *OSDI*, pages 367–380, 2006.
- [27] P. Marques, R. Fernando, E. Chen, and P. Mohapatra. Advertisement of the best external route in BGP. IETF Internet-Draft draft-marques-idr-best-external, Mar. 2009.
- [28] D. McPherson, V. Gill, D. Walton, and A. Retana. Border Gateway Protocol (BGP) Persistent Route Oscillation Condition. RFC 3345, Aug. 2002.
- [29] 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.
- [30] D. Meyer, L. Zhang, and K. Fall. Report from the IAB Workshop on Routing and Addressing. RFC 4984, Sept. 2007.

- [31] R. Raszuk. OSPF Extensions for BGP Peer Discovery. IETF Internet-Draft draft-raszuk-ospf-bgp-peer-discovery, June 2003.
- [32] R. Raszuk. To Add-Paths or not to Add-Paths. NANOG48, Feb. 2010.
- [33] A. Rawat and M. A. Shayman. Preventing persistent oscillations and loops in IBGP configuration with route reflection. *Computer Networks*, 50(18):3642–3665, 2006.
- [34] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271, Jan. 2006.
- [35] J. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Netw.*, 13(5):1160–1173, 2005.
- [36] N. T. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with rocketfuel. In *SIGCOMM*, pages 133–145, 2002.
- [37] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 5065, Aug. 2007.
- [38] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. E. van der Merwe. Wrestling Control from BGP: Scalable Fine-Grained Route Control. In *USENIX Annual Technical Conference*, pages 295–308, 2007.
- [39] M. Vutukuru, P. Valiant, S. Kopparty, and H. Balakrishnan. How to Construct a Correct and Scalable iBGP Configuration. In *INFOCOM*, 2006.
- [40] D. Walton, A. Retana, E. Chen, and J. Scudder. Advertisement of Multiple Paths in BGP. IETF Internet-Draft draft-walton-bgp-add-paths, Jan. 2009.
- [41] D. Walton, A. Retana, E. Chen, and J. Scudder. BGP Persistent Route Oscillation Solutions. IETF Internet-Draft draft-walton-bgp-route-oscillation-stop, Nov. 2009.