Design and Implementation of a Routing Control Platform

Matthew Caesar
UC Berkeley

Donald Caldwell
AT&T Labs-Research

Nick Feamster
MIT

Jennifer Rexford
Princeton University

Aman Shaikh
AT&T Labs-Research

Jacobus van der Merwe
AT&T Labs-Research

Abstract

The routers in an Autonomous System (AS) must distribute the information they learn about how to reach external destinations. Unfortunately, today's internal Border Gateway Protocol (iBGP) architectures have serious problems: a "full mesh" iBGP configuration does not scale to large networks and "route reflection" can introduce problems such as protocol oscillations and persistent loops. Instead, we argue that a Routing Control Platform (RCP) should collect information about external destinations and internal topology and select the BGP routes for each router in an AS. RCP is a logically-centralized platform, separate from the IP forwarding plane, that performs route selection on behalf of routers and communicates selected routes to the routers using the unmodified iBGP protocol. RCP provides scalability without sacrificing correctness. In this paper, we present the design and implementation of an RCP prototype on commodity hardware. Using traces of BGP and internal routing data from a Tier-1 backbone, we demonstrate that RCP is fast and reliable enough to drive the BGP routing decisions for a large network. We show that RCP assigns routes correctly, even when the functionality is replicated and distributed, and that networks using RCP can expect comparable convergence delays to those using today's iBGP architectures.

1 Introduction

The Border Gateway Protocol (BGP), the Internet's interdomain routing protocol, is prone to protocol oscillation and forwarding loops, highly sensitive to topology changes inside an Autonomous System (AS), and difficult for operators to understand and manage. We address these problems by introducing a Routing Control Platform (RCP) that computes the BGP routes for each router in an AS based on complete routing information and higher-level network engineering goals [1, 2]. This paper describes the design and implementation of an RCP prototype that is fast and reliable enough to coordinate routing for a large backbone network.

1.1 Route Distribution Inside an AS

The routers in a single AS exchange routes to external destinations using a protocol called internal BGP (iBGP). Small networks are typically configured as a "full mesh" iBGP topology, with an iBGP session between each pair of routers. However, a full-mesh configuration does not scale because each router must: (i) have an iBGP session with every other router, (ii) send BGP update messages to every other router, (iii) store a local copy of the advertisements sent by each neighbor for each destination prefix, and (iv) have a new iBGP session configured whenever a new router is added to the network. Although having a faster processor and more memory on every router would support larger full-mesh configurations, the installed base of routers lags behind the technology curve, and upgrading routers is costly. In addition, BGP-speaking routers do not always degrade gracefully when their resource limitations are reached; for example, routers crashing or experiencing persistent routing instability under such conditions have been reported [3]. In this paper, we present the design, implementation, and evaluation of a solution that behaves like a full-mesh iBGP configuration with much less overhead and no changes to the installed base of routers.

To avoid the scaling problems of a full mesh, today's large networks typically configure iBGP as a hierarchy of *route reflectors* [4]. A route reflector selects a single BGP route for each destination prefix and advertises the route to its clients. Adding a new router to the system simply requires configuring iBGP sessions to the router's route reflector(s). Using route reflectors reduces the memory and connection overhead on the routers, at the expense of compromising the behavior of the underlying network. In particular, a route reflector does not necessarily select

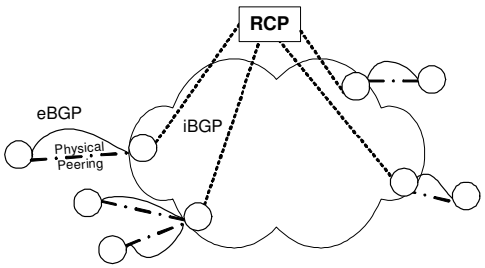

Figure 1: Routing Control Platform (RCP) in an AS

the same BGP route that its clients would have chosen in a full-mesh configuration. Unfortunately, the routers along a path through the AS may be assigned different BGP routes from different route reflectors, leading to inconsistencies [5]. These inconsistencies can cause protocol oscillation [6, 7, 8] and persistent forwarding loops [6]. To prevent these problems, operators must ensure that route reflectors and their clients have a consistent view of the internal topology, which requires configuring a large number of routers as route reflectors. This forces large backbone networks to have dozens of route reflectors to reduce the likelihood of inconsistencies.

1.2 Routing Control Platform (RCP)

RCP provides both the intrinsic correctness of a full-mesh iBGP configuration and the scalability benefits of route reflectors. RCP selects BGP routes on behalf of the routers in an AS using a complete view of the available routes and IGP topology. As shown in Figure 1, RCP has iBGP sessions with each of the routers; these sessions allow RCP to learn BGP routes and to send each router a routing decision for each destination prefix. Unlike a route reflector, RCP may send a different BGP route to each router. This flexibility allows RCP to assign each router the route that it would have selected in a full-mesh configuration, while making the number of iBGP sessions at each router independent of the size of the network. We envision that RCP may ultimately exchange interdomain routing information with neighboring domains, while still using iBGP to communicate with its own routers. Using the RCP to exchange reachability information across domains would enable the Internet's routing architecture to evolve [1].

To be a viable alternative to today's iBGP solutions, RCP must satisfy two main design goals: (i) consistent assignment of routes even when the functionality is replicated and distributed for reliability and (ii) fast response to network events, such as link failures and external BGP routing changes, even when computing routes for a large number of destination prefixes and routers. This paper demonstrates that RCP can be made fast and reliable enough to supplant today's iBGP architectures, without requiring any changes to the implementation of the legacy routers. After a brief overview of BGP routing in Section 2, Section 3 presents the RCP architecture and describes how to compute consistent forwarding paths, without requiring any explicit coordination between the replicas. In Section 4, we describe a prototype implementation, built on commodity hardware, that can compute and disseminate routing decisions for a network with hundreds of routers. Section 5 demonstrates the effectiveness of our prototype by replaying BGP and OSPF messages from a large backbone network; we also discuss the challenges of handling OSPF-induced BGP routing changes and evaluate one potential solution. Section 6 summarizes the contributions of the paper.

1.3 Related Work

We extend previous work on route monitoring [9, 10] by building a system that also *controls* the BGP routing decisions for a network. In addition, RCP relates to recent work on router software [11, 12, 13], including the proprietary systems used in today's commercial routers; in contrast to these efforts, RCP makes per-router routing decisions for an *entire network*, rather than a single router. Our work relates to earlier work on applying routing policy at route servers at the exchange points [14], to obviate the need for a full mesh of eBGP sessions; in contrast, RCP focuses on improving the scalability and correctness of distributing and selecting BGP routes within a single AS. The techniques used by the RCP for efficient storage of the per-router routes are similar to those employed in route-server implementations [15].

Previous work has proposed changes to iBGP that prevent oscillations [16, 7]; unlike RCP, these other proposals require significant modifications to BGP-speaking routers. RCP's logic for determining the BGP routes for each router relates to previous research on network-wide routing models for traffic engineering [17, 18]; RCP focuses on *real-time control* of BGP routes rather than modeling the BGP routes in today's routing system. Previous work has highlighted the need for a system that has network-wide control of BGP routing [1, 2]; in this paper, we present the design, implementation, and evaluation of such a system. For an overview of *architecture* and *standards* activities on separating routing from routers, see the related work discussions in [1, 2].

2 Interoperating With Existing Routers

This section presents an overview of BGP routing inside an AS and highlights the implications on how RCP must work to avoid requiring changes to the installed base of IP routers.

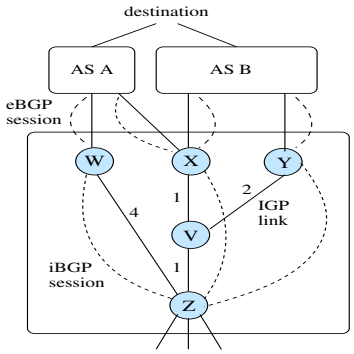

Figure 2: Network with three egress routers connecting to two neighboring ASes: Solid lines correspond to physical links (annotated with IGP link weights) and dashed lines correspond to BGP sessions.

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

Table 1: Steps in the BGP route-selection process

Partitioning of functionality across routing protocols: In most backbone networks, the routers participate in three different routing protocols: external Border Gateway Protocol (eBGP) to exchange reachability information with neighboring domains, internal BGP (iBGP) to propagate the information inside the AS, and an Interior Gateway Protocol (IGP) to learn how to reach other routers in the same AS, as shown in Figure 2. BGP is a path-vector protocol where each network adds its own AS number to the path before propagating the announcement to the next domain; in contrast, IGPs such as OSPF and IS-IS are typically link-state protocols with a tunable weight on each link. Each router combines the information from the routing protocols to construct a local forwarding table that maps each destination prefix to the next link in the path. *In our design, RCP assumes responsibility for assigning a single best BGP route for each prefix to each router and distributing the routes using iBGP, while relying on the routers to "merge" the BGP and IGP data to construct their forwarding tables.*

BGP route-selection process: To select a route for each prefix, each router applies the decision process in Table 1 to the set of routes learned from its eBGP and iBGP neighbors [19]. The decision process essentially compares the routes based on their many attributes. In the simplest case, a router selects the route with the shortest AS path (step 2), breaking a tie based on the ID of the router who advertised the route (step 7). However, other steps depend on route attributes, such as local preference, that are assigned by the routing policies configured on the border routers. *RCP must deal with the fact that the border routers apply policies to the routes learned from their eBGP neighbors and all routers apply the route-selection process to the BGP routes they learn.*

Selecting the closest egress router: In backbone networks, a router often has multiple BGP routes that are "equally good" through step 5 of the decision process. For example, router Z in Figure 2 learns routes to the destination with the same AS path length from three border routers W, X, and Y. To reduce network resource consumption, the BGP decision process at each router selects the route with the *closest* egress router, in terms of the IGP path costs. Router Z selects the BGP route learned from router X with an IGP path cost of 2. This practice is known as "early-exit" or "hot-potato" routing. *RCP must have a real-time view of the IGP topology to select the closest egress router for each destination prefix on behalf of each router. When the IGP topology changes, RCP must identify which routers should change the egress router they are using.*

Challenges introduced by hot-potato routing: A single IGP topology change may cause multiple routers to change their BGP routing decisions for multiple prefixes. If the IGP weight of link V–X in Figure 2 increased from 1 to 3, then router Z would start directing traffic through egress Y instead of X. When multiple destination prefixes are affected, these hot-potato routing changes can lead to large, unpredictable shifts in traffic [20]. In addition, the network may experience long convergence delays because of the overhead on the routers to revisit the BGP routing decisions across many prefixes. Delays of one to two minutes are not uncommon [20]. *To implement hot-potato routing, RCP must determine the influence of an IGP change on every router for every prefix. Ultimately, we view RCP as a way to move beyond hot-potato routing toward more flexible ways to select egress routers, as discussed in Section 5.4.*

3 RCP Architecture

In this section, we describe the RCP architecture. We first present the three building blocks of the RCP: the IGP Viewer, the BGP Engine, and the Route Control Server (RCS). We describe the information that is available to each module, as well as the constraints that the RCS must satisfy when assigning routes. We then discuss how RCP's functionality can be replicated and distributed across many physical nodes in an AS while maintaining consistency and correctness. Our analysis shows that there is no need for the replicas to run a separate consistency protocol: since the RCP is designed such that each RCS replica makes routing decisions only for the partitions for which it has complete IGP topology

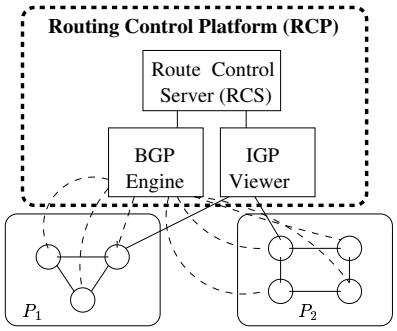

Figure 3: RCP interacts with the routers using standard routing protocols. RCP obtains IGP topology information by establishing IGP adjacencies (shown with solid lines) with one or more routers in the AS and BGP routes via iBGP sessions with each router (shown with dashed lines). RCP can control and obtain routing information from routers in separate network partitions (P_1 and P_2). Although this figure shows RCP as a single box, the functionality can be replicated and distributed, as we describe in Section 3.2.

and BGP routes, every replica will make the same routing assignments, even without a consistency protocol.

3.1 RCP Modules

To compute the routes that each router would have selected in a "full mesh" iBGP configuration, RCP must obtain both the IGP topology information and the best route to the destination from every router that learns a route from neighboring ASes. As such, RCP comprises of three modules: the IGP Viewer, the BGP Engine, and the Route Control Server. The *IGP Viewer* establishes IGP adjacencies to one or more routers, which allows the RCP to receive IGP topology information. The *BGP Engine* learns BGP routes from the routers and sends the RCS's route assignments to each router. The *Route Control Server (RCS)* then uses the IGP topology from the IGP Viewer information and the BGP routes from the BGP engine to compute the best BGP route for each router.

RCP communicates with the routers in an AS using standard routing protocols, as summarized in Figure 3. Suppose the routers R in a single AS form an IGP connectivity graph $G = (R, E)$, where E are the edges in the IGP topology. Although the IGP topology within an AS is typically a single connected component, failures of links, routers, or interfaces may occasionally create partitions. Thus, G contains one or more connected components; i.e., $G = \{P_1, P_2, \ldots, P_n\}$. The RCS only computes routes for partitions P_i for which it has complete IGP and BGP information, and it computes routes for each partition independently.

3.1.1 IGP Viewer

The RCP's *IGP Viewer* monitors the IGP topology and provides this information to the RCS. The IGP Viewer establishes IGP adjacencies to receive the IGP's link-state advertisements (LSAs). To ensure that the IGP Viewer never routes data packets, the links between the IGP Viewer and the routers should be configured with large IGP weights to ensure that the IGP Viewer is not an intermediate hop on any shortest path. Since IGPs such as OSPF and IS-IS perform *reliable flooding* of LSAs, the IGP Viewer maintains an up-to-date view of the IGP topology as the link weights change or equipment goes up and down. Use of flooding to disseminate LSAs implies that the IGP Viewer can receive LSAs from all routers in a partition by simply having an adjacency to a single router in that partition. This seemingly obvious property has an important implication:

Observation 1 *The IGP Viewer has the complete IGP topology for all partitions that it connects to.*

The IGP Viewer computes pairwise shortest paths for all routers in the AS and provides this information to the RCS. The IGP Viewer must discover only the path costs between any two routers in the AS, but it need not discover the weights of each IGP edge. The RCS then uses these path costs to determine, from any router in the AS, what the closest egress router should be for that router.

In some cases, a group of routers in the IGP graph all select the same router en route to one or more destinations. For example, a network may have a group of access routers in a city, all of which send packets out of that city towards one or more destinations via a single gateway router. These routers would always use the same BGP router as the gateway. These groups can be formed according to the IGP topology: for example, routers can be grouped according to OSPF "areas", since all routers in the same area typically make the same BGP routing decision. Because the IGP Viewer knows the IGP topology, it can determine which groups of routers should be assigned the same BGP route. By clustering routers in this fashion, the IGP Viewer can reduce the number of independent route computations that the RCS must perform. While IGP topology is a convenient way for the IGP Viewer to determine these groups of routers, the groups need not correspond to the IGP topology; for example, an operator could dictate the grouping.

3.1.2 BGP Engine

The *BGP Engine* maintains an iBGP session with each router in the AS. These iBGP sessions allow the RCP to (1) learn about candidate routes and (2) communicate its routing decisions to the routers. Since iBGP runs over

TCP, a BGP Engine need not be physically adjacent to every router. In fact, a BGP Engine can establish and maintain iBGP sessions with any router that is reachable via the IGP topology, which allows us to make the following observation:

Observation 2 *A BGP Engine can establish iBGP sessions to all routers in the IGP partitions that it connects to.*

Here, we make a reasonable assumption that IGP connectivity between two endpoints is sufficient to establish a BGP session between them; in reality, persistent congestion or misconfiguration could cause this assumption to be violated, but these two cases are anomalous. In practice, routers are often configured to place BGP packets in a high-priority queue in the forwarding path to ensure the delivery of these packets even during times of congestion.

In addition to receiving BGP updates, the RCP uses the iBGP sessions to send the chosen BGP routes to the routers. Because BGP updates have a "next hop" attribute, the BGP Engine can advertise BGP routes with "next hop" addresses of other routers in the network. This characteristic means that *the BGP Engine does not need to forward data packets*. The BGP routes typically carry "next hop" attributes according to the egress router at which they were learned. Thus, the RCS can send a route to a router with the next hop attribute unchanged, and routers will forward packets towards the egress router.

A router interacts with the BGP Engine in the same way as it would with a normal BGP-speaking router, but the BGP Engine can send a different route to each router. (In contrast, a traditional route reflector would send the same route to each of its neighboring routers.) A router only sends BGP update messages to the BGP Engine when selecting a new best route learned from a neighboring AS. Similarly, the BGP Engine only sends an update when a router's decision should change.

3.1.3 Route Control Server (RCS)

The RCS receives IGP topology information from the IGP Viewer and BGP routes from the BGP Engine, computes the routes for a group of routers, and returns the resulting route assignments to the routers using the BGP Engine. The RCS does not return a route assignment to any router that has already selected a route that is "better" than any of the other candidate routes, according to the decision process in Table 1. To make routing decisions for a group of routers in some partition, the following must be true:

Observation 3 *An RCS can only make routing decisions for routers in a partition for which it has* both *IGP and BGP routing information.*

Note that the previous observations guarantee that the RCS can (and will) make path assignments for all routers in that partition. Although the RCS has considerable flexibility in assigning routes to routers, one reasonable approach would be to have the RCS send to each router the route that it would have selected in a "full mesh" iBGP configuration. To emulate a full-mesh iBGP configuration, the RCS executes the BGP decision process in Table 1 on behalf of each router. The RCS can perform this computation because: (1) knowing the IGP topology, the RCS can determine the set of egress routers that are reachable from any router in the partitions that it sees; (2) the next four steps in the decision process compare attributes that appear in the BGP messages themselves; (3) for step 5, the RCS considers a route as eBGP-learned for the router that sent the route to the RCP, and as an iBGP-learned route for other routers; (4) for step 6, the RCS compares the IGP path costs sent by the IGP Viewer; and (5) for step 7, the RCS knows the router ID of each router because the BGP Engine has an iBGP session with each of them. After computing the routes, the RCS can send each router the appropriate route.

Using the high-level correctness properties from previous work as a guide [21], we recognize that routing within the network must satisfy the following properties (note that iBGP does *not* intrinsically satisfy them [6, 21]):

Route validity: The RCS should not assign routes that create forwarding loops, blackholes, or other anomalies that prevent packets from reaching their intended destinations. To satisfy this property, two invariants must hold. First, the RCS must assign routes such that the routers along the shortest IGP path from any router to its assigned egress router must be assigned a route with the same egress router. Second, the RCS must assign a BGP route such that the IGP path to the next-hop of the route only traverses routers in the same partition as the next-hop.

When the RCS computes the same route assignments as those the routers would select in a full mesh iBGP configuration, the first invariant will always hold, for the same reason that it holds in the case of full mesh iBGP configuration. In a full mesh, each router simply selects the egress router with the shortest IGP path. All routers along the shortest path to that egress also select the same closest egress router. The second invariant is satisfied because the RCS never assigns an egress router to a router in some other partition. Generally, the RCS has considerable flexibility in assigning paths; the RCS must guarantee that these properties hold even when it is not emu-

lating a full mesh configuration.

Path visibility: Every router should be able to exchange routes with at least one RCS. Each router in the AS should receive some route to an external destination, assuming one exists. To ensure that this property is satisfied, each partition must have at least one IGP Viewer, one BGP Engine, and one RCS. Replicating these modules reduces the likelihood that a group of routers is partitioned such that it cannot reach at least one instance of these three components. If the RCS is replicated, then two replicas may assign BGP routes to groups of routers along the same IGP path between a router and an egress. To guarantee that two replicas do not create forwarding loops when they assign routes to routers in the same partition, they must make consistent routing decisions. If a network has multiple RCSes, the route computation performed by the RCS must be deterministic: the same IGP topology and BGP route inputs must always produce the same outcome for the routers.

If a partition forms such that a router is partitioned from RCP, then we note that (1) the situation is no worse than today's scenario, when a router cannot receive BGP routes from its route reflector and (2) in many cases, the router will still be able to route packets using the routes it learns via eBGP, which will likely be its best routes since it is partitioned from most of the remaining network anyway.

3.2 Consistency with Distributed RCP

In this section, we discuss the potential consistency problems introduced by replicating and distributing the RCP modules. To be robust to network partitions and avoid creating a single point of failure, the RCP modules should be replicated. (We expect that many possible design strategies will emerge for assigning routers to replicas. Possible schemes include using the closest replica, having primary and backup replicas, etc.) Replication introduces the possibility that each RCS replica may have different views of the network state (i.e., the IGP topology and BGP routes). These inconsistencies may be either transient or persistent and could create problems such as routing loops if routers were learning routes from different replicas.[1] The potential for these inconsistencies would seem to create the need for a consistency protocol to ensure that each RCS replica has the same view of the network state (and, thus, make consistent routing decisions). In this section, we discuss the nature and consequences of these inconsistencies and present the surprising result that no consistency protocol is required to prevent persistent inconsistencies.

After discussing why we are primarily concerned with consistency of the RCS replicas in steady state, we explain how our replication strategy guarantees that the

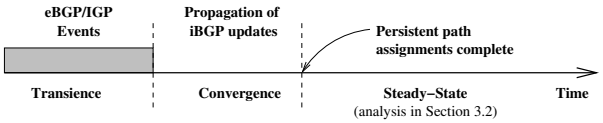

Figure 4: Periods during convergence to steady state for a single destination. Routes to a destination within an AS are stable most of the time, with periods of transience (caused by IGP or eBGP updates). Rather than addressing the behavior during the transient period, we analyze the consistency of paths assigned during steady state.

RCS replicas make the same routing decisions for each router in the steady state. Specifically, we show that, if multiple RCS replicas have IGP connectivity to some router in the AS, then those replicas will all make the same path assignment for that router. We focus our analysis on the consistency of RCS path assignments in steady state (as shown in Figure 4).

3.2.1 Transient vs. Persistent Inconsistencies

Since each replica may receive BGP and IGP updates at different times, the replicas may not have the same view of the routes to every destination at any given time; as a result, each replica may make different routing decisions for the same set of routers. Figure 4 illustrates a timeline that shows this transient period. During transient periods, routes may be inconsistent. On a per-prefix basis, long transient periods are *not the common case*: although BGP update traffic is fairly continuous, the update traffic *for a single destination* as seen by a single AS is relatively bursty, with prolonged periods of silence. That is, a group of updates may arrive at several routers in an AS during a relatively short time interval (i.e., seconds to minutes), but, on longer timescales (i.e., hours), the BGP routes for external destinations are relatively stable [22].

We are concerned with the consistency of routes for each destination *after* the transient period has ended. Because the network may actually be partitioned in "steady state", the RCP must still consider network partitions that may exist during these periods. Note that *any intra-AS routing protocol, including any iBGP configuration, will temporarily have inconsistent path assignments when BGP and IGP routes are changing continually*. Comparing the nature and extent of these transient inconsistencies in RCP to those that occur under a typical iBGP configuration is an area for future work.

3.2.2 RCP Replicas are Consistent in Steady State

The RCS replicas should make consistent routing decisions in steady state. Although it might seem that such a consistency requirement mandates a separate consistency protocol, we show in this section that such a protocol is not necessary.

Proposition 1 *If multiple RCSes assign paths to routers in P_i, then each router in P_i would receive the same route assignment from each RCS.*

Proof. Recall that two RCSes will only make different assignments to a router in some partition P_i if the replicas receive different inputs (i.e., as a result of having BGP routes from different groups of routers or different views of IGP topology). Suppose that RCSes A and B both assign routes to some router in P_i. By Observation 1, both RCSes A and B must have IGP topology information for *all* routers in P_i, and from Observation 2, they also have complete BGP routing information. It follows from Observation 3 that both RCSes A and B can make route assignments for all routers in P_i. Furthermore, since both RCSes have complete IGP and BGP information for the routers in P_i (i.e., the replicas receive the same inputs), then RCSes A and B will make the same route assignment to each router in P_i. ■

We note that certain failure scenarios may violate Observation 2; there may be circumstances under which IGP-level connectivity exists between the BGP engine and some router but, for some reason, the iBGP session fails (e.g., due to congestion, misconfiguration, software failure, etc.) As a result, Observation 3 may be overly conservative, because there may exist routers in some partition for which two RCSes may have BGP routing information from different subsets of routers in that partition. If this is the case, then, by design, neither RCS will assign routes to *any* routers in this partition, even though, collectively, both RCSes have complete BGP routing information. In this case, not having a consistency protocol affects liveness, but not correctness—in other words, two or more RCSes may fail to assign routes to routers in some partition even when they collectively have complete routing information, but in no case will two or more RCSes assign different routes to the same router.

4 RCP Architecture and Implementation

To demonstrate the feasibility of the RCP architecture, this section presents the design and implementation of an RCP prototype. Scalability and efficiency pose the main challenges, because backbone ASes typically have many routers (e.g., 500–1000) and destination prefixes (e.g., 150,000–200,000), and the routing protocols must converge quickly. First, we describe how the RCS computes the BGP routes for each group of routers in response to BGP and IGP routing changes. We then explain how the IGP Viewer obtains a view of the IGP topology and provides the RCS with only the necessary information for computing BGP routes. Our prototype of the IGP Viewer is implemented for OSPF; when describing our prototype, we will describe the IGP Viewer as the "OSPF Viewer". Finally, we describe how the BGP Engine exchanges BGP routing information with the routers in the AS and the RCS.

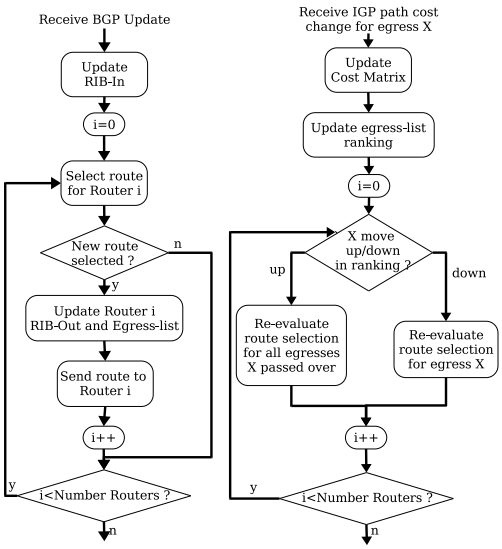

Figure 5: Route Control Server (RCS) functionality

4.1 Route Control Server (RCS)

The RCS processes messages received from both the BGP Engine(s) and the OSPF Viewer(s). Figure 5 shows the high level processing performed by the RCS. The RCS receives update messages from the BGP Engine(s) and stores the incoming routes in a Routing Information Base (RIB). The RCS perform per-router route selection and stores the selected routes in a per-router RIB-Out. The RIB-In and RIB-Out tables are implemented as a trie indexed on prefix. The RIB-In maintains a list of routes learned for each prefix; each BGP route has a "next hop" attribute that uniquely identifies the egress router where the route was learned. As shown in Figure 5, the RCS also receives the IGP path cost for each pair of routers from the IGP Viewer. The RCS uses the RIB-In to compute the best BGP routes for each router, using the IGP path costs in steps 0 and 6 of Table 1. After computing a route assignment for a router, the RCS sends that route assignment to the BGP Engine, which sends the update message to the router. The path cost changes received from the OSPF Viewer might require the RCS to re-compute selected routes when step 6 in the BGP decision process was used to select a route and the path cost to the selected egress router changes. Finding the routes that are affected can be an expensive process and as shown in Figure 5, our design uses a path-cost based ranking of egress routers to perform this efficiently. We now describe this approach and other design insights in

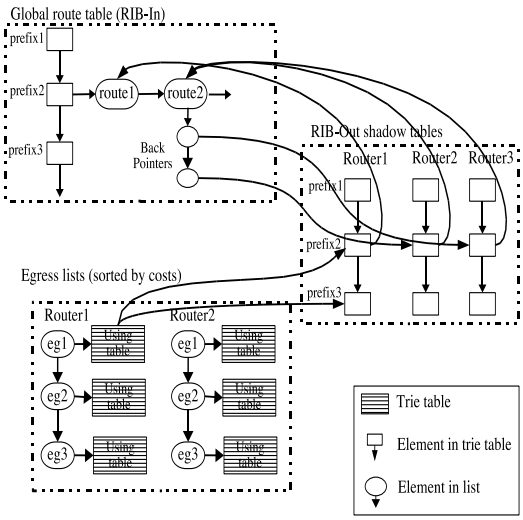

Figure 6: RCS RIB-In and RIB-Out data structures and egress lists

more detail with the aid of Figure 6, which shows the main RCS data structures:

Store only a single copy of each BGP route. Storing a separate copy of each router's BGP routes for every destination prefix would require an extraordinary amount of memory. To reduce storage requirements, the RCS only stores routes in the RIB-In table. The "next hop" attribute of the BGP route uniquely identifies the egress router where the BGP route was learned. Upon receiving an update message, the RCS can index the RIB-In by prefix and can add, update, or remove the appropriate route based on the next-hop attribute. To implement the RIB-Out, the RCS employs per-router *shadow tables* as a prefix-indexed trie containing pointers to the RIB-In table. Figure 6 shows two examples of these pointers from the RIB-Out to the RIB-In: router1 has been assigned the route1 for prefix2, whereas router2 and router3 have both been assigned route2 for prefix2.

Keep track of the routers that have been assigned each route. When a route is withdrawn, the RCS must recompute the route assignment for any router that was using the withdrawn route. To quickly identify the affected routers, each route stored in the RIB-In table includes a list of back pointers to the routers assigned this route. For example, Figure 6 shows two pointers from route2 in the RIB-In for prefix2 to indicate that router2 and router3 have been assigned this route. Upon receiving a withdrawal of the prefix from this next-hop attribute, the RCS reruns the decision process for each router in this list, with the remaining routes in the RIB-In, for those routers and prefix. Unfortunately, this ME optimization cannot be used for BGP *announcements*, because when a new route arrives, the RCS must recompute the route assignment for each router[2].

Maintain a ranking of egress routers for each router based on IGP path cost. A single IGP path-cost change may affect the BGP decisions for many destination prefixes at the ingress router. To avoid revisiting the routing decision for every prefix and router, the RCS maintains a ranking of egress points for each router sorted by the IGP path cost to the egress point (the "Egress lists" table in Figure 6). For each egress, the RCS stores pointers to the prefixes and routes in the RIB-Out that use the egress point (the "using table"). For example, router1 uses eg1 to reach both prefix2 and prefix3, and its using table contains pointers to those entries in the RIB-Out for router1 (which in turn point to the routes stored in the RIB-In). If the IGP path cost from router1 to eg1 increases, the RCS moves eg1 down the egress list until it encounters an egress router with a higher IGP path cost. The RCS then only recomputes BGP decisions for the prefixes that previously had been assigned the BGP route from eg1 (i.e., the prefixes contained in the using table). Similarly, if a path-cost change causes eg3 to become router1's closest egress point, the RCS resorts the egress list (moving eg3 to the top of the list) and only recomputes the routes for prefixes associated with the egresses routers "passed over" in the sorting process, i.e., eg1 and eg2, since they may now need to be assigned to eg3.

Assign routes to groups of related routers. Rather than computing BGP routes for each router, the RCS can assign the same BGP route for a destination prefix to a *group* of routers. These groups can be identified by the IGP Viewer or explicitly configured by the network operator. When the RCS uses groups, the RIB-Out and Egress-lists tables have entries for each *group* rather than each router, leading to a substantial reduction in storage and CPU overhead. The RCS also maintains a list of the routers in each group to instruct the BGP Engine to send the BGP routes to each member of the group. Groups introduce a trade-off between the desire to reduce overhead and the flexibility to assign different routes to routers in the same group. In our prototype implementation, we use the Points-of-Presence (which correspond to OSPF areas) to form the groups, essentially treating each POP as a single "node" in the graph when making BGP routing decisions.

4.2 IGP Viewer Instance: OSPF Viewer

The OSPF Viewer connects to one or more routers in the network to receive link-state advertisements (LSAs), as shown in Figure 3. The OSPF Viewer maintains an up-to-date view of the network topology and computes the path cost for each pair of routers. Figure 7 shows an overview of the processing performed by the OSPF Viewer. By providing path-cost changes and group membership information, the OSPF Viewer offloads work from the RCS in two main ways:

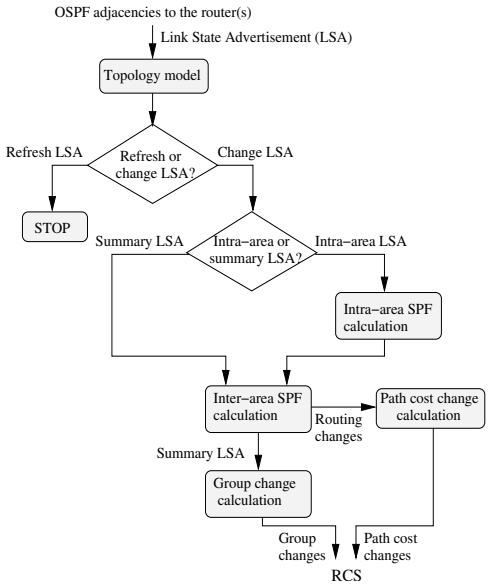

Figure 7: LSA Processing in OSPF Viewer

Send only path-cost changes to the RCS. In addition to originating an LSA upon a network change, OSPF periodically refreshes LSAs even if the network is stable. The OSPF Viewer filters the refresh LSAs since they do not require any action from the RCS. The OSPF Viewer does so by maintaining the network state as a topology model [9], and uses the model to determine whether a newly received LSA indicates a change in the network topology, or is merely a refresh as shown in Figure 7. For a change LSA, the OSPF Viewer runs shortest-path first (SPF) calculations from each router's viewpoint to determine the new path costs. Rather than sending all path costs to the RCS, the OSPF Viewer only passes the path costs that changed as determined by the "path cost change calculation" stage.

The OSPF Viewer must capture the influence of OSPF areas on the path costs. For scalability purposes, an OSPF domain may be divided into areas to form a hub-and-spoke topology. Area 0, known as the *backbone area*, forms the hub and provides connectivity to the non-backbone areas that form the spokes. Each link belongs to exactly one area. The routers that have links to multiple areas are called *border routers*. A router learns the entire topology of the area it has links into through "intra-area" LSAs. However, it does not learn the entire topology of remote areas (i.e., the areas in which the router does not have links), but instead learns the total cost of the paths to every node in remote areas from each border router the area has through "summary" LSAs.

It may seem that the OSPF Viewer can perform the SPF calculation over the entire topology, ignoring area boundaries. However, OSPF mandates that if two routers belong to the same area, the path between them must stay within the area even if a shorter path exists that traverses multiple areas. As such, the OSPF Viewer cannot ignore area boundaries while performing the calculation, and instead has to perform the calculation in two stages. In the first stage, termed the *intra-area stage*, the viewer computes path costs for each area separately using the intra-area LSAs as shown in Figure 7. Subsequently, the OSPF Viewer computes path costs between routers in different areas by combining paths from individual areas. We will term this stage of the SPF calculation as the *inter-area stage*. In some circumstances, the OSPF Viewer knows the topology of only a subset of areas, and not all areas. In this case, the OSPF Viewer can perform intra-area stage calculations only for the visible areas. However, use of summary LSAs from the border routers allows the OSPF Viewer to determine path costs to routers in non-visible areas from routers in visible areas during inter-area stage.

Reduce overhead at the RCS by combining routers into groups. The OSPF Viewer can capitalize on the area structure to reduce the number of routers the RCS must consider. To achieve this, the OSPF Viewer: (i) provides path cost information for all area 0 routers (which also includes border routers in non-zero areas), and (ii) forms a group of routers for each non-zero area and provides this group information. As an added benefit, the OSPF Viewer does not need physical connections to non-zero areas, since the summary LSAs from area 0 allows it to compute path costs from every area 0 router to every other router. The OSPF Viewer also uses the summary LSAs to determine the groups of routers. It is important to note that combining routers into groups is a construct internal to the RCP to improve efficiency, and it does not require any protocol or configuration changes in the routers.

4.3 BGP Engine

The BGP Engine receives BGP messages from the routers and sends them to the RCS. The BGP Engine also receives instructions from the RCS to send BGP routes to individual routers. We have implemented the BGP Engine by modifying the Quagga [11] software router to store the outbound routes on a *per-router* basis and accept route assignments from the RCS rather than computing the route assignments itself. The BGP Engine offloads work from the RCS by applying the following two design insights:

Cache BGP routes for efficient refreshes. The BGP Engine stores a local cache of the RIB-In and RIB-Out. The RIB-In cache allows the BGP Engine to provide the RCS with a fresh copy of the routes without affecting the routers, which makes it easy to introduce a new RCS replica or to recover from an RCS failure. Similarly, the RIB-Out cache allows the BGP Engine to re-send BGP

route assignments to operational routers without affecting the RCS, which is useful for recovering from the temporary loss of iBGP connectivity to the router. Because routes are assigned on a per-router basis, the BGP Engine maintains a RIB-Out for each router, using the same kind of data structure as the RCS.

Manage the low-level communication with the routers. The BGP Engine provides a simple, stable layer that interacts with the routers and maintains BGP sessions with the routers and multiplexes the update messages into a single stream to and from the RCS. It manages a large number of TCP connections and supports the low-level details of establishing BGP sessions and exchanging updates with the routers.

5 Evaluation

In this section, we evaluate our prototype implementation, with an emphasis on the scalability and efficiency of the system. The purpose of the evaluation is twofold. First, to determine the feasible operating conditions for our prototype, i.e., its performance as a function of the number of prefixes and routes, and the number of routers or router groups. Second, we want to determine what the bottlenecks (if any), would require further enhancements. We present our methodology in Section 5.1 and the evaluation results in Sections 5.2 and 5.3. In Section 5.4 we present experimental results of an approach that weakens the current tight coupling between IGP path-cost changes and BGP decision making.

5.1 Methodology

For a realistic evaluation, we use BGP and OSPF data collected from a Tier-1 ISP backbone on August 1, 2004. The BGP data contains both timestamped BGP updates as well as periodic table dumps from the network[3]. Similarly, the OSPF data contains timestamped Link State Advertisements (LSAs). We developed a *router-emulator* tool that reads the timestamped BGP and OSPF data and then "plays back" these messages against instrumented implementations of the RCP components. To initialize the RCS to realistic conditions, the router-emulator reads and replays the BGP table dumps before any experiments are conducted.

By selectively filtering the data, we use this single data set to consider the impact of network size (i.e., the number of routers or router groups in the network) and number of routes (i.e., the number of prefixes for which routes were received). We vary the network size by only calculating routes for a subset of the router groups in the network. Similarly, we only consider a subset of the prefixes to evaluate the impact of the number of routes on the RCP. Considering a subset of routes is relevant for networks that do not have to use a full set of Internet routes but might still benefit from the RCP functionality, such as private or virtual private networks.

For the RCS evaluation, the key metrics of interest are (i) the time taken to perform customized per-router route selection under different conditions and (ii) the memory required to maintain the various data structures. We measure these metrics in three ways:

- *Whitebox:* First, we perform whitebox testing by instrumenting specific RCS functions and measuring on the RCS both the memory usage and the time required to perform route selection when BGP and OSPF related messages are being processed.
- *Blackbox no queuing:* For blackbox no queuing, the router-emulator replays one message at a time and waits to see a response before sending the next message. This technique measures the additional overhead of the message passing protocol needed to communicate with the RCS.
- *Blackbox real-time:* For blackbox real-time testing, the router-emulator replays messages based on the timestamps recorded in the data. In this case, ongoing processing on the RCS can cause messages to be queued, thus increasing the effective processing times as measured at the router-emulator.

For all blackbox tests, the RCS sends routes back to the router-emulator to allow measurements to be done.

In Section 5.2, we focus our evaluation on how the RCP processes BGP updates and performs customized route selection. Our BGP Engine implementation extends the Quagga BGP daemon process and as such inherits many of its qualities from Quagga. Since we made no enhancements to the BGP protocol part of the BGP Engine but rely on the Quagga implementation we do not present an evaluation of its scalability in this paper[4]. Our main enhancement, the shadow tables maintained to realize per-router RIB-Outs, use the same data structures as the RCS, and hence, the evaluation of the RCS memory requirements is sufficient to show its feasibility.

In Section 5.3, we present an evaluation of the OSPF Viewer and the OSPF-related processing in the RCS. We evaluate the OSPF Viewer by having it read and process LSAs that were previously dumped to a file by a monitoring process. The whitebox performance of the OSPF Viewer is determined by measuring the time it takes to calculate the all pairs shortest paths and OSPF groups. The OSPF Viewer can also be executed in a test mode where it can log the path cost changes and group changes that would be passed to the RCS under normal operating conditions. The router-emulator reads and then plays back these logs against the RCS for blackbox evaluation of the RCS OSPF processing.

The evaluations were performed with the RCS and OSPF Viewer running on a dual 3.2 GHz Pentium-4 processor Intel system with 8 GB of memory and running a Linux 2.6.5 kernel. We ran the router-emulator on a 1 GHz Pentium-3 Intel system with 1 GB of memory and running a Linux 2.4.22 kernel.

5.2 BGP Processing

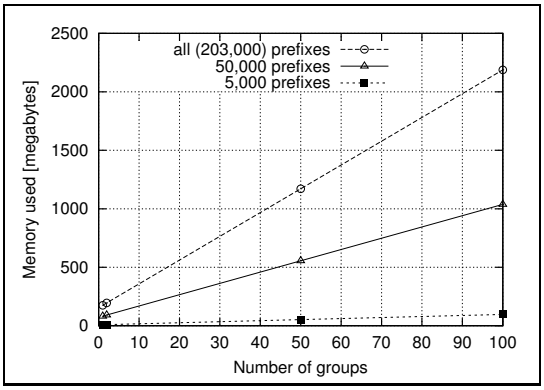

Figure 8: *Memory*: Memory used for varying numbers of prefixes.

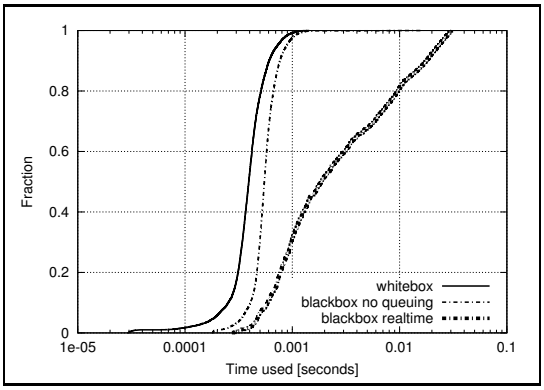

Figure 9: *Decision time, BGP updates*: RCS route selection time for whitebox testing (instrumented RCS), blackbox testing no queuing (single BGP announcements sent to RCS at a time), blackbox testing real-time (BGP announcements sent to RCS in real-time)

Figure 8 shows the amount of memory required by the RCS as a function of group size and for different numbers of prefixes. Recall that a group is a set of routers that would be receiving the same routes from the RCS. Backbone network topologies are typically built with a core set of backbone routers that interconnect points-of-presence (POPs), which in turn contain access routers [23]. All access routers in a POP would typically be considered part of a single group. Thus the number of groups required in a particular network becomes a function of the number of POPs and the number of backbone routers, but is independent of the number of access routers. A 100-group network therefore translates to quite a large network [5].

LSA Type	Percentage
Refresh	99.9244
Area 0 change	0.0057
Non-zero area change	0.0699

Table 2: LSA traffic breakdown for August 1, 2004

We saw more than 200,000 unique prefixes in our data. The effectiveness of the RCS shadow tables is evident by the modest rate of increase of the memory needs as the number of groups are increased. For example, storing all 203,000 prefixes for 1 group takes 175MB, while maintaining the table for 2 groups only requires an additional 21MB, because adding a group only increases the number of pointers into the global table, not the total number of unique routes maintained by the system. The total amount of memory needed for all prefixes and 100 groups is 2.2 GB, a fairly modest amount of memory by today's standards. We also show the memory requirements for networks requiring fewer prefixes.

For the BGP (only) processing considered in this subsection, we evaluate the RCS using 100 groups, all 203,000 prefixes and BGP updates only. Specifically, for these experiments the RCS used static IGP information and no OSPF related events were played back at the RCS.

Figure 9 shows BGP decision process times for 100 groups and all 203,000 prefixes for three different tests. First, the whitebox processing times are shown. The 90th percentile of the processing times for whitebox evaluation is 726 microseconds. The graph also shows the two blackbox test results, namely blackbox no queuing and blackbox realtime. As expected, the message passing adds some overhead to the processing times. The difference between the two blackbox results are due to the bursty arrival nature of the BGP updates, which produces a queuing effect on the RCS. An analysis of the BGP data show that the average number of BGP updates over 24 hours is only 6 messages per second. However, averaged over 30 second intervals, the maximum rate is much higher, going well over 100 messages per second several times during the day.

5.3 OSPF and Overall Processing

In this section, we first evaluate only the OSPF processing of RCP by considering both the performance of the OSPF Viewer and the performance of the RCS in processing OSPF-related messages. Then we evaluate the overall performance of RCP for combined BGP and OSPF related processing.

Measurement type	Area 0 change LSA	Non-zero area change LSA
Topology model	0.0089	0.0029
Intra-area SPF	0.2106	–
Inter-area SPF	0.3528	0.0559
Path cost change	0.2009	0.0053
Group change	–	0.0000
Miscellaneous	0.0084	0.0010
Total (whitebox)	0.7817	0.0653
Total (blackbox no queuing)	0.7944	0.0732
Total (blackbox realtime)	0.7957	0.1096

Table 3: Mean LSA processing time (in seconds) for the OSPF Viewer

OSPF: Recall that per LSA processing on the OSPF Viewer depends on the type of LSA. Table 2 shows the breakdown of LSA traffic into these types for August 1, 2004 data. Note that the refreshes account for 99.9% of the LSAs and require minimal processing in the OSPF Viewer; furthermore, the OSPF Viewer completely shields RCS from the refresh LSAs. For the remaining, i.e., change LSAs, Table 3 shows the whitebox, blackbox no queuing, and blackbox real-time measurements of the OSPF Viewer. The table also shows the breakdown of white-box measurements into various calculation steps.

The results in Table 3 allow us to make several important conclusions. First, and most importantly, the OSPF Viewer can process all change LSAs in a reasonable amount of time. Second, the SPF calculation and path cost change steps are the main contributors to the processing time. Third, the area 0 change LSAs take an order of magnitude more processing time than non-zero change LSAs, since area 0 changes require recomputing the path costs to every router; fortunately, the delay is still less than 0.8 seconds and, as shown in Table 2, area 0 changes are responsible for a very small portion of the change LSA traffic.

We now consider the impact of OSPF related events on the RCS processing times. Recall that OSPF events can cause the recalculation of routes by the RCS. We consider OSPF related events in isolation by playing back to the RCS only OSPF path cost changes; i.e., the RCS was pre-loaded with BGP table dumps into a realistic operational state, but no other BGP updates were played back.

Figure 10 shows RCS processing times caused by path cost changes for three different experiments with 100 router groups. Recall from Section 4.1 and Figure 6 that the sorted egress lists are used to allow the RCS to quickly find routes that are affected by a particular path cost change. The effectiveness of this scheme can be seen from Figure 10 where the 90th percentile for the whitebox processing is approximately 82 milliseconds. Figure 10 also shows the blackbox results for no queuing and realtime evaluation. As before the difference between the whitebox and blackbox no queuing results are due to the message passing overhead between the route-emulator (emulating the OSPF Viewer in this case) and the RCS. The processing times dominate relative to the message passing overhead, so these two curves are almost indistinguishable. The difference between the two blackbox evaluations suggests significant queuing effects in the RCS, where processing gets delayed because the RCS is processing earlier path cost changes, which is confirmed by an analysis of the characteristics of the path cost changes: while relatively few events occur during the day, some generate several hundred path cost changes per second. The 90th percentile of the blackbox realtime curve is 150 seconds. This result highlights the difficulty in processing internal topology changes. We discuss a more efficient way of dealing with this (the "filtered" curve in Figure 10) in Section 5.4.

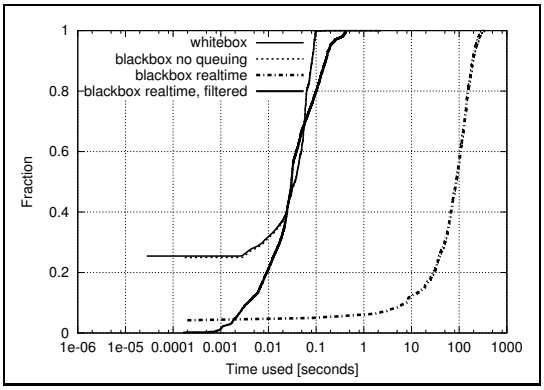

Figure 10: *Decision time, Path cost changes*: RCS route selection time for whitebox testing (instrumented RCS), blackbox testing no queuing (single path cost change sent to RCS at a time), blackbox testing realtime (path cost changes sent to RCS in real-time), blackbox testing real-time with filtered path cost changes

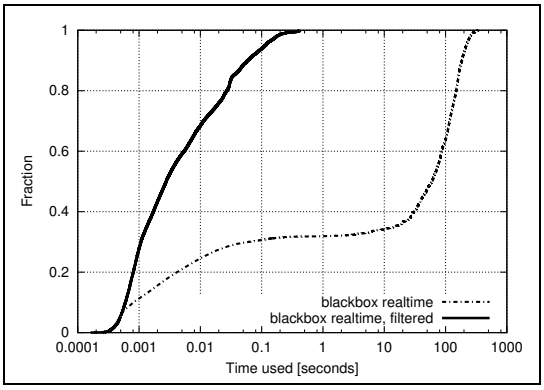

Figure 11: *Overall Processing Time, Blackbox testing BGP updates and Path cost changes combined*: All path cost changes (unfiltered) and filtered path cost changes

Overall: The above evaluation suggests that process-

ing OSPF path cost changes would dominate the overall processing time. This is indeed the case and Figure 11 shows the combined effect of playing back both BGP updates and OSPF path cost changes against the RCS. Clearly the OSPF path cost changes dominate the overall processing with the 90th percentile at 192 seconds. (The curve labeled "filtered" will be considered in the next section.)

5.4 Decoupling BGP from IGP

Although our RCP prototype handles BGP update messages very quickly, processing the internal topology changes introduces a significant challenge. The problem stems from the fact that a single event (such as a link failure) can change the IGP path costs for numerous pairs of routers, which can change the BGP route assignments for multiple routers and destination prefixes. This is fundamental to the way the BGP decision process uses the IGP path cost information to implement hot-potato routing.

The vendors of commercial routers also face challenges in processing the many BGP routing changes that can result from a single IGP event. In fact, some vendors do not execute the BGP decision process after IGP events and instead resort to performing a periodic scan of the BGP routing table to revisit the routing decision for each destination prefix. For example, some versions of commercial routers scan the BGP routing table once every 60 seconds, introducing the possibility of long inconsistencies across routers that cause forwarding loops to persist for tens of seconds [20]. The router can be configured to scan the BGP routing table more frequently, at the risk of increasing the processing load on the router.

RCP arguably faces a larger challenge from hot-potato routing changes than a conventional router, since RCP must compute BGP routes for multiple routers. Although optimizing the software would reduce the time for RCP to respond to path-cost changes, such enhancements cannot make the problem disappear entirely. Instead, we believe RCP should be used as a platform for moving beyond the artifact of hot-potato routing. In today's networks, a small IGP event can trigger a large, abrupt shift of traffic in a network [20]. We would like RCP to prevent these traffic shifts from happening, except when they are necessary to avoid congestion or delay.

To explore this direction, we performed an experiment where the RCP would not have to react to all internal IGP path cost changes, but only to those that impact the availability of the tunnel endpoint. We assume a backbone where RCP can freely direct an ingress router to any egress point that has a BGP route for the destination prefix, and can have this assignment persist across internal topology changes. This would be the case in a "BGP-free" core network, where internal routers do not have to run BGP, for example, an MPLS network or indeed any tunneled network. The edge routers in such a network still run BGP and therefore would still use IGP distances to select amongst different routes to the same destination. Some commercial router vendors accommodate this behavior by assigning an IGP weight to the tunnels and treating the tunnels as virtual IGP links. In the case of RCP, we need not necessarily treat the tunnels as IGP links, but would still need to assign some ranking to tunnels in order to facilitate the decision process.

We simulate this kind of environment by only considering OSPF path cost changes that would affect the availability of the egress points (or tunnel endpoints) but ignoring all changes that would only cause internal topology changes. The results for this experiment are shown with the *filtered* lines in Figures 10 and 11 respectively. From Figure 10, the 90th percentile for the decision time drops from 185 seconds when all path cost changes are processed to 0.059 seconds when the filtered path cost changes are used. Similarly, from Figure 11, the 90th percentile for the combined processing times drops from 192 seconds to 0.158 seconds when the filtered set is used. Not having to react to all path cost changes leads to a dramatic improvement on the processing times. Ignoring all path cost changes except those that would cause tunnel endpoints to disappear is clearly somewhat optimistic (e.g., a more sophisticated evaluation might also take traffic engineering goals into account), but it does show the benefit of this approach.

The results presented in this paper, while critically important, do not tell the whole story. From a network-wide perspective, we ultimately want to understand how long an RCP-enabled network will take to converge after a *BGP event*. Our initial results, presented in the technical report version of this paper [24], suggest that RCP convergence should be comparable to that of an iBGP route reflector hierarchy. In an iBGP topology with route reflection, convergence can actually take *longer* than with RCP in cases where routes must traverse the network multiple times before routing converges.

6 Conclusion

The networking research community has been struggling to find an effective way to redesign the Internet's routing architecture in the face of the large installed base of legacy routers and the difficulty of having a "flag day" to replace BGP. We believe that RCP provides an evolutionary path toward improving, and gradually replacing, BGP while remaining compatible with existing routers.

This paper takes an important first step by demonstrating that RCP is a viable alternative to the way BGP routes are distributed inside ASes today. RCP can emulate a full-mesh iBGP configuration while substantially reduc-

ing the overhead on the routers. By sending a customized routing decision to each router, RCP avoids the problems with forwarding loops and protocol oscillations that have plagued route-reflector configurations. RCP assigns routes consistently even when the functionality is replicated and distributed. Experiments with our initial prototype implementation show that the delays for reacting to BGP events are small enough to make RCP a viable alternative to today's iBGP architectures. We also showed the performance benefit of reducing the tight coupling between IGP path cost changes and the BGP decision process.

Acknowledgments

We would like to thank Albert Greenberg, Han Nguyen, and Brian Freeman at AT&T for suggesting the idea of an "Network Control Point for IP networks." Thanks also to Chris Chase, Brian Freeman, Albert Greenberg, Ali Iloglu, Chuck Kalmanek, John Mulligan, Han Nguyen, Arvind Ramarajan, and Samir Saad for collaborating with us on this project. We are grateful to Chen-Nee Chuah and Mythili Vutukuru, and our shepherd Ramesh Govindan, for their feedback on drafts of this paper.

7 REFERENCES

[1] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and J. van der Merwe, "The case for separating routing from routers," in *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture*, August 2004.

[2] O. Bonaventure, S. Uhlig, and B. Quoitin, "The case for more versatile BGP route reflectors." Internet Draft draft-bonaventure-bgp-route-reflectors-00.txt, July 2004.

[3] D.-F. Chang, R. Govindan, and J. Heidemann, "An empirical study of router response to large BGP routing table load," in *Proc. Internet Measurement Workshop*, November 2002.

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

[5] R. Dube, "A comparison of scaling techniques for BGP," *ACM Computer Communications Review*, vol. 29, July 1999.

[6] T. G. Griffin and G. Wilfong, "On the correctness of IBGP configuration," in *Proc. ACM SIGCOMM*, August 2002.

[7] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route oscillations in IBGP with route reflection," in *Proc. ACM SIGCOMM*, August 2002.

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

[9] A. Shaikh and A. Greenberg, "OSPF monitoring: Architecture, design, and deployment experience," in *Proc. Networked Systems Design and Implementation*, March 2004.

[10] Ipsum Route Dynamics. `http://www.ipsumnetworks.com/route_dynamics_overview.html`.

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

[12] M. Handley, O. Hudson, and E. Kohler, "XORP: An open platform for network research," in *Proc. SIGCOMM Workshop on Hot Topics in Networking*, October 2002.

[13] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Trans. Computer Systems*, vol. 18, pp. 263–297, August 2000.

[14] R. Govindan, C. Alaettinoglu, K. Varadhan, and D. Estrin, "Route servers for inter-domain routing," *Computer Networks and ISDN Systems*, vol. 30, pp. 1157–1174, 1998.

[15] R. Govindan, "Time-space tradeoffs in route-server implementation," *Journal of Internetworking: Research and Experience*, vol. 6, June 1995.

[16] V. Jacobson, C. Alaettinoglu, and K. Poduri, "BST - BGP Scalable Transport ." NANOG27 `http://www.nanog.org/mtg-0302/ppt/van.pdf`, February 2003.

[17] N. Feamster, J. Winick, and J. Rexford, "A model of BGP routing for network engineering," in *Proc. ACM SIGMETRICS*, June 2004.

[18] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "NetScope: Traffic engineering for IP networks," *IEEE Network Magazine*, pp. 11–19, March 2000.

[19] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)." Internet Draft draft-ietf-idr-bgp4-26.txt, work in progress, October 2004.

[20] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford, "Dynamics of hot-potato routing in IP networks," in *Proc. ACM SIGMETRICS*, June 2004.

[21] N. Feamster and H. Balakrishnan, "Detecting BGP configuration faults with static analysis," in *Proc. Networked Systems Design and Implementation*, May 2005.

[22] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *Proc. Internet Measurement Workshop*, November 2002.

[23] N. Spring, R. Mahajan, and D. Wetheral, "Measuring ISP topologies with RocketFuel," in *Proc. ACM SIGCOMM*, August 2002.

[24] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and J. van der Merwe, "Design and implementation of a routing control platform." `http://www.research.att.com/~kobus/rcp-nsdi-tr.pdf`, 2005.

Notes

[1] The seriousness of these inconsistencies depends on the mechanism that routers use to forward packets to a chosen egress router. If the AS uses an IGP to forward packets between ingress and egress routers, then inconsistent egress assignments along a single IGP path could result in persistent forwarding loops. On the other hand, if the AS runs a tunneling protocol (e.g., MPLS) to establish paths between ingress and egress routers, inconsistent route assignments are not likely to cause loops, assuming that the tunnels themselves are loop-free.

[2] Note that this optimization requires MED attributes to be compared across all routes in step 4 in Table 1. If MED attributes are only compared between routes with the same next-hop AS, the BGP decision process does not necessarily form a total ordering on a set of routes; consequently, the presence or absence of a non-preferred route may influence the BGP decision [17]. In this case, our optimization could cause the RCS to select a different best route than the router would in a regular BGP configuration.

[3] We filtered the BGP data so that only externally learned BGP updates were used. This represents the BGP traffic that an RCP would process when deployed.

[4] Our modular architecture would allow other BGP Engine implementations to be utilized if needed. Indeed, if required for scalability reasons, multiple BGP Engines can be deployed to "cover" a network.

[5] The per-process memory restrictions on our 32-bit platform prevented us from evaluating more groups.