How to Construct a Correct and Scalable iBGP Configuration

Mythili Vutukuru, Paul Valiant, Swastik Kopparty, and Hari Balakrishnan
MIT Computer Science and Artificial Intelligence Laboratory
{mythili,pvaliant,swastik,hari}@csail.mit.edu

***Abstract*— The Internet's current interdomain routing protocol, BGP (Border Gateway Protocol), has two modes of operation: eBGP (external BGP), used to exchange routing information between autonomous systems, and iBGP (internal BGP), used to propagate that information within an autonomous system (AS). In a "full mesh" iBGP configuration, every router has a BGP session with every border router in the AS. Because a full mesh configuration has a large number of iBGP sessions and does not scale well, configurations based on *route reflectors* are commonly used for intra-AS route dissemination [2]. Unfortunately, route reflector configurations violate important correctness properties [12], including *loop-free forwarding* and *complete visibility* to all eBGP-learned best routes, especially in the face of router and link failures.**

This paper presents and analyzes the first (to our knowledge) algorithm, `BGPSep`, to construct an iBGP session configuration that is both correct and more scalable than a full mesh. `BGPSep` uses the notion of a *graph separator*—a small set of nodes whose removal partitions a graph into connected components of roughly equal sizes—to choose route reflectors and iBGP sessions in a way that guarantees correctness. We evaluate an implementation of the `BGPSep` algorithm on several real-world and simulated network topologies and find that iBGP configurations generated by `BGPSep` have between 2.5 to 5$\times$ fewer iBGP sessions than a full mesh.

I. Introduction

The Internet is made up of many Autonomous Systems (ASes), which are networks or groups of networks under a common administration. An *egress router* at the border of an AS uses the Border Gateway Protocol (BGP) to exchange routes on an *external BGP* (eBGP) session with a peer router. Each eBGP router picks its best route to each destination, and then must disseminate that route to every external destination prefix to the other routers in the AS. These other routers in the AS obtain information about all possible routes to the destination, and pick their own best choice.

One approach to this intra-AS route dissemination is to rely on the Interior Gateway Protocol (IGP) used to propagate routing information for destinations within an AS, and introduce routes for external destinations into the IGP. This approach, however, does not work well because common IGPs such as OSPF [17], IS-IS [3], EIGRP [14], and RIP [16] do not handle the required scale well, and do not offer the policy expressiveness offered by BGP.

Another approach, used by some ASes, is to eliminate the problem of intra-AS route dissemination altogether by tunneling. Only egress routers maintain state about external destinations. Each interior router encapsulates the packet and tunnels it to some egress router, perhaps the closest one; that router then decapsulates the packet and forwards it to the destination. This approach may lead to inefficient paths or may require an additional protocol (such as MPLS [19]) inside the AS.

A third approach, used in many ASes today, is to set up BGP sessions between routers *inside* an AS, in a mode called *internal BGP* (iBGP). The designers of BGP proposed the use of a "full-mesh" iBGP configuration. Here, each eBGP router in the AS establishes BGP sessions with all the other routers (both eBGP routers and internal routers) in the network. These *internal BGP (iBGP)* sessions rely on the AS's underlying IGP to achieve connectivity.

The full-mesh configuration satisfies the following desirable correctness properties (Section II):

P1 **Complete visibility:** The dissemination of information amongst the routers should be "complete" in the sense that, for every external destination, each router picks the same route that it would have picked had it seen the best routes from every other eBGP router in the AS.

P2 **Loop-free forwarding:**[1] After the dissemination of eBGP learned routes, the resulting routes (and the subsequent forwarding paths of packets sent along those routes) picked by all routers should be free of deflections and forwarding loops [5], [12].

P3 **Robustness to IGP failures:** The route dissemination mechanism should be robust to node or link failures and IGP path cost changes—such changes should not result in a violation of the correctness property P2.

Unfortunately, the full-mesh configuration does not scale well: an AS with e eBGP routers and i interior routers needs to have $e(e-1)/2 + ei$ iBGP sessions. An Internet Service Provider (ISP) networks with hundreds of routers would require many thousands of sessions.

This lack of scalability has long been a problem with the full-mesh configuration, and has led to a few different proposals to improve scalability [22], [2]. The most common technique used today is *route reflection* [2], where a subset of BGP routers are designated as *route reflectors*, providing their best routes to other routers configured as their *clients*. Large networks use route reflectors hierarchically, but they are often configured in an unprincipled fashion. As a result, researchers have found that the three correctness properties described above can be violated ([5], [12], [8], [6]), leading to forwarding loops and sub-optimal paths.

Although previous work on iBGP configuration correctness [12], [8], [6], [7] gives sufficient conditions to *check* if a given iBGP configuration is correct, the problem of *constructing* correct and scalable iBGP configurations has not received much attention. In this paper, we describe the design, implementation, and evaluation of `BGPSep`, an algorithm to generate an iBGP configuration that guarantees properties P1, P2 and P3. `BGPSep` takes an IGP topology as input and produces a hierarchical configuration of route reflectors and reflector clients, as well as the associated iBGP sessions (Section III). We prove that the resulting iBGP configuration satisfies the desired correctness properties (Section IV), and show using an analysis of real-world ISP and synthetic network topologies that the number of iBGP sessions with `BGPSep` is significantly (between a factor of 2.5 and 5$\times$ in the ISP topologies) smaller than in a full-mesh configuration (Section VI).

`BGPSep` is based on the notion of a *graph separator*, a (small) set of nodes whose removal partitions a graph into roughly equal-sized connected components. `BGPSep` is practical—efficient algorithms for graph separators using spectral techniques are known [20], and our implementation uses this method (Section V). The run time of the spectral partitioning algorithm is cubic in the number of nodes in the graph. `BGPSep` takes between 5 seconds and 20 seconds to produce the iBGP configuration for real-world ISP topologies whose sizes range from 80 to 300 routers.

II. Background and Motivation

A. BGP route selection rules

For every external destination prefix, every BGP speaking router invokes the BGP decision process [18] to select one best route from the set of routes learned through eBGP and iBGP for that destination prefix. BGP's route selection process involves the comparison of the following attributes in this order: local preference, AS path length, multi-exit discriminator (MED), origin AS, and the IGP path cost to reach the egress (the route through the egress router with the lowest IGP cost is preferred). If two egresses have the same IGP path cost, then some deterministic mechanism such as the smallest router ID is used to break ties.[2] Every router then combines information about the egress router of the best route with the reachability information about the physical topology to map external destinations to outgoing links.

B. Route reflection

Route reflection [2] is a scalable way of disseminating external routes within an AS. Some BGP routers in an AS are designated as route reflectors. Normal iBGP peers do not propagate the route learned from one peer to another to avoid routing loops. Route reflectors, however, have different rules. A route reflector has two types of iBGP sessions: client iBGP sessions with some peers configured as route reflector clients and normal iBGP sessions with non-client peers. When a route reflector receives a route from an iBGP peer, it

[1] P1 subsumes P2 if the IGP enforces shortest path routing (Section II).

[2] In this paper, we do not consider the case of tied IGP costs explicitly. We assume a deterministic tie-breaking mechanism between routers with same IGP path costs.

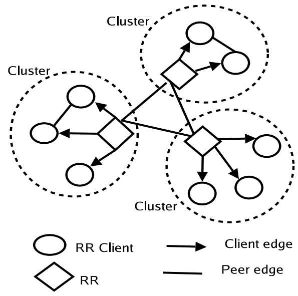

Figure 1. iBGP using route reflection.

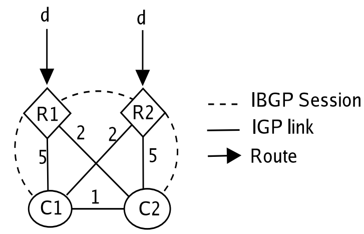

Figure 2. An incorrect iBGP configuration.

selects the best route using the rules mentioned above. After the best path is selected, it must do the following depending on the type of the peer it is receiving the best path from: (1) A route from a non-client iBGP peer is reflected to all the clients, and (2) A route from a client peer is reflected to all the non-client peers and also to the client peers. Because the client peers do not need to be fully meshed, the total number of iBGP sessions is much smaller than in a full mesh.

Typically, the route reflectors in a large AS are clustered hierarchically. Every route reflector cluster has one or more route reflectors. The other routers in the cluster are designated as clients of the route reflectors in the cluster. The clients within a cluster may or may not peer with each other. There could also be a hierarchy of route reflectors where a client of a route reflector acts as a route reflector for other routers. An example of an iBGP configuration with a single level hierarchy of route reflectors is shown in Figure 1. In general, this graph is different from the "physical" IGP graph of the network.

Feamster and Balakrishnan proved that for all the route reflectors to hear of the best routes available at each eBGP router in the network, the route reflectors at the top of the hierarchy must form a full mesh (a property they call "visibility", which is closely related to property P1) [8].

C. Problems with route reflection

In practice, iBGP configurations that use route reflection do not necessarily satisfy the properties P1, P2, and P3. We provide some examples of how these properties can be violated in typical iBGP configurations.[3]

a) Complete visibility: In an iBGP configuration using route reflection, a route reflector reflects *only* its best route (and not all routes it learns) to its clients and thus every router does not choose the same routes that it would have chosen had it seen all the eBGP-learned routes. Note that complete visibility would never be violated in a full mesh iBGP.

For example, consider the iBGP configuration shown in Figure 2. Route reflector R1 and its client C1 constitute a cluster, as do R2 and C2. The IGP and iBGP interconnections are as shown in the figure. Two routes, tied up to the step of comparing the IGP costs to the next hop, arrive at R1 and R2. R1 and R2 choose the routes through themselves as their best routes and advertise them to their clients. C1 chooses the route through R1 and C2 the one through R2. Had C1 learned of all the eBGP routes, however, it would have picked the route through R2 because C1 has a lower IGP cost to R2.

The property of complete visibility is important for efficient and predictable routing. If this property is violated, the system will fail to implement "hot-potato" routing, causing network resources to be wasted. Moreover, predicting the outcome of the complex BGP decision process is much easier when complete visibility is achieved, because every router is guaranteed to pick the route it would have picked had it seen all the eBGP learned routes. Such predictions prove useful while modeling BGP and for traffic engineering [10], [6].

[3] In the discussion that follows, the set of routes and egresses will refer to the set of routes filtered in the steps of comparing other attributes like local preference, AS path length, MED etc., that are tied up to the step of comparing the IGP cost. We will also refer to the "route" and the "egress router" that announces that route interchangeably.

b) Loop-free forwarding: Route reflector configurations are susceptible to forwarding anomalies such as deflections and forwarding loops [12], which make networks harder to maintain and debug. At every router, BGP selects only the egress router for a destination, while the actual forwarding path from that router to the egress is provided by the IGP. Some router on the shortest path to the egress may choose a different egress for the same external destination, causing the packets to be deflected along this forwarding path. Multiple deflections may interact to produce persistent forwarding loops [5], [12].

For example, again see Figure 2. Recall that C1 chooses the route through R1 and C2 the one through R2. C1's shortest path to R1 goes through C2 and C2's shortest path to R2 goes through C1. Thus, when C1 sends the packets destined to d to C2 (intending that they should reach R1), C2 sends them back to C1, because C2-C1-R2 is its chosen path to d. Any packet destined to d that reaches either C1 or C2 would be stuck in a loop.

Such forwarding anomalies never occur in an iBGP configuration that satisfies complete visibility if the IGP implements shortest path routing. If all routers choose the routes that they would have in a full mesh iBGP configuration, then all routers on the shortest path between a node and its egress router would also choose the route through the same egress router consistently. Thus property P1 subsumes P2 in this case.

c) Robustness to IGP failures: In arbitrary route reflector configurations, correctness properties like loop-free forwarding can be violated in the face of both IGP link cost changes and router or link failures. For example, consider the iBGP configuration shown in Figure 3. This IGP topology is similar to the topology in Figure 2, with the addition of a new route reflector R3, of which both C1 and C2 are clients. Both C1 and C2 choose R3 as their next hop for destination d and there are no deflections en route to R3. When R3 fails, the topology, now equivalent to the one in Figure 2, has a forwarding loop. Thus it is difficult to guarantee loop-free forwarding and complete visibility in arbitrary iBGP topologies with route reflection in the face of node or link failures. For the same reason, it is inherently difficult to build route reflector topologies with redundancy, because if a route reflector fails, the "backup" route reflector might end up causing forwarding loops!

Setting up correct iBGP configurations with route reflection in real world networks with a large number of BGP routers is a non-trivial task. Little work has been done on automated ways to set up such configurations. Today, network operators configure the iBGP based largely on heuristics. If the resulting system goes into a forwarding loop, they adopt "quick-fix" solutions like tweaking the IGP weights until the problem disappears, with the result that the IGP weights, which were initially set to represent meaningful quantities like end-to-end latency, lose their significance. We believe that there is a need for an organized framework to solve this problem, a way to configure iBGP using route reflection that gives *provable guarantees* on loop-free forwarding, complete visibility, and robustness to failures, without losing the scalability offered by route reflection.

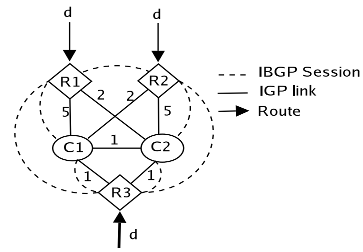

Figure 3. An iBGP configuration using route reflectors that has low fault-tolerance.

III. THE BGPSEP ALGORITHM

We now describe our algorithm, `BGPSep`, which constructs an iBGP configuration that provably satisfies the properties of complete visibility, loop-free forwarding, and robustness to IGP failures. `BGPSep` uses the graph-theoretic notion of graph separators to generate iBGP configurations.

A. Graph separators

A *graph separator* is a set of vertices whose removal separates a graph into two or more connected components. More formally, given a graph $G = (V, E)$, with a set V of vertices and a set E of edges, with $|V| = n$, a (k, ϵ)-separator is a set $S \subseteq V$ with the following properties:

- The induced subgraph on $V - S$ has no connected component of size $> n(\frac{1+\epsilon}{2})$.
- $|S| \leq k$

Let G_i and G_j be any two components in the induced subgraph on $V - S$. Then, any path beginning in a component G_i and ending in a different component G_j must pass through one or more routers in S. Our solution uses this property of graph separators.

The problem of finding the optimal graph separators of a graph is NP-hard in general. However, fast and practical algorithms for finding small separators are known for many families of graphs.[4] Our implementation of `BGPSep` uses the $O(n^3)$ *spectral partitioning algorithm* described in [20].

B. A simple algorithm

Let G denote the IGP subgraph induced by the eBGP routers and V denote the set of eBGP routers.[5] We now describe how to construct a basic iBGP configuration that satisfies P1, P2 and P3 without requiring a full-mesh iBGP. The next subsection optimizes this basic construction.

- **Step 1:** Consider a graph separator S of G. Make all the routers in S route reflectors.
- **Step 2:** For every $u, v \in S$, configure routers u and v as iBGP peers. Doing so forms a full mesh at the top level of the route reflector hierarchy (Section II).
- **Step 3:** Make each router in $V - S$ a route reflector client of *every* route reflector in S.
- **Step 4:** For each connected component G_i into which S separates G, set up an iBGP session between routers in G_i (i.e., construct a full-mesh configuration within each connected component, G_i).

We now argue informally that the property P1 (complete visibility) holds for this basic construction. The formal proofs that properties P1, P2 and P3 hold are given in Section IV.

To see that P1 holds, it is enough to show that any router picks the same route as it would have picked had it seen the best routes from all eBGP routers. Suppose router A would have picked the best route through egress router B in an iBGP configuration satisfying P1. Then, B is the closest egress to A with a route to d amongst the set of egress routers with the best route to d. The following claim proves that P1 is satisfied in the construction described above.

[4]Algorithms for finding $(\sqrt{8n}, 1/6)$-separators are known for planar graphs [20]. However IGP graphs of ISPs are not guaranteed to be planar.

[5]We assume for now that all BGP routers in the AS are egress routers, an assumption that we will relax later.

Claim 1: A learns of the route via B and thus picks B as its egress for destination d.

Proof: If A and B are in the same component (say, G_i), then they have an iBGP session between them because each component is fully meshed, and thus A will learn of the route via B. Otherwise, suppose A and B are in different components. Then, the shortest path between A and B will pass through S. Because all routers in each connected component G_j are clients of all route reflectors in S, there exists at least one route reflector R on the shortest path between A and B, of which both A and B are clients. If B is the closest egress to A, then B will be the closest egress to R as well. Thus, R will choose the route via B as its best route and reflect it to all its clients, and A will learn of that route through B. ■

Although this basic construction satisfies the correctness properties and has a smaller number of iBGP sessions compared to the full mesh iBGP (routers in different connected components no longer need to connect to each other, they only need to connect to the route reflectors in S), it still uses a full mesh within each of the individual components. To further reduce the number of iBGP sessions, we observe that the problem of avoiding a full mesh iBGP within G_1 and G_2 without violating P1, P2 and P3 is just a smaller instance of the original problem we started to solve on G. Thus, we can recursively apply the same algorithm within each of the components.

The recursion can terminate when the components are small enough to be fully meshed. We discuss the complete algorithm next.

C. The complete algorithm

Algorithm 1 shows the centralized recursive algorithm, `BGPSep`. The algorithm takes the graph $G = (V, E)$ formed by the BGP routers and produces the set I of iBGP sessions that must be established between the routers. Every element in I denotes an iBGP session and is of the form (u, v, t) where u and v are the routers between which the iBGP session is established and t is the type of the iBGP session. If t = "client", then the iBGP session between u and v is a client-route reflector session (with u being the client of route reflector v). If t = "peer", then the iBGP session between u and v is a normal non-client iBGP session. The recursion stops when the component has one or two routers. The algorithm uses a procedure `Graph-Separator`, which is a graph partitioning algorithm (e.g., the algorithm described in [20]) that takes a graph G and returns a graph separator S.

Algorithm BGPSep
Input: IGP Graph G, set V of BGP routers
Output: Set I of iBGP sessions
if $|V| = 1$ **then**
    $I = \emptyset$;
else if $|V| = 2$ **then**
    $\{u, v\} \leftarrow V$;
    $I = \{(u, v, \text{peer})\}$;
else
    /* Step 1: Choose a graph separator $S \subseteq V$. Routers in S are the route reflectors. */
    $S \leftarrow$ Graph-Separator(G) ;
    $G_1, \ldots, G_m \leftarrow$ components of $V - S$;
    /* Step 2: Fully mesh the set of route reflectors */
    foreach $u, v \in S, u \neq v$ **do**
        $I = I \cup \{(u, v, \text{peer})\}$;
    end
    foreach G_i **do**
        /* Step 3: Make every router in each component G_i a route reflector client of every route reflector */
        foreach $u \in G_i, v \in S$ **do**
            $I = I \cup \{(u, v, \text{client})\}$;
        end
        /* Step 4: Recursively apply BGPSep over each component */
        $I_i =$ BGPSep(G_i) ;
        $I = I \cup I_i$;
    end
end
return I;

Algorithm 1: BGPSep: A recursive separator-based algorithm to construct an iBGP configuration satisfying P1, P2, and P3.

Although the recursion shown in Algorithm 1 terminates when each component has one or two routers, it is easy to modify the algorithm to terminate the recursion at an earlier stage and fully mesh the routers in each connected component. In practice, it is likely that the maximum number of levels of recursion (which is also equal to the number of levels in the resulting route reflector hierarchy) will be a user-defined parameter.

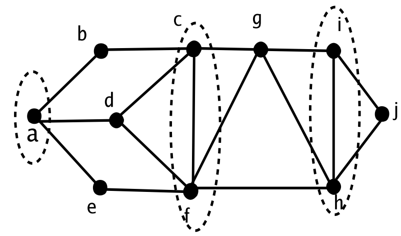

Figure 4. An example showing how BGPSep works.

D. An example

We now give a simple example to illustrate the BGPSep algorithm. Consider a network with ten BGP routers, as shown in Figure 4. Step 1 of the algorithm chooses the set of routers $S = \{c, f\}$ to separate the graph into two components, $G_1 = \{a, b, d, e\}$, and $G_2 = \{g, h, i, j\}$. In step 2, the algorithm fully meshes the set of route reflectors. Thus the set I of iBGP sessions will be $\{(c, f, \text{peer})\}$. In step 3, the algorithm makes each router in G_1 and G_2 a client of every route reflector in S.

Step 4 recursively applies this algorithm over G_1 and G_2. In step 1 of the recursion over G_1, the separator algorithm finds $S_1 = \{a\}$ as the separator of G_1 and $G_{11} = \{b\}$, $G_{12} = \{d\}$ and $G_{13} = \{e\}$ as the components in $G_1 - S_1$. No iBGP sessions are added in step 2 because the set $S_1 = \{a\}$ is a singleton here. In step 3, the algorithm adds the following iBGP sessions: (b, a, client), (d, a, client), and (e, a, client). The recursion terminates in each of the components G_{11}, G_{12} and G_{13} because they each have one router. Similarly, the algorithm recurses over component G_2, choosing the separator $S_2 = \{i, h\}$ in step 1. The algorithm adds the iBGP session (i, h, peer) in step 2 and the iBGP sessions (g, i, client), (g, h, client), (j, i, client), and (j, h, client) in step 3. The recursion now terminates because the components $\{g\}$ and $\{j\}$ have just one router each. The resulting iBGP configuration has two levels of route reflectors, five route reflectors, and 25 iBGP sessions. In contrast, a full-mesh iBGP configuration for this example has 45 iBGP sessions.

E. Variants

We now describe two variants of BGPSep optimized for different types of networks. Note that, in all these cases, the algorithm proposed in the previous section would still be valid and the variants are either further optimizations or for convenience.

1) Networks with internal routers: If the network contains interior routers that do not receive any external routes, then those routers need not mesh with each other. As presented earlier, when run over the entire topology of internal and egress routers, `BGPSep` may establish some unnecessary iBGP sessions. To avoid these sessions, in Step 1, the algorithm first finds a set S of routers to separate the graph into two components: G_{int}, containing the internal routers, and G_{ext} containing the egress routers. Steps 2 and 3 remain the same. In step 4, the algorithm only recurses on the component containing the egress routers because the internal routers need not have iBGP sessions with each other.

This modified algorithm is inefficient if the number of egress routers is very small. If $|S| > |G_{ext}|$, it is better to simply mesh each internal router to every egress router.

2) Backbone-like ISP networks: The IGP topology of a large ISP typically consists of a set of points-of-presence (PoPs) spread across the ISP's area of coverage [21]. Every PoP has some access routers that connect to customer networks, and one or two (for redundancy) backbone routers that connect the PoP to the rest of the network.[6] Typically, a route reflector configuration is formed over the IGP topology by configuring each PoP as an iBGP route reflector cluster. The backbone routers in a PoP are made route reflectors and all the access routers in a PoP are made clients of those route reflectors. For visibility, the route reflectors at the top level in the hierarchy should be fully meshed [7], [6]. In addition, a route reflector hierarchy can also be built over the route reflectors in the backbone.

Running the `BGPSep` algorithm on the entire IGP topology of an ISP produces an iBGP configuration that is likely to be very different from these conventional iBGP configurations. For example, an access router might have to connect to multiple route reflectors in different PoPs, which might be too much load on the access router.

To solve this problem, we propose a variant, `BGPSep-Backbone`, of the `BGPSep` algorithm that is better suited for ISP-like backbone networks. The idea is simple: run `BGPSep` on just the backbone routers to establish a route reflector hierarchy over the backbone alone. Configure the backbone routers in each PoP as route reflectors for the access routers in the PoP, as done in practice today. Configuring a route reflector hierarchy according to the `BGPSep` only ensures that every shortest path between two backbone routers has a route reflector hierarchy on it. The same property does not hold for shortest paths between access routers. Hence, the access routers within each PoP must be fully meshed for the correctness properties to hold.

[6]These PoPs typically correspond to "areas" in OSPF.

IV. Proof of Correctness

In this section, we prove that the iBGP configuration produced by `BGPSep` satisfies the properties P1, P2 and P3 described in Section I.

A. Complete visibility (P1)

Consider the IGP subgraph G induced by the BGP routers of a network. Let V denote the set of BGP routers. Let d denote any destination. Let E_d denote the set of egresses that have equally good routes (i.e., routes that have not been filtered in the steps of comparing the local preference, AS path length and MED values) to destination d. For every router A, let $\text{Egress}_d(A)$ denote the egress router from E_d that has the shortest IGP path cost from A.

For complete visibility to hold, we require that every router A chooses the route via egress $\text{Egress}_d(A)$ as its best route for destination d. We begin the proof by defining a *signaling chain*.

Definition 2: A signaling chain between two routers A and B is defined as a set of routers $A(= R_0), R_1, R_2, \dots, R_r, B = (R_{r+1})$ such that, for $i = 1 \dots r$, (i) R_i is a route reflector and (ii) at least one of R_{i+1} or R_{i-1} is a route reflector client of R_i.

Lemma 3: If there exists a signaling chain between routers A and B, and if B is an egress router for a destination d, then A learns of the best route via B for destination d.

Proof: For $i = 1 \dots r$, we first claim that R_i propagates the routes learned from R_{i+1} to R_{i-1}. If R_{i+1} is a route reflector client of R_i, then R_i propagates the routes learned from R_{i+1} to R_{i-1} because a route reflector reflects routes learned from clients along all other iBGP sessions. On the other hand, if R_{i-1} is a client of R_i, then the claim is true because a route reflector reflects routes learned on any of its iBGP sessions to all its clients. Thus the routes learned at $R_{r+1} = B$ propagate along the "chain" to $R_r, R_{r-1} \ldots$ and eventually reach $R_0 = A$. ■

Lemma 4: In the iBGP configuration produced by `BGPSep`, for any destination d, there exists a signaling chain between every router $A \in V$ and the egress router $\text{Egress}_d(A)$.

Proof: Let $B = \text{Egress}_d(A)$. If A and B have an iBGP session with each other, then the proof is trivial. Otherwise, consider the shortest path between A and B in G. It follows from the construction in `BGPSep` that this shortest path should pass through a set of recursively produced graph separators. Because the graph separators are configured as route reflectors and the routers in the components (separated by the separator) are all clients of these route reflectors, it follows that there exist router reflectors $R_1, \ldots R_r$ on the shortest path (in that order) such that $A(= R_0), R_1, R_2, \ldots, R_r, B = (R_{r+1})$ is a signaling chain. (Note that $R_1, \ldots R_r$ need not be adjacent to each other on the shortest path.

■

The following theorem follows from Lemmas 3 and 4.

Theorem 5: The iBGP configuration output by `BGPSep` satisfies the property of complete visibility.

B. Loop-free forwarding (P2)

We now use Theorem 5 to prove the following:

Theorem 6: The iBGP configuration output by `BGPSep` satisfies the property of loop-free forwarding.

Proof: From Theorem 5, we know that every router A learns of the best route to any destination d via its closest egress $B = \text{Egress}_d(A)$. For every router C on the shortest path from A to B, $\text{Egress}_d(C) = \text{Egress}_d(A) = B$. Thus every router on the shortest path between A and B also chooses B as its egress router. Therefore, there are no deflections when packets are forwarded along the shortest path from A to B, guaranteeing loop-free forwarding. In fact, by this argument, any iBGP configuration that satisfies complete visibility will also satisfy loop-free forwarding on an IGP that uses shortest path routing (i.e., P1 subsumes P2 in this case). ■

C. Robustness to IGP changes (P3)

Lemma 7: The iBGP configuration produced by `BGPSep` is not affected by changes in IGP link costs.

Proof: The proof is trivial because Algorithm 1 does not use IGP link costs in computing the iBGP configuration. ■

Lemma 8: The iBGP configuration produced by `BGPSep` satisfies the properties of loop-free forwarding and complete visibility in the face of IGP router and link failures.

Proof: If S is a separator of $G = (V, E)$, then for any subgraph $G' = (V', E')$ of G, $S \cap V'$ is a separator of G'. This property ensures that properties P1 and P2 hold even in the face of IGP router and link failures for the iBGP sessions produced by `BGPSep`. No reconfiguration is required to cope with these failures. ■

The proofs of this section are valid after the IGP has converged following a link cost change or failure.

D. Caveats and limitations

There is a class of failures, however, which break the correctness properties of our algorithm: iBGP failures, where only the iBGP configuration changes, without changing the underlying IGP topology (that is, when only the BGP function of a router or a BGP session between a pair of routers fails with the IP forwarding function still intact). If that happens, we can no longer assume that the nodes in the graph separators are all route reflectors, and thus our correctness guarantees break down. By similar reasoning, however, the correctness guarantees of the full-mesh iBGP also become invalid.

Through simulation, we have analyzed the probability of violation of loop-free forwarding and complete visibility on the iBGP configurations produced by `BGPSep` in the face of iBGP failures. The results are presented in Section VI.

Finally, we note that `BGPSep` is not an incremental algorithm, and must be re-run and new separators computed when new nodes or links are provisioned in the network. That is the only time when rerunning the algorithm is required for the correctness properties to hold. In particular, as explained earlier, the algorithm does not need to be run on failures or removals of links and routers.

V. Implementation

We implemented the `BGPSep` algorithm in less than 100 lines of Matlab code. The program reads the IGP graph from a file and writes the iBGP sessions to a file. We implemented the $O(n^3)$ spectral partitioning algorithm from [20] to find graph

AS	Name	Number of routers	Number of links
1221	Telstra	108	306
1239	Sprint	315	1944
1755	Ebone	87	322
3257	Tiscali	161	656
3967	Exodus	79	294
6461	Abovenet	138	748

Table I

ISP TOPOLOGIES USED.

separators. Our implementation of BGPSep took between 5 seconds and 20 seconds to run for real network topologies having between 80 and 300 nodes.

A good direction for future work would be to develop a tool that takes the router configuration files as input, infers the IGP topology from the configuration files, and produces the lines of configuration code corresponding to the iBGP sessions for each router. When integrated with a utility like rcc [8] that performs many of these tasks, our BGPSep implementation can prevent certain routing anomalies and ease the tasks of network configuration and network management.

VI. EVALUATION

In this section, we evaluate the performance of BGPSep on various real-world and synthetic topologies. For real-world topologies, we used the backbone topologies of 6 ISPs annotated with inferred link costs from the Rocketfuel project [15]. The ISP backbone topologies are summarized in Table I. We also evaluated BGPSep on synthetic topologies generated using GT-ITM [4]. The GT-ITM parameters in the graphs were set according to the suggestions in [13].

The main questions we address are:

1) How does the number of iBGP sessions produced by BGPSep compare with a full-mesh configuration? How many route reflectors are required? Our intent is to assess the extent to which recursively computed graph separators help reduce the number of iBGP sessions.
2) For a full-mesh configuration, the number of sessions scales quadratically with the number of eBGP routers. What is the empirically observed scaling behavior for the configurations produced by BGPSep?
3) How robust are the configurations produced by BGPSep to iBGP failures?

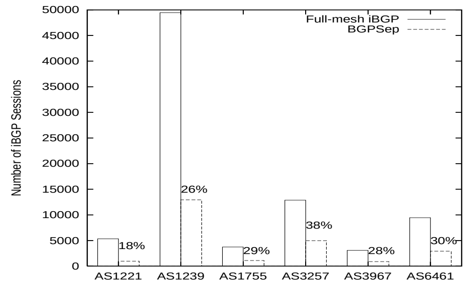

Figure 5. BGPSep vs. full-mesh iBGP: Rocketfuel ISP topologies.

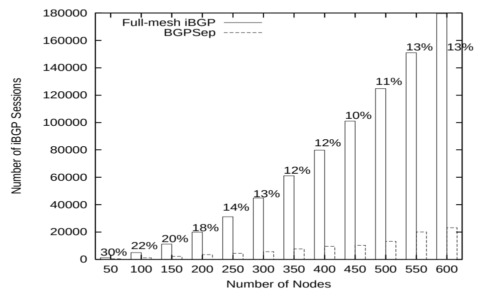

Figure 6. BGPSep vs. full-mesh iBGP: GT-ITM topologies.

For each network, let n denote the number of routers in the network and let N_{ibgp} denote the number of iBGP sessions produced by the BGPSep algorithm.

A. BGPSep *vs. full-mesh iBGP*

We compare N_{ibgp} to the number of iBGP sessions in a full-mesh iBGP for various networks. We use the 6 ISP backbones from Rocketfuel and synthetically generated Internet topologies from GT-ITM. We assume (conservatively) that all the nodes in the topology are external BGP routers.

The results are shown in Figures 5 (ISP) and 6 (GT-ITM). We observe that the iBGP configuration produced by BGPSep results in a $2.5\times$ to $5\times$ reduction in the number of iBGP sessions on the Rocketfuel ISP topologies, and a $5\times$ to $10\times$ reduction on GT-ITM topologies.[7]

From these results, we conclude that BGPSep makes a significant difference to the number of iBGP sessions compared to a full-mesh in all cases. The number of iBGP sessions in the Rocketfuel

[7] Another interesting number to comparison would have been to the number of iBGP sessions in the current deployment in each ISP. Unfortunately, these numbers are not publicly available.

AS	Routers	RRs	Top RRs	Levels
1221	108	34	5	6
1239	315	128	26	8
1755	87	56	3	5
3257	161	77	20	6
3967	79	45	4	5
6461	138	83	11	6

Table II

NUMBER OF ROUTE REFLECTORS AND TOP-LEVEL ROUTE REFLECTORS PRODUCED BY BGPSEP IN THE ROCKETFUEL ISP TOPOLOGIES.

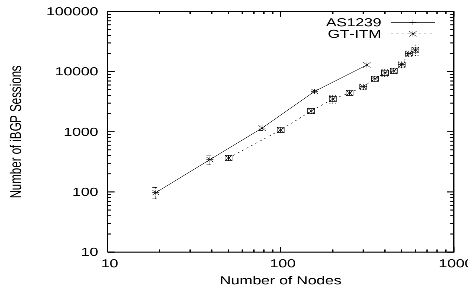

Figure 7. Number of iBGP sessions vs n: AS1239 and GT-ITM. The curves show the mean value and the error bars show one standard deviation.

1221	1239	1755	3257	3967	6461	GT-ITM
1.75	1.74	1.82	1.96	1.95	1.73	1.69

Table III

THE NUMBER OF IBGP SESSIONS SCALES AS n^k. THE TABLE SHOWS THE MEASURED VALUES OF k FOR EACH AS.

topologies is larger than in the synthetic GT-ITM topologies. We believe that the reason for the better performance on the synthetic topologies is that those topologies are produced using well-defined structured rules (a certain number of central cores, a certain number of stubs hanging off each core, etc.), leading to smaller-sized separators. Real-world topologies perhaps do not have as much structure and so have bigger separators.

Another key aspect of `BGPSep`'s scalability compared to a full mesh is the number of route reflectors and the number of levels in the resulting route reflector hierarchy. Let N_{rr}, N_{toprr}, and N_{level} denote the number of route reflectors, top-level route reflectors, and the number of levels in the route reflector hierarchy respectively. These values for various Rocketfuel ISP topologies are listed in Table II. We report the number of route reflectors at the top-most level of the hierarchy, because the top-level route reflectors usually have the most clients. These results show that although a substantial number of nodes are route reflectors, the number of top-level route reflectors (which have the most complex configuration) is usually relatively small.

B. Scaling behavior

We are interested in comparing the scaling behavior of the configurations produced by `BGPSep` to the quadratic behavior of a full-mesh configuration. To conduct this evaluation, we need topologies of various sizes, n. While it is easy to generate GT-ITM topologies with a varying number of routers, we don't have enough real-world ISP topologies of different sizes. We therefore turn to constructing subgraphs of various Rocketfuel topologies.

Let n_{orig} denote the number of routers in an ISP network. For each network topology, pick a random sub-topology of n nodes, where $n = \frac{n_{orig}}{2^i}, i = 0 \ldots 4$. This approach generates sub-topologies of five different sizes for each original topology, on which we run `BGPSep`. For each size (except when $i = 0$), we conducted 10 trials.

The resulting mean and standard deviation of the number of iBGP sessions for the AS1239 and GT-ITM topologies is shown in Figure 7. The graph is a log-log plot, and the slope of the best-fit line gives the desired scaling behavior (i.e., it gives the value of k such that the observed N_{ibgp} varies in proportion to n^k.

The value of this slope, k, for various ISPs is shown in Table III. The scaling behavior is not far from quadratic in all cases, suggesting that the reduction in the number of sessions is not because of a dramatic improvement in asymptotic scaling, but because the constant factor is significantly smaller than in full-mesh configurations.

We use the same method to investigate the scaling behavior of the number of route reflectors and the number of top-level route reflectors with n. The mean and the standard deviation of the number of route reflectors and the number of top-level route reflectors is shown in Figure 8 for AS1239. The slopes of the best-fit lines, k, are 0.95 and 0.53, respectively. The results on the other topologies were similar.

C. Robustness

In Section IV, we proved that the configuration produced by `BGPSep` is resilient to IGP failures

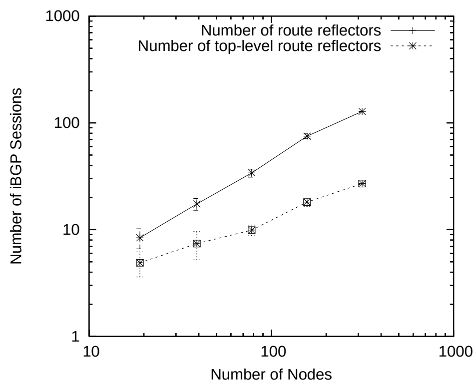

Figure 8. Number of route reflectors vs. n for AS1239. The curves show the mean value and the error bars show one standard deviation.

once the IGP converges to a loop-free topology. However, BGPSep does not guarantee correctness in the face of iBGP failures.

To study how resilient BGPSep is to iBGP failures, we randomly simulate iBGP failures on the configurations produced by BGPSep and determine how often the properties of loop-free forwarding and complete visibility are violated. We evaluate the extent of violation of loop-free forwarding by computing the fraction, $f_{incorrect}$, of forwarding paths that have a forwarding anomaly (such as deflections). For high values of failure rates, we also observed instances of persistent forwarding loops. We report the number of configurations, N_{FL}, that have a forwarding loop. We evaluate the extent of violation of complete visibility by computing the ratio, $r_{complete}$, of the path length taken by a packet to a destination with a given failure rate to the path length taken by the packet in a configuration that satisfies complete visibility.

Table IV shows the values of $r_{complete}$, $f_{incorrect}$, and N_{FL} for various failure rates on the configuration produced by BGPSep on the AS1221 Rocketfuel topology. We find that BGPSep handles low iBGP session failure rates well, but performs poorly at rates above 20%. We believe, however, that this lack of resilience at high failure rates is not a significant practical problem for two reasons: first, such high failure rates are highly unlikely, and second, BGPSep is no worse than the full-mesh, and is strictly better than heuristically placed route reflectors in general (as explained in Section IV).

Failure %	$r_{complete}$	$f_{incorrect}$	N_{FL}
0%	1	0	0
5%	1.0037	0.017	0
20%	1.0478	0.0911	4
40%	1.0839	0.1866	4
60%	1.1239	0.2773	3

Table IV
ROBUSTNESS TO IBGP FAILURES.

VII. RELATED WORK

The possibility of the occurrence of forwarding loops with route reflection was first reported by Dube [5]. The property of loop-free forwarding was studied in detail by Griffin and Wilfong [12], who proved that verifying whether an arbitrary iBGP configuration is "forwarding correct" is NP-hard. They also described a set of sufficient conditions so that an iBGP configuration is free of deflections and forwarding loops: (i) route reflectors should prefer client routes to non-client routes, and (ii) every shortest path between any two routers should be a valid "signaling path" (which is a generalization of the signaling chain described in Section IV)

The configuration generated by BGPSep does not satisfy either of these two sufficient conditions: it does not require that a route reflector prefer client routes to non-client routes, and the signaling chain in BGPSep is only a subset of the shortest path between any two routers. Yet, BGPSep guarantees forwarding correctness, implying also that the sufficient conditions in [12] are perhaps too restrictive.

Our work looks at the correctness properties of iBGP *after* the path assignment has converged and does not address BGP convergence. Basu *et al.* [1] study the problem of route oscillations in iBGP with route reflection. They show that deciding whether an iBGP configuration with route reflection can converge is NP-complete and propose a modification to iBGP that guarantees convergence. Griffin and Wilfong study the conditions under which the BGP configuration converges to a stable path assignment [12], and also examine MED-induced oscillations [11].

Feamster and Balakrishnan [7], [8] developed sufficient conditions to guarantee path visibility in iBGP configurations. BGPSep uses one of their results, making the top-level of the route reflector configuration a full mesh.

VIII. Conclusion

Perhaps the most complex part of the interaction between exterior and interior routing protocols on the Internet today arises in the scalable dissemination of external routes within an autonomous system. Unless done with care, this dissemination causes problems that include routing loops, forwarding loops, and forwarding path deflections, all of which lead to packet losses and sub-optimal paths. These problems are hard to diagnose and debug, and networks with these problems are hard to manage.

We proposed the `BGPSep` algorithm, which takes an IGP topology as input and produces a set of route reflectors and clients of route reflectors, such that the resulting iBGP configuration provably satisfies the correctness properties of complete visibility, loop-free forwarding and robustness to node and link failures. An evaluation of the algorithm on real-world ISP topologies and synthetic networks showed that `BGPSep`'s configurations achieve all the correctness guarantees of a full-mesh iBGP with a much smaller number of iBGP sessions. In particular, `BGPSep` requires between 2.5$\times$ and 5$\times$ fewer iBGP sessions across six real-world ISP topologies.

To our knowledge, `BGPSep` is the first *constructive* algorithm to generate iBGP configurations with useful correctness guarantees, while scaling better than a full mesh. This algorithm answers one of the several questions posed by Feamster *et al.* in their paper on open problems in inter-domain routing [9]. The algorithm admits an efficient and practical implementation, and can easily be integrated into tools that produce router configuration code. We believe that `BGPSep` can eliminate some hard-to-debug network problems that network operators face.

Acknowledgments

We thank Nick Feamster, Robert Morris, and Dina Katabi for useful discussions on this work and for comments on drafts of this paper. This work was supported by the National Science Foundation under Cooperative Agreements CNS-0225560 and CNS-0520241, and by a Cisco URP Grant. The views expressed in this paper are not necessarily those of the National Science Foundation or Cisco Systems.

References

[1] Anindya Basu, Chih-Hao Luke Ong, April Rasala, F. Bruce Shepherd, and Gordon Wilfong. Route Oscillations in I-BGP with Route Reflection. In *Proc. ACM SIGCOMM*, pages 235–247, Pittsburgh, PA, August 2002.

[2] T. Bates, R. Chandra, and E. H. Chen. BGP Route Reflection—An Alternative to Full Mesh IBGP. RFC 2796, IETF, April 2000.

[3] R. Callon. Use of OSI IS-IS for Routing in TCP/IP and Dual Environments. RFC 1195, IETF, December 1990.

[4] Kenneth L. Calvert, Matthew B. Doar, and Ellen W. Zegura. Modeling Internet Topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[5] Rohit Dube. A Comparison of Scaling Techniques for BGP. *Computer Communications Review*, 29(3):44–46, July 1999.

[6] Nick Feamster. *Proactive Techniques for Correct and Predictable Internet Routing*. PhD thesis, Massachusetts Institute of Technology, September 2005.

[7] Nick Feamster and Hari Balakrishnan. Correctness Properties for Internet Routing. In *The 43rd Annual Allerton Conference on Communication, Control, and Computing*, Monticello, IL, September 2005.

[8] Nick Feamster and Hari Balakrishnan. Detecting BGP Configuration Faults with Static Analysis. In *Proc. 2nd Symp. on Networked Systems Design and Implementation (NSDI)*, pages 49–56, Boston, MA, May 2005.

[9] Nick Feamster, Hari Balakrishnan, and Jennifer Rexford. Some Foundational Problems in Interdomain Routing. In *Proc. 3rd ACM SIGCOMM HotNets Workshop*, pages 41–46, San Diego, CA, November 2004.

[10] Nick Feamster, Jared Winick, and Jennifer Rexford. A Model of BGP Routing for Network Engineering. In *Proc. ACM Sigmetrics*, pages 331–342, New York, NY, June 2004.

[11] Timothy Griffin and Gordon T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *Proc. 10th IEEE International Conference on Network Protocols*, pages 90–99, Paris, France, November 2002.

[12] Timothy G. Griffin and Gordon Wilfong. On the correctness of IBGP configuration. In *Proc. ACM SIGCOMM*, pages 17–29, Pittsburgh, PA, August 2002.

[13] Oliver Heckmann, Michael Piringer, Jens Schmitt, and Ralf Steinmetz. On Realistic Network Topologies for Simulation. In *Proc. ACM SIGCOMM MoMeTools Workshop*, pages 28–32, Karlsruhe, Germany, August 2003.

[14] Enhanced IGRP. `http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/en_igrp.htm`.

[15] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring Link Weights Esing End-to-end Measurements. In *Proc. 2nd ACM SIGCOMM Internet Measurement Workshop*, pages 231–236, Marseille, France, 2002.

[16] G. Malkin. RIP version 2. RFC 2453, IETF, November 1998.

[17] J. Moy. OSPF Version 2. RFC 2328, IETF, April 1998.

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

[19] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, IETF, January 2001.

[20] Daniel A. Spielman and Shang-Hua Teng. Spectral Partitioning Works: Planar Graphs and Finite Element Meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[21] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring ISP topologies with Rocketfuel. *IEEE/ACM Trans. on Networking*, 12(1):2–16, 2004.

[22] P. Traina, D. McPherson, and J. Scudder. Autonomous System Confederations for BGP. RFC 3065, IETF, February 2001.