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 Border Gateway Protocol (BGP), the current inter domain routing protocol in the Internet, 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). This paper focuses on the construction of an iBGP session configuration that guarantees two correctness properties—*loop-free forwarding paths* and *complete visibility* to all eBGP-learned best routes—while attempting to minimize the number of iBGP sessions (for scalability) and ensuring that the constructed configuration guarantees the two correctness properties even in the face of link failures and IGP path changes. Our algorithm constructs an iBGP configuration based on route reflectors [2], a commonly used way to control the number of iBGP sessions. The algorithm, `BGPSep`, uses the notion of a *graph separator*, a (small) set of nodes that partition a graph into connected components of roughly equal sizes, recursively applies this idea to the connected components, and produces a route reflector hierarchy and the associated iBGP sessions. We prove that `BGPSep` guarantees the desired correctness properties, and evaluate an implementation of the `BGPSep` algorithm on several real-world and simulated network topologies. Across these topologies, we find that the number of iBGP sessions with `BGPSep` is a factor of 2.5 to 5× smaller than with a "full mesh" iBGP, while guaranteeing the desired correctness properties.**

I. INTRODUCTION

The Internet is made up of many Autonomous Systems (ASes), which are networks or groups of networks under a common administration and with a common routing policy. 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 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 [13], and RIP [16] don't handle the required scale well, and don't offer the policy expressiveness offered by BGP.

To solve the intra-AS route dissemination problem, the designers of BGP proposed the use of a "full mesh" intra-AS BGP 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 advantage of the full-mesh iBGP configuration is that it satisfies two important correctness properties:

- P1 **Loop-free forwarding:** After the dissemination of eBGP learned routes, that the resulting routes (and the subsequent forwarding paths of packets sent along those routes) picked by all routers are free of deflections and forwarding loops [5], [11].
- P2 **Complete visibility:** The dissemination of information amongst the routers is "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 each eBGP router in the AS.

These two desirable properties, however, are achieved at significant cost in the full-mesh iBGP configuration, because the number of iBGP sessions in an AS could be quite large. In many Internet Service Provider (ISP) networks, the number of sessions per router is several hundred, and the number of sessions in the AS is a few hundred thousand. An AS with e eBGP routers and i interior routers has $e(e-1)/2 + ei$ iBGP 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 [21], [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 two correctness properties (P1 and P2) described above can be violated ([5], [11], [6], [7]), leading to routing loops, forwarding loops, and sub-optimal paths. Moreover, as explained in Section II, route reflection does not preserve properties P1 and P2 when IGP costs change, or when links or routers fail.

Previous work about iBGP configuration correctness [11] gives sufficient conditions to *check* if a given iBGP configuration is correct. However, there has not been any work on how to *generate* correct iBGP configurations that are scalable. In this paper, we describe the design, implementation, and evaluation of the `BGPSep` algorithm to generate an iBGP configuration that guarantees that properties P1 and P2 hold under static conditions, and when IGP costs change or failures occur. `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 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 that partition a graph into roughly equal-sized connected components. We observe that if routers in different connected components each have iBGP sessions with all routers in a separator, then each router will learn the same information as when it is directly connected to all the eBGP routers in the other components. That's because the shortest path (in fact, all paths) between nodes in different connected components must traverse one or more nodes in the separator. `BGPSep` applies this idea recursively to construct a route reflector hierarchy that has a relatively small number of sessions. `BGPSep` is practical—efficient algorithms for graph separators using spectral techniques are known [19], and our implementation uses this method (Section V). The run time of the spectral partitioning algorithm is linear in the number of nodes in the graph. `BGPSep` takes between 5 seconds and 60 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, 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. BGP's route selection process involves the comparison of the following attributes in the given order: local preference, AS path length, 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.[1] 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] improves the scalability of intra-AS route dissemination. 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, in order to guarantee loop-free forwarding. Route reflectors, however, have different rules. The internal peers of a route reflector are divided into two groups: (1) client peers and (2) non-client peers. When a route reflector receives a route from an iBGP peer, it selects the best route based on the rules mentioned above. After the best path is selected, it must do the following depending

[1] In the discussions that follow, we do not deal with the case of tied IGP costs separately. We implicitly assume a deterministic tie-breaking mechanism between routers with same IGP path costs.

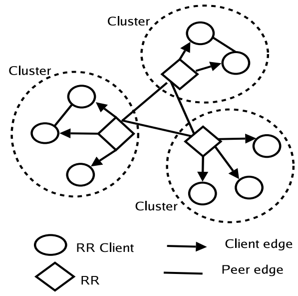

Figure 1. iBGP using route reflection

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 (2) A route from a client peer is reflected to all the non-client peers and also to the client peers. Hence, the client peers are not required to be fully meshed, reducing the number of iBGP sessions.

Typically, the route reflectors in an AS are arranged as clusters. Every route reflector cluster has one or more route reflectors. The other routers in the cluster are made the clients of the route reflectors in the cluster. All the top level route reflectors (i.e., the route reflectors which are not clients of any other route reflector) in an AS are fully meshed with each other to ensure that all route reflectors hear of the best routes of all routers in the system [6].[2] 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 iBGP configuration with a single level hierarchy of route reflectors is shown in Figure 1. Note that this graph is different from the physical IGP graph of the network.

C. Problems with Route Reflection

The iBGP configurations that use route reflection do not *provably* satisfy the properties P1 and P2. We now describe how these properties may be violated in some iBGP configurations.[3]

[2]This property is called visibility by the authors of [6].

[3]In the discussion that follows, the set of routes and egresses, will, by default, 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.

a) Loop-free forwarding: Route reflection iBGP configurations are susceptible to forwarding anomalies like deflections and forwarding loops [11], which make such configurations hard 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], [11].

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. However, 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 loop forever.

Such forwarding anomalies would never occur in a full mesh iBGP configuration where every router learns of the best routes from all egresses and thus picks the egress closest to itself, from amongst the egresses that have an otherwise equally good route (i.e., equally good upto the step of comparing the IGP cost to the egress) to a given destination.

b) 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. For example, in Figure 2, router C1 chooses the route through R1. Had it learned of all the eBGP routes, however, it would have picked the route through R2 because C1 has a lower IGP cost to R2.

If the property of complete visibility is violated, the system will fail to implement "hot-potato" routing and this might cause network resources to be wasted. Also, predicting the outcome of the complex BGP decision process become trivial with an AS

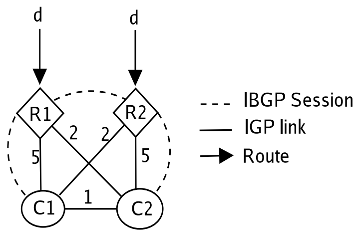

Figure 2. An incorrect iBGP configuration

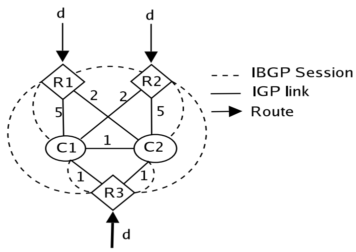

Figure 3. An iBGP configuration using route reflectors that has low fault tolerance

when complete visibility is ensured, 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 [9].

c) Robustness to IGP changes: In arbitrary iBGP configurations with route reflection, correctness properties like loop-free forwarding can be violated in the face of IGP link cost changes or IGP failures. For example, consider the iBGP configuration shown in Figure 3. This IGP topology is similar to the topology in Figure 2, except for 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!

Needless to say, setting up correct iBGP configurations with route reflection in real world networks with a large number of BGP routers is a non-trivial task. Not much 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 are 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. We need to come up with a way to configure iBGP using route reflection that gives us *provable* guarantees about loop-free forwarding, complete visibility and robustness like the full-mesh iBGP configuration, without losing the scalability offered by route reflection.

III. THE BGPSEP ALGORITHM

A. The Invariants

Recall that for each destination, every router in an AS picks the best route from amongst the routes that it learns and forwards packets towards the egress router of the best route of that destination. We translate the two correctness conditions P1 and P2 to invariants INV1 and INV2 respectively, which which are sufficient conditions for the properties to be satisfied.

- **INV1:** Let d denote any destination. If a router A picks a router B as its egress for a destination d, then every router on the IGP shortest path from A to B should also pick B as its egress for the destination d. (Otherwise, there would be deflections when packets are forwarded from A towards B and thus P1 would be violated.)
- **INV2:** Every router picks its closest egress from amongst the set of egresses that have routes with the best path attributes, breaking any ties deterministically.

Our solution constructs an iBGP configuration that satisfies INV1 and INV2 and thus satisfying P1 and P2. Our solution uses the graph theoretic notion of graph separators to generate iBGP configurations.

B. Graph separators

A graph separator is a set of nodes that 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, $|V| = n$, 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, we observe that any path beginning in a component G_i and ending in a different component G_j must have one or more nodes from S. Our solution uses this property of graph separators.

The problem of finding the optimal graph separators is NP–hard in general. However, fast and practical algorithms for finding small separators are known for many families of graphs.[4]. We use the *spectral partitioning algorithm* described in [19] to find graph separators. The run time of the spectral partitioning algorithm is linear in the number of nodes in the graph.

C. A simple algorithm

Let G denote the IGP subgraph induced by the eBGP routers and V denote the set of eBGP routers.[5] As noted in Section II, P1 and P2 can be satisfied if we use a full mesh iBGP for G. We now describe a naive iBGP configuration that satisfies P1 and P2 without requiring a full mesh iBGP.

- **Step 1:** Consider a graph separator S of G. Make all the routers in S as route reflectors.
- **Step 2:** For every $u, v \in S$, configure routers u and v as iBGP peers. This is in accordance with the principle of fully meshing the top level route reflectors in every AS, as described in Section II.
- **Step 3:** Let G_1 and G_2 be the two components into which S separates G.[6] Make *each* BGP router in G_1 and G_2 a route reflector client of *every* route reflector in S.
- **Step 4:** Connect all routers in G_1 and G_2 with each other i.e., construct a full mesh iBGP within G_1 and G_2.

[4]Provable guarantees on the size of the separator have been shown for many families of graphs including planar graphs ($O(\sqrt{n})$) and bounded genus graphs ($O(\sqrt{gn})$) [19] where g is the genus of the graph. (The genus of a graph quantifies how close to planar it is.)

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

[6]The arguments hold even if there are more than two components.

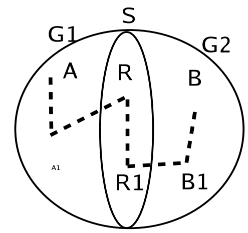

Figure 4. Proving the invariants on a simple iBGP configuration

We now prove informally that the invariants INV1 and INV2 hold on this strawman iBGP configuration. Let B denote the closest egress to A with a route to d with the best path attributes.

Claim 1: A learns of the route via B and thus picks B as its egress for destination d. (INV2)

Proof: If A and B are in the same component (say, G_1), then they have an iBGP session between them (since each component is fully meshed) and thus A will learn of the route via B. Suppose A and B are in different components. Then, the shortest path between A and B should pass through S, as shown in Figure 4. Since all nodes in G_1 and G_2 are clients of all route reflectors in S, there exists a route reflector R (in fact, in this example, there exist two route reflectors - R and $R1$) 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 also. Thus R will choose the route via B as its best route and reflect it to all its clients. Thus A will learn of the route through B. ■

Claim 2: Every router on the shortest path from A to B will also choose B as its egress for destination d. (INV1)

Proof: If A and B are in the same component then the shortest path between A and B would also be in the same component. Then, INV1 will hold by the same logic that P1 would hold in a full mesh iBGP (as proved in Section II). Suppose A and B are in different components. Suppose the shortest path between A and B is A–$A1$–R–$R1$–$B1$–B, as shown in Figure 4. We will now prove that all routers on the shortest path also choose B as their egress. First, we observe that if B is the closest egress to A then B is the closest egress to all routers on the shortest path from A to B. Second, all routers on the shortest path would learn of the route via B. (R and $R1$ learn of the route since they are route reflectors of B. They choose this as their best route and reflect it

to all their clients and thus $A1$ and $B1$ learn it too.) From these two observations, we can conclude that all routers on the shortest path from A to B also pick B as their egress from destination d. ■
Thus, our strawman design ensures that P1 and P2 are satisfied on the resulting iBGP configuration. Though we have reduced the number of iBGP sessions compared to the full mesh iBGP (e.g., routers in G_1 and G_2 no longer need to connect to each other, they only need to connect to the route reflectors in S), this design still uses a full mesh within each of the individual components G_1 and G_2. To further reduce the number of iBGP sessions, we can apply the separator idea on G_1 and G_2 recursively. The recursion can terminate when the components are small enough to be fully meshed. Note that INV1 and INV2 continue to hold at each stage of the recursion.

D. The Complete Algorithm

Algorithm 1 shows the 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. The algorithm is a centralized algorithm. 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 the value of t is "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 the value of t is "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 nodes. The algorithm uses a procedure `Graph-Separator`, which is a graph partitioning algorithm (e.g., the algorithm described in [19]) that takes a graph G and returns a graph separator S.

Note that while the recursion in Algorithm 1 terminates when each component has one or two nodes, the algorithm can be modified to terminate the recursion at an earlier stage and fully mesh the remaining components. Infact, the maximum number of levels of recursion (which is also equal to the number of levels in the resulting route reflector hierarchy) can be a user–defined parameter.

E. An example

We now give a simple example to illustrate the `BGPSep` algorithm. Consider a network with ten BGP routers, as shown in Figure 5. Step 1 of the algorithm chooses the set of nodes $S = \{c, f\}$ to separate the graph into two components $G_1 = \{a, b, d, e\}$ and $G_2 = \{g, h, i, j\}$. In step 2, we fully mesh the set of route reflectors. Thus the set I of iBGP sessions will be $\{(c, f, \text{peer})\}$. In step 3, we make each node in G_1 and G_2 a client of every route reflector in S.

```
BGPSep Input: IGP Graph G, set V of BGP routers
Output: Set I of iBGP sessions
if |V| = 1 then
    /* Recursion base case */
    I = ∅;
    return I;
else if |V| = 2 then
    /* Recursion base case */
    {u, v} ← V ;
    I = {(u, v, 'P')};
    return I;
else
    /* Step 1: Choose a graph separator S. Nodes in S
       are the route reflectors. S ⊆ V */
    S ← Graph-separator(G) ;
    G_1, ..., G_m ← subgraph on V − S;
    /* Step 2: Fully mesh the set of route reflectors */
    foreach u, v ∈ S, u ≠ v do
        I = I ∪ {(u, v, peer)};
    end
    foreach G_i do
        /* Step 3: Make every node in each component G_i a
           route reflector client of every route reflector */
        foreach u ∈ G_i, v ∈ S do
            I = I ∪ {(u, v, client)};
        end
        /* Step 4: Recursively apply the algorithm over
           each component */
        I_i = BGPSep(G_i) ;
        I = I ∪ I_i ;
    end
end
return I;
```

Algorithm 1: Recursive separator algorithm to output an iBGP configuration

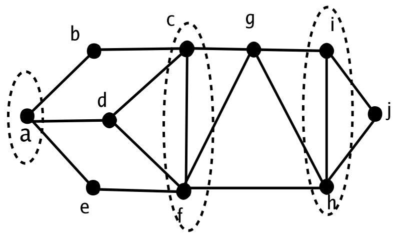

Figure 5. An IGP topology to illustrate the algorithm

In step 4, we recursively apply 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 is a singleton here. In step 3, we add 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} since they have one node. Similarly, we recurse over component G_2 also. The separator $S_2 = \{i, h\}$ is chosen in step 1. We add the iBGP session (i, h, peer) in step 2 and the iBGP sessions (g, i, client), (g, h, client) (j, i, client), (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, 5 route reflectors and 25 iBGP sessions, compared to the 45 iBGP sessions in a full mesh iBGP configuration.

F. Variants

In this section, we describe two variants of the `BGPSep` optimized for different types of networks. Note that, in all these cases, the original algorithm would still be valid and the variants are only for convenience.

1) Networks with internal routers: In the case when the network contains internal routers (which do not receive any external routes), the internal routers need not mesh with each other (because they do not learn any external routes). Thus the algorithm `BGPSep`, when run over the entire topology of internal and egress routers, may establish some unnecessary iBGP sessions. We now describe how `BGPSep` can be modified to be used in networks with a large number of internal routers. In Step 1, we first find a set S of nodes to first 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, we recurse on the component containing the egress routers only since the internal routers need not have iBGP sessions with each other. Note that using our algorithm is inefficient if the number of egress routers is very small. If $|S| > |G_{ext}|$, it would be easier 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 [20]. Every PoP has some access routers which connect to customer networks and one or two (for redundancy) backbone routers that connect the PoP to the rest of the network.[7] A route reflection iBGP 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. The route reflectors at the top level in the hierarchy are fully meshed. Additionally a route reflector hierarchy can also be built over the route reflectors in the backbone.

Running our `BGPSep` algorithm on the entire IGP topology of an ISP outputs an iBGP configuration that might look very different from conventional iBGP configurations. For example, in our construction, it might so happen that an access router might have to connect to multiple route reflectors in different PoPs, which might be too much load on an access router. Hence we propose a variant `BGPSep-Backbone` of our original algorithm that is more suited for ISP-like backbone networks.

The backbone of an ISP consists of backbone routers which connect all the PoPs and which also serve as route reflectors to the access routers in the PoPs. Since the top level route reflectors must all be fully meshed, we expect the ISP backbone to have a full mesh iBGP or a route reflector hierarchy. Thus, we can apply our algorithm on just the backbone (as opposed to on all the BGP routers) and establish an route reflector hierarchy over the the backbone. The backbone routers in each PoP are then configured as route reflectors to the access routers in each PoP. Configuring a route reflector hierarchy according to the separator algorithm only ensures that every shortest path between two backbone routers has a route reflector hierarchy on it. However, the same

[7]These PoPs typically correspond to areas in OSPF.

property does not hold for shortest paths between access routers. While shortest paths between access routers in two different PoPs traverse the backbone (a property of backbone–like networks) and thus has a hierarchy route reflector on it, the same property does not hold for intra-PoP shortest paths which do not traverse the backbone. We thus need to fully-mesh all the access routers within a PoP.

IV. Proof of correctness

In this section, we formally prove that the iBGP configuration output by `BGPSep` algorithm satisfies the properties P1 and P2 described in Section I. We also prove that properties P1 and P2 hold in the face of IGP link cost changes, node failures and link failures.

A. Proof of Complete Visibility

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 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. Let I denote the set of iBGP sessions output by the `BGPSep` algorithm.

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 with the definition if a *signaling chain*.

Definition 3: A signaling chain between two routers A and B is defined as a set of routers $A(=R_0), R_1, R_2, \ldots, R_r, B = (R_{r+1})$ such that (i) R_i is a route reflector and (ii) atleast one of R_{i+1} or R_{i-1} is a route reflector client of R_i for $i = 1 \ldots r$.

Lemma 4: If there exists a signaling chain between routers A and B and B is an egress router for a destination d, then A learns of the best route via B for destination d.

Proof: For $i = 1 \ldots r$, we claim that R_i propagates the routes learned from R_{i+1} to R_{i-1}. This claim is true because, 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 to all its peers. 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 from all peers to 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 5: In the iBGP configuration output by the `BGPSep` algorithm, for any destination d, every router $A \in V$ learns of the best route via $\text{Egress}_d(A)$.

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 our the construction in our `BGPSep` algorithm that this shortest path should pass through a set of recursively produced graph separators. Since the graph separators are configured as route reflectors and the components (separated by the separator) as route reflector clients, 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. (As an example, $A-R-B$ is a signaling chain on the shortest path $A - A1 - R - R1 - B1 - B$ between A and B in Figure 4. Hence, it follows from Lemma 4 that A learns of the route through B. ∎

The following theorem is a direct consequence of Lemma 5.

Theorem 6: The iBGP configuration output by our `BGPSep` algorithm satisfies the property of complete visibility.

B. Proof of Loop–free forwarding

The proof in this section use the result of Theorem 6 to prove that the iBGP configuration output by our `BGPSep` algorithm satisfies the property of loop–free forwarding.

Theorem 7: The iBGP configuration output by our `BGPSep` algorithm satisfies the property of loop–free forwarding.

Proof: From Theorem 6, 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. So there are no deflections when packets are forwarded along the shortest path from A to B, thus guaranteeing loop–free forwarding. ∎

C. *Proof of Robustness to IGP changes*

Lemma 8: The iBGP configuration output by our `BGPSep` algorithm is not affected by change in IGP link costs.

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

Lemma 9: The iBGP configuration output by our `BGPSep` algorithm satisfies the properties of loop–free forwarding and complete visibility in the face of IGP failures (e.g., node and link failures).

Proof: The graph separator of a graph is a separator of any IGP subgraph of the graph. Thus when nodes and links fail, the properties of loop–free forwarding and complete visibility hold on the remaining graph by the same logic as they did on the original IGP graph. ■

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

D. *Caveats of the algorithm*

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). In such a case, we can no longer assume that the nodes in the graph separators are all route reflectors, and thus all our correctness guarantees break down. By similar reasoning however, the correctness guarantees of the full mesh iBGP also become invalid.

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

Also, the `BGPSep` algorithm is not an incremental algorithm, and the algorithm must be re-run and new separators computed when new nodes or links are provisioned in the network.

V. Implementation

We implemented the `BGPSep` algorithm in about 100 lines of Matlab code. The program reads the IGP graph from a file and outputs the iBGP sessions to a file. We implemented the spectral partitioning algorithm [19] to find graph separators. The run time of the spectral partitioning algorithm is linear in the number of nodes in the graph. The algorithm took between 5 seconds and 60 seconds to run for real network topologies having between 80 and 300 nodes.

As part of future work, we propose to develop a tool that takes the router configuration files as input, infers the IGP topology from the configuration files and outputs the lines of configuration code corresponding to the iBGP sessions for each router. Thus, our tool can ease the task of network configuration and management.

AS	Name	Number of nodes	Number of edges
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.

VI. Evaluation

In this section, we evaluate the performance of the separator algorithm 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 our algorithm on synthetic topologies generated using GT-ITM [4]. The GT-ITM parameters in the graphs were set according to the suggestions in [12]. The key aspects on which we evaluated the iBGP configuration produced by the `BGPSep` algorithm are (a) scalability in terms of the number of iBGP sessions and the number of route reflectors (b) robustness to iBGP failures.

A. *Scalability*

In this section, we show that the iBGP configuration produced by our algorithm is scalable in terms of the number of iBGP sessions and the number of route reflectors. 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.

1) Comparison with full mesh iBGP: In this section, we compare N_{ibgp} against the number of iBGP sessions in the full mesh iBGP for various networks.

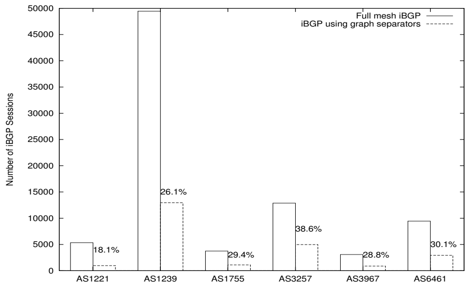

Figure 6. Comparison with full mesh - real ISP topologies.

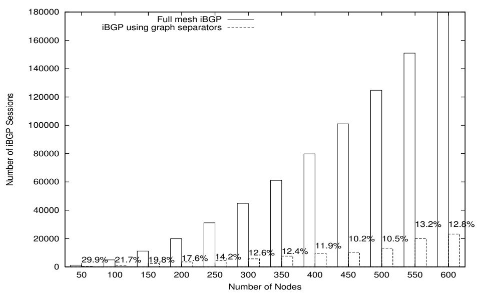

Figure 7. Comparison with full mesh - GT-ITM topologies.

We use the 6 ISP backbones from Rocketfuel and synthetically generated internet topologies from GT-ITM. We assume that all the nodes in the topology are external BGP routers. The corresponding results are shown in Figures 6 and 7. We see that the iBGP configuration produced by `BGPSep` results in a $2.5\times$ to $5\times$ reduction in the number of iBGP sessions on Rocketfuel ISP topologies and a $5\times$ to $10\times$ reduction on GT-ITM topologies.[8]

2) Scaling of the number of iBGP sessions with the number of nodes: We evaluate how N_{ibgp} scales with the number of routers, n. Let $n_{original}$ denote the number of routers in a network. For each of the network topologies, we picked 10 random subgraphs with a n nodes where $n = \frac{n_{original}}{2^i}, i = 0 \ldots 4$, and ran `BGPSep` on these subgraphs. The resulting mean and standard deviation of the number of iBGP sessions over 10 runs for AS 1239 and GT-ITM topologies is shown in Figure 8 on a log–log scale. The results for the other ISPs were similar. The slope of this graph gives us the value of the k such that $N_{ibgp} = O(n^k)$. The value of this slope for various ISPs is shown in Table II.

[8]Another interesting number to compare against would have been the actual number of iBGP sessions in the iBGP configurations of these networks. However such figures are not made public by the ISPs.

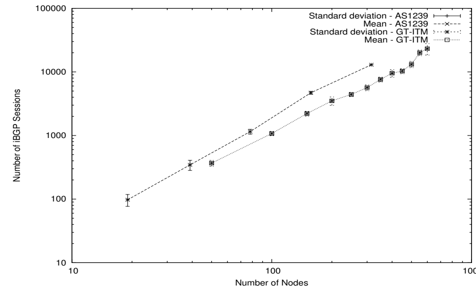

Figure 8. Number of iBGP sessions vs n: AS1239 and GT-ITM.

1221	1239	1755	3257	3967	6461	GT-ITM
1.75	1.74	1.82	2.11	1.95	1.73	1.69

Table II

SLOPE OF LOG(NUMBER OF IBGP SESSIONS) VS LOG(N). THE TOP ROW SHOWS THE AS NUMBER OF THE ISP.

3) Number of route reflectors: Another key aspect of scalability is the number of route reflectors and the number of levels in the 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 III. We report the number of route reflectors at the top–most level of the hierarchy, because the top–level route reflectors have the largest number of clients. We observe that although a substantial number of nodes are route reflectors, the number of top–level route reflectors (which have the most complex configuration) is relatively small.

4) Scaling of the number of route reflectors: In this section, we observe how the number of route reflectors and the number of top–level route reflectors scale with the number of nodes in the network. As in Section VI-A.2, we construct various subgraphs for the Rocketfuel topologies, varying

AS	Nodes	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 III

NUMBER OF ROUTE REFLECTORS IN ROCKETFUEL ISP TOPOLOGIES

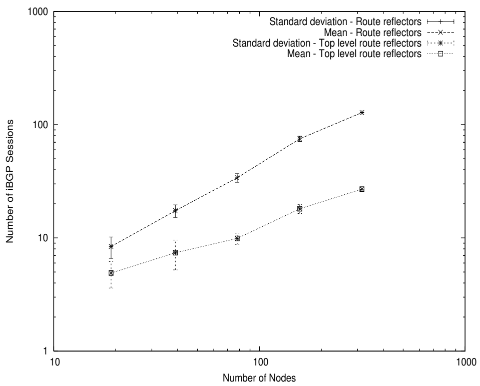

Figure 9. Number of route reflectors vs n - AS1239

Parameter	Slope
Number of route reflectors	0.95
Number of top level route reflectors	0.53

Table IV

SLOPE OF NUMBER OF ROUTE REFLECTORS VS n FOR AS 1239

the number of nodes. We construct 10 random subgraphs for each value of the number of nodes. The mean and the standard deviation of the number of route reflectors and the number of top–level route reflectors is shown in Figure 9. The slope of these graphs is shown in Table IV. We report the results on the topology of AS1239. The results on the other topologies were similar.

B. Robustness

As proved in Section IV, the iBGP configuration produced by `BGPSep` is resilient to IGP failures once the IGP converges to a loop–free topology. It does not guarantee robustness in the face of iBGP failures. We randomly simulate iBGP failures on the iBGP configuration 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 had a forwarding anomaly (like deflections). For high values of failure rates, we also observed instances of persistent forwarding loops in our iBGP topology. We report the number of configurations N_{FL} that had 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

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 V

ROBUSTNESS TO IBGP FAILURES

a given failure rate to the path length taken by the packet in a configuration that satisfies complete visibility. Note that, when complete visibility is violated, the routers no longer choose their closest egress available, and thus $r_{complete}$ exceeds 1.

The values of $r_{complete}$, $f_{incorrect}$ and N_{FL}, for failure rates of 0%, 5%, 20%, 40% and 60%, using the iBGP configuration produced on AS1221 of the Rocketfuel data, are shown in Table V. We conclude that while our iBGP configuration is tolerant to low iBGP session failure rates, it is not resilient to very high failure rates. We believe however, that this lack of resilience is not a significant practical problem for two reasons: first, such high failure rates are highly unlikely, and second, we are no worse than the status quo (as explained in Section IV).

VII. RELATED WORK

The possibility of the occurrence of forwarding loops with Route reflection was first reported by Dube *et. al* in [5]. The property of loop–free forwarding was studied in detail by Griffin and Wilfong [11] (it is called "forwarding correctness" by the authors). The authors prove that verifying whether an arbitrary iBGP configuration is forwarding correct is NP–hard. The authors also describe 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.[9] The iBGP configuration generated by our algorithm does not satisfy either of the two sufficient conditions – (i) we do not require that a route reflector prefer client routes to non–client routes and (ii) the signaling chain in our solution is only a subset of the shortest path between any two routers. Yet, we guarantee forwarding correctness. Thus, we believe that the

[9] A signal path, as defined in [11], is a more general version of the signaling chain described in Section IV, and is defined as a path along which a routing message can propagate.

correctness conditions in [11] are quite strong. Also, unlike [11], our work presents a *constructive* algorithm to generate iBGP configurations with correctness guarantees.

Our work looks at the correctness properties of iBGP *after* the path assignment has converged. There are other aspects of correctness like convergence of the BGP protocol itself that we have not examined in our work. 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 [11] study the conditions under which the BGP configuration converges to a stable path assignment. The authors also study the problem of oscillations in BGP caused by the MED attribute [10].

Previous work [6], [7] has identified routing anomalies like forwarding loops in iBGP configurations. [8] identifies many open problems in inter–domain routing, one of them being the the problem of constructing scalable, yet correct iBGP configurations.

The authors of [14] implement the idea of automatically generating router configuration lines from a network–wide specification of routing policies. This idea is similar to our implementation scheme of automatically generating the iBGP configuration for configuring route reflectors.

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. For instance, a recent study of routing configurations found iBGP signaling problems in many different autonomous systems [7].

This paper develops sufficient conditions for two important correctness properties, loop-free forwarding and complete visibility. (These conditions are inspired by, but distinct from, the conditions developed in [11].) These sufficient conditions allow us to develop and prove the correctness of a separator-based iBGP configuration algorithm, `BGPSep`. The algorithm 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 these two correctness properties. The resulting iBGP configuration retains these correctness properties even in the face of 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 and between $3\times$ and $10\times$ fewer sessions in the simulated topologies.

To our knowledge, `BGPSep` is the first algorithm that constructs provably correct (in terms of properties P1 and P2) and scalable (compared to full-mesh) iBGP configurations. The algorithm admits an efficient and practical implementation, and can easily be integrated into tools that produce router configuration code. We believe that the algorithm can eliminate various hard-to-debug network problems that network operators face today.

Acknowledgments

We thank Nick Feamster, Robert Morris and Dina Katabi for several useful discussions on this work and for comments on drafts of this paper. This work was based in part upon support from the National Science Foundation under Cooperative Agreement CNS-0225560 and 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 *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 235–247, New York, NY, USA, 2002. ACM Press.

[2] T. Bates, R. Chandra, and E. H. Chen. BGP route reflection - an alternative to full mesh IBGP. RFC 2796, Internet Engineering Task Force, April 2000.

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

[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 and Hari Balakrishnan. Towards a Logic for Wide-Area Internet Routing. In *ACM SIGCOMM Workshop on Future Directions in Network Architecture*, pages 289–300, Karlsruhe, Germany, August 2003.

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

[8] Nick Feamster, Hari Balakrishnan, and Jennifer Rexford. Some Foundational Problems in Interdomain Routing. In *3rd ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets)*, pages 41–46, San Diego, CA, November 2004.

[9] Nick Feamster, Jared Winick, and Jennifer Rexford. A Model of BGP Routing for Network Engineering. In *ACM Sigmetrics - Performance 2004*, New York, NY, June 2004.

[10] Timothy Griffin and Gordon T. Wilfong. Analysis of the MED Oscillation Problem in BGP. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 90–99, Washington, DC, USA, 2002. IEEE Computer Society.

[11] Timothy G. Griffin and Gordon Wilfong. On the correctness of IBGP configuration. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 17–29. ACM Press, 2002.

[12] Oliver Heckmann, Michael Piringer, Jens Schmitt, and Ralf Steinmetz. On realistic network topologies for simulation. In *MoMeTools '03: Proceedings of the ACM SIGCOMM workshop on Models, Methods and Tools For Reproducible Network Research*, pages 28–32, New York, NY, USA, 2003. ACM Press.

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

[14] Olaf Maennel, Anja Feldmann, Christian Reiser, Ruediger Volk, and Hagen Boehm. Network-Wide Inter-Domain Routing Policies: Design and Realiz ation. In *NANOG 34*, Seattle, WA, May 2005.

[15] Ratul Mahajan, Neil Spring, David Wetherall, and Tom Anderson. Inferring link weights using end-to-end measurements. In *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 231–236, New York, NY, USA, 2002. ACM Press.

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

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

[18] Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4). RFC 1771, IETF, March 1995.

[19] 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.

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

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