On Constructing Correct and Scalable iBGP Configurations

by

Mythili Vutukuru

B. Tech., Indian Institute of Technology, Madras (2004)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2006

Author .....
Department of Electrical Engineering and Computer Science
August 11, 2006

Certified by .....
Hari Balakrishnan
Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

On Constructing Correct and Scalable iBGP Configurations

by

Mythili Vutukuru

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2006, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

BGP (Border Gateway Protocol), the Internet's current inter-domain routing protocol, has two modes of operation: eBGP (external BGP, used to exchange routing information between autonomous systems (ASes)), and iBGP (internal BGP, used to propagate that information about external destinations to other BGP routers within an AS). Full-mesh iBGP and iBGP with route reflection are the two most common methods of configuring iBGP. Although a full-mesh iBGP guarantees correct and predictable routing, it requires a large number of iBGP sessions—approximately quadratic in the number of BGP routers. Such configurations do not scale well in the number of BGP routers in the AS because of the memory, bandwidth and CPU overhead involved in exchanging routes over a large number of iBGP sessions at each router. Hence configurations based on *route reflectors* are commonly used for intra-AS route dissemination in large ASes. However, researchers have found that configuring route reflectors in an unprincipled fashion can result in routing anomalies like forwarding loops and sub-optimal paths.

Although previous work on iBGP configuration correctness gives sufficient conditions to check if a given iBGP configuration is correct, the problem of constructing correct and scalable iBGP configurations using route reflection has not received much attention. This thesis proposes and analyzes the first (to our knowledge) algorithm to construct iBGP session configurations that are both correct and more scalable than a full-mesh iBGP. Our algorithm, `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 network topologies and find that iBGP configurations generated by `BGPSep` have between 2.5 to 5 times fewer iBGP sessions than a full-mesh.

Thesis Supervisor: Hari Balakrishnan
Title: Professor

Acknowledgments

I thank my advisor Hari Balakrishnan for his excellent mentoring, encouragement and guidance during the past two years. I am also grateful to him for his care and patience while I overcame the initial glitches of graduate student life.

This work, which started as a class project, greatly benefited from the ideas and constructive criticism of the other members of my team, Paul Valiant and Swastik Kopparty, and the course instructors, Robert Morris and Dina Katabi.

I would like to thank Nick Feamster for many insightful discussions, both on this work as well as on many topics related to Internet Routing. I would also like to thank Mike Walfish for comments on a presentation of this work, and for the many things that I learned from him.

This work was supported by the National Science Foundation under Cooperative Agreements CNS-0225560 and CNS-0520241, and by a Cisco URP Grant. I am grateful for the same.

I am, and forever will be, grateful to my parents, family and friends for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we present an algorithm to construct iBGP configurations that emulate the full-mesh iBGP, but are more scalable in terms of the number of iBGP sessions.

1.1 Motivation

The Internet is a collection of independently operated Autonomous Systems (ASes) each of which is a network under a common administration. Routers in an AS run an Interior Gateway Protocol (IGP) to maintain connectivity amongst themselves. In general, subset of routers in each AS also runs an *inter-domain* routing protocol to maintain connectivity between ASes. The Border Gateway Protocol (BGP) is the de-facto inter-domain protocol used in the Internet today. The BGP routers at the border of an AS (also referred to as *egress* routers[1]) establish BGP sessions with border routers in other ASes to exchange reachability information about external destinations. This mode of operation of BGP is called eBGP (external BGP). An egress router that learns a route to an external destination via eBGP must then disseminate that route to the other egress routers and non-egress (internal) BGP

[1]We will use the terms egress and border router interchangeably.

routers[2] in the AS.

One approach to intra-AS route dissemination is to introduce the routes to external destination prefixes into the IGP. This approach, however, does not work well because common IGPs such as OSPF [16], IS-IS [3], EIGRP [13], and RIP [15] do not handle the required scale well, and do not offer the policy expressiveness offered by BGP.

The most common way to disseminate external routes in an AS is to set up BGP sessions between the BGP routers *in the same AS*, in a mode called *internal BGP* (iBGP). An iBGP session between two iBGP "peers" runs as a TCP session between the routers and relies on the AS's underlying IGP to achieve connectivity.

BGP routers do not re-advertise the routes learned on one iBGP session over other iBGP sessions in order to avoid the routing messages looping forever between the routers. In order to learn all the external routes coming into the AS, every BGP router (be it an egress router or an internal router) needs to establish an iBGP session with every egress router of the AS. This method of configuring iBGP is called a "full-mesh" iBGP configuration.

The full-mesh configuration satisfies the following desirable correctness properties (explained in more detail in Chapter 2):

P1 **Complete visibility:** The dissemination of information amongst the routers is "complete" in the sense that, for every external destination, each router picks the best route[3] from the set of routes learned by all the egress routers in the AS to that destination.

P2 **Loop-free forwarding:**[4] After the dissemination of eBGP learned routes converges, the resulting routes (and the subsequent forwarding paths of packets

[2]Not all BGP routers are egress routers—some BGP routers maintain information about all external destinations and act as gateway routers to the non-BGP routers in the AS, but do not learn routes via eBGP themselves.

[3]We will elaborate on the BGP route selection rules in Chapter 2

[4]P1 subsumes P2 if the IGP enforces shortest path routing (Chapter 2).

sent along those routes) picked by all routers are free of forwarding anomalies like deflections and forwarding loops [5, 11].

P3 **Robustness to IGP failures:** The route dissemination mechanism is robust to node or link failures and IGP path cost changes—such changes do not result in a violation of the correctness properties P1 or 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, which can translate into many thousands of iBGP sessions in large networks with a few hundred BGP routers. The large number of iBGP sessions per router results in a large number of routes in the routing tables[5]. Large routing tables consume more router memory, CPU cycles to compute the best-path from amongst a larger set of routes and bandwidth to exchange the routes. Large number of iBGP sessions are also cumbersome to manage because of the huge manual configuration effort involved in establishing and maintaining them. In addition, the large number of TCP sessions (over which the iBGP sessions run) also impose another scaling limit.

This lack of scalability has long been a problem with the full-mesh configuration, and has led to a few different proposals to configure iBGP in a scalable way ([21], [2]). The most common technique used today is *route reflection* [2], where a subset of BGP routers, called *route reflectors*, re-advertise (or "reflect") routes to a few other routers configured as their *clients*, thus avoiding the need for a full-mesh. Route reflectors reflect only their best route (and not all routes) and the resulting path assignments are often not the same as in a full-mesh iBGP, with the result that the correctness properties of a full-mesh iBGP are no longer guaranteed to hold. Researchers have found that configuring route reflectors in an unprincipled fashion can result in routing anomalies like forwarding loops and sub-optimal paths([5, 11, 8, 6]).

[5]The routing table at a router stores all routes learned for every destination, unlike the forwarding table, which stores only one best route per destination.

1.2 Problem Statement

Although previous work on iBGP configuration correctness [11, 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. This is the problem we address in this thesis. Though there have been proposals suggesting changes to route reflectors [1] to guarantee correct iBGP configurations with route reflection, such proposals require changing every route reflector deployed in the Internet today, and hence are not feasible to implement. We aim to arrive at a solution that is deployable without any changes to routers or end-hosts.

1.3 Contribution

In this thesis, 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 (*i.e.,* IP-level connectivity graph) as input and produces a hierarchical configuration of route reflectors and reflector clients, as well as the associated iBGP sessions (Chapter 3). **BGPSep** does not require any changes to route reflectors. We prove that the iBGP configurations output by **BGPSep** satisfy the desired correctness properties (Chapter 4), and show using an analysis of real-world ISP and synthetic network topologies that the number of iBGP sessions with **BGPSep** is significantly smaller (between a factor of 2.5 and 5× in the ISP topologies) than in a full-mesh configuration (Chapter 5).

BGPSep uses 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—our implementation (Section 5.1) uses an efficient algorithm for finding graph separators using spectral techniques [19]. The run time of the spectral partitioning algorithm is cubic in the number of nodes in the graph. **BGPSep** takes under 5 seconds to produce the iBGP configuration for real-world ISP topologies whose sizes

range from 80 to 300 routers. The iBGP configurations produced by **BGPSep** are also easy to maintain because they need not be recomputed on IGP failures or link cost changes. Because **BGPSep** does not require any changes to today's routers, it can be easily deployed in ISP networks.

Chapter 2

Background and Motivation

2.1 BGP route selection rules

A BGP router in an AS can potentially learn multiple routes via eBGP and iBGP to an external destination prefix. The router then invokes the BGP decision process [17] to select one best route from the set of routes learned 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 the egress router that introduced the route into the AS (the route through the egress router with the lowest IGP cost is preferred). If two routes are tied at the step of comparing the IGP cost to the egress, then some deterministic mechanism such as the a comparison of the router ID of the egress 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. The use of IGP costs in the control plane (*i.e.*, to choose the best BGP route) as well as the data plane (*i.e.*, to choose the IGP path to the egress during forwarding a packet) results in unfortunate interactions between

[1] 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.

IGP and BGP, which is the source of many correctness problems.

2.2 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. Recall that routers do not re-advertise the route learned from one iBGP session onto another to avoid routing loops. Route reflectors, however, have different rules. A route reflector establishes two types of iBGP sessions: a special type of "client" iBGP session with some routers configured as route reflector clients and normal "peer" iBGP sessions with non-clients. When a route reflector receives a route on an iBGP session, it selects the best route using the BGP route selection rules (Section 2.1) and does one of the following depending on the type of the router it received the best route from:

- A best route received from a non-client iBGP peer is reflected to all the clients.
- A best route received from a client is reflected to all the non-client peers and also to the other clients.

By "reflecting" routes to and from clients, route reflectors obviate the need for a full-mesh. Route reflectors also have extra mechanisms [2] to avoid routing loops.

In an iBGP configuration using route reflection, a route reflector reflects *only* its best route (and not all routes it learns) to its clients. A best route chosen by a route reflector may not be the best available route for each of its clients. In particular, if a route reflector chooses between routes based on the IGP cost to their egresses, a route whose egress has the lowest cost to a client may not be the egress with the lowest cost to the route reflector, with the result that the "best" route for a client may never be reflected by the route reflector and hence never be learned by the client. Hence the route assignments in iBGP configurations with route reflection can be different from those in a full-mesh iBGP and the correctness properties of the full-mesh iBGP are not guaranteed to hold.

2.3 Correctness properties

In this section, we describe the correctness properties of complete visibility (P1), loop-free forwarding (P2) and robustness to IGP failures (P3) in more detail. We argue that full-mesh satisfies these properties and provide some examples of how they can be violated in typical iBGP configurations using route reflection.[2]

2.3.1 Complete visibility

An iBGP configuration satisfies complete visibility if every router picks the same routes that it would have picked had it seen the best routes learned by all egress routers for every external destination. It is easy to see that a full mesh iBGP always satisfies complete visibility. However, complete visibility can easily be violated in iBGP configurations with route reflection.

For example, consider the iBGP configuration shown in Figure 2-1. C1 is a client of route reflector R1 and C2 a client of R2. R1 and R2 have a normal peer iBGP session with each other. The IGP and iBGP interconnections are as shown in the figure. Two routes to a destination, tied up to the step of comparing the IGP costs to the egress, 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. In the absence of complete visibility, routers pick sub-optimal paths to forward packets on, causing network resources to be wasted. Moreover, predicting the outcome of the complex BGP decision process—which is useful for modeling BGP and

[2]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.

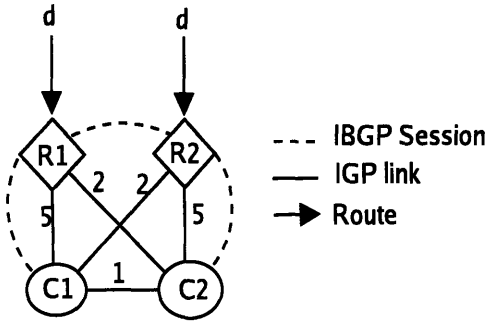

Figure 2-1: An incorrect iBGP configuration.

for traffic engineering [9, 6]—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.

2.3.2 Loop-free forwarding

iBGP configurations with route reflectors are susceptible to forwarding anomalies such as deflections and forwarding loops [11], 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, 11].

For example, again see Figure 2-1. 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.[3] 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 (*e.g.,* full-mesh) if the IGP implements shortest path routing. If a full-mesh iBGP configuration, 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. Forwarding loops also do not occur when routers tunnel packets to egress routers (*e.g.,* using MPLS [18]).

2.3.3 Robustness to IGP failures

The full-mesh iBGP configuration is robust to IGP changes. An iBGP session between two routers runs even in the presence of IGP failures, as long as the two routers are alive and there exists some path between the two routers in the underlying IGP graph. Hence complete visibility and loop-free forwarding continue to hold in a full-mesh iBGP configuration even in the presence of IGP failures. In arbitrary route reflector configurations, however, the afore-mentioned correctness properties can be violated when IGP failures occur.

For example, consider the iBGP configuration shown in Figure 2-2. This IGP topology is similar to the topology in Figure 2-1, except for an additional route reflector R3, of which both C1 and C2 are clients. A route to d arrives at R1, R2 and R3. Both C1 and C2 choose R3 as their next hop for destination d and there are no deflections en route R3. When R3 fails, however, the topology, now equivalent to the one in Figure 2-1, 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!

[3]We are assuming destination-based forwarding.

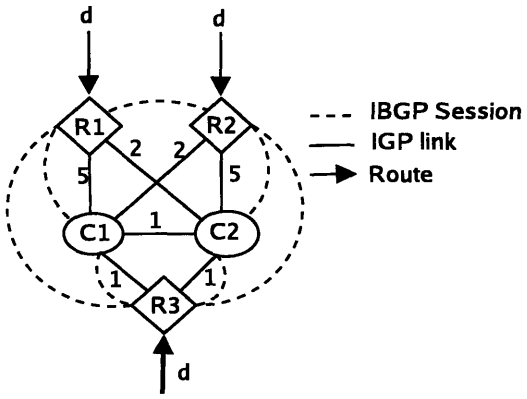

Figure 2-2: An iBGP configuration using route reflectors that has low fault-tolerance.

Although setting up correct iBGP configurations with route reflection in real world networks with a large number of BGP routers is a non-trivial and cumbersome task, little work has been done on automated ways to set up such configurations. Proposals that advocate changing the way route reflection works [1] have not received much encouragement because of the the infeasibility of changing the large base of deployed route reflectors. Today, network operators configure the iBGP largely based on heuristics. While the route reflector configurations today seem to work, there are no correctness guarantees and network operators have to constantly be on the watch for problems. If the resulting system goes into a forwarding loop, they adopt "quick-fix" solutions, such as 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 IGP failures, without losing the scalability offered by route reflection.

2.4 Related Work

The related work broadly falls into three categories.

iBGP correctness: The problem of iBGP correctness has been well-studied in the research community. 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 great detail by Griffin and Wilfong [11], who proved that verifying whether an arbitrary iBGP configuration is "forwarding correct" is NP-hard. They also described a set of sufficient conditions to check if an iBGP configuration is devoid of and forwarding loops. However, their work does not address the problem of actually constructing correct configurations. Feamster and Balakrishnan [7] develop correctness specifications for Internet routing, and formalize the notion of loop-free forwarding.

iBGP routing convergence: Our work deals with the correctness properties of iBGP *after* the path assignments at the routers have converged and does not address the problem of the path assignments at the routers not converging to stable values at all in the first place. BGP convergence is a well-studied problem in the research community. Basu *et al.* [1] study the problem of route oscillations in iBGP with route reflection, 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 iBGP configuration converges to a stable path assignment [11], and examine MED-induced oscillations [10].

Visibility: Feamster and Balakrishnan [7, 8] define the property of visibility as follows: an iBGP configuration has visibility if the existence of a path to a destination implies the existence of *some* route to the destination (which need not necessarily be the best possible route to the destination, as required by the definition of complete

visibility) at every router. They prove that the top-level route reflectors must be in a full-mesh for an iBGP configuration to satisfy visibility. Because the property of complete visibility requires visibility to hold in the first place, `BGPSep` constructs a full-mesh over the route reflectors in every graph separator.

Chapter 3

The BGPSep algorithm

This chapter describes our **BGPSep** algorithm, which takes as input the IGP graph of the network and produces an iBGP configuration that emulates the full-mesh iBGP with fewer iBGP sessions than the full-mesh. We optimize the number of iBGP sessions, because iBGP sessions are the main bottleneck to the scalability of iBGP configurations (as discussed in Chapter 1).

This chapter begins with an overview of the graph-theoretic notion of a graph separator (Section 3.1). We then describe a simplified version of our algorithm (Section 3.2) followed by a more sophisticated and complete version (Section 3.3). We also illustrate our algorithm with a simple example (Section 3.4) and describe some variants that are optimized for specific types of network topologies (Section 3.5).

3.1 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 connected components in the induced subgraph on $V - S$. Then, any path beginning in 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.[1] Our implementation of `BGPSep` uses the $O(n^3)$ *spectral partitioning algorithm* described in [19].

3.2 Basic algorithm

Let G denote the IGP subgraph induced by the egress routers and V denote the set of egress routers.[2] We now describe a simple iBGP configuration that satisfies P1, P2 and P3 without requiring a full-mesh iBGP. The next section 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, which ensures that all route reflectors see the routes from all other route reflectors [7].

Step 3 For every $u \in V - S$ and $v \in S$, make u a route reflector client of v.

Step 4 For each connected component G_i into which S separates G, set up iBGP sessions between every pair of routers in G_i (*i.e.*, construct a full-mesh configuration within each connected component, G_i).

[1] Algorithms for finding $(\sqrt{8n}, 1/6)$-separators are known for planar graphs [19]. However IGP graphs of ISPs are not guaranteed to be planar.

[2] We assume for now that all BGP routers in the AS are egress routers. In Section 3.5, we explain why the basic algorithm is inefficient for networks with internal routers and describe a variant that is more efficient for networks with non-egress BGP routers.

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 Chapter 4.

To show that complete visibility (P1) holds, it is enough to show that the route assignments at all routers are the same as they would have been in a full-mesh iBGP configuration. Suppose some router A would have picked the best route through egress router B to destination d in the full-mesh iBGP configuration, *i.e.,* B is the closest egress to A with a route to d amongst the set of egress routers with an equally good route to d. For complete visibility to hold, A should always learn of the route through B. The following claim completes the proof that complete visibility holds in the iBGP configuration described above.

Claim 3.2.1 *A learns of the route to d via B.*

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 every router in each connected component G_j is a client of every route reflector in S, there exists at least one route reflector R on the shortest path between A and B with both A and B as its clients. If B is the closest egress to A, then B will be the closest egress to R as well. 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. ■

Although this basic construction satisfies the correctness properties and has a smaller number of iBGP sessions compared to the full mesh iBGP (because 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 each G_i without violating P1,

P2 and P3 is just a smaller instance of the original problem we started to solve on G. Hence 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 (*i.e.,* have one or two nodes) or after a desired number of iterations. We now discuss the complete algorithm which applies the idea of the simplified construction recursively.

3.3 Complete algorithm

Algorithm 1 shows the recursive algorithm `BGPSep`. The algorithm takes the graph $G = (V, E)$ formed by the BGP routers as input and outputs the set I of iBGP sessions that must be established between the routers. Every iBGP session in the set I is represented as a triple 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 algorithm assumes the existence of a procedure `Graph-Separator`, a graph partitioning algorithm (*e.g.*, the algorithm described in [19]) that takes a graph G as input 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 = ∅;
else if |V| = 2 then
    {u, v} ← V ;
    I = {(u, v, peer)};
else
    /* Step 1: Choose a graph separator S ⊆ V. Routers in S are
       the route reflectors.                                    */
    S ← Graph-Separator(G) ;
    G_1, ..., G_m ← components of 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 router 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 BGPSep over each component */
        I_i = BGPSep(G_i) ;
        I = I ∪ 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. 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. Also note that the route reflectors produced in subsequent recursive iterations of the algorithm will be clients of the route reflectors produced in all previous iterations, thus forming a route reflector hierarchy.

3.4 An example

We now give a simple example to illustrate the `BGPSep` algorithm. Consider a network with ten BGP routers, as shown in Figure 3-1, all of which participate in eBGP. 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

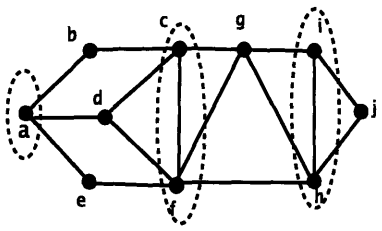

Figure 3-1: Illustration of BGPSep.

route reflectors, and 25 iBGP sessions. In contrast, a full-mesh iBGP configuration for this example has 45 iBGP sessions.

3.5 Variants

We now describe two variants of BGPSep optimized for different types of networks. Note that, in each case, the algorithm proposed in Section 3.3 would still be valid. The variants are either further optimizations to reduce the number of iBGP sessions or for convenience.

3.5.1 Networks with internal BGP routers

If the network contains internal BGP routers that do not receive any external routes, those routers need not have iBGP sessions with each other. So the BGPSep algorithm, which aims to emulate a full-mesh over all BGP routers, may establish some unnecessary iBGP sessions when run over the entire topology of internal and egress routers. To avoid these extra sessions, we propose a variant, BGPSep-Internal that works better for ASes with a large proportion of non-egress BGP routers. In Step 1 of BGPSep-Internal, the algorithm finds a set S of routers to separate the graph into a set of connected components: with one component G_{ext} containing all the egress routers (and possibly some internal BGP routers as well). Steps 2 and 3 remain the same. In step 4, the algorithm only recurses on G_{ext} because the internal routers

need not have iBGP sessions with each other. Algorithm 2 specifies the complete algorithm.

```
BGPSep-Internal;
Input: IGP Graph G = (V, E) of all BGP routers
Output: Set I of iBGP sessions
I = ∅ ;
S ← Graph-separator(G) ;
/* G_int and G_ext and the components containing the internal and
   egress routers respectively                                    */
G_int, G_ext ← subgraph on V − S;
/* Step 2                                                         */
foreach u, v ∈ S, u ≠ v do
 |  I = I ∪ {(u, v, peer)};
end
/* Step 3                                                         */
foreach u ∈ G_int, v ∈ S do
 |  I = I ∪ {(u, v, client)};
end
/* Step 4:  Recurse over the component with egress routers        */
foreach u ∈ G_ext, v ∈ S do
 |  I = I ∪ {(u, v, client)};
end
I_ext = BGPSep(G_ext) ;
I = I ∪ I_ext ;
return I;
```

Algorithm 2: A variant of BGPSep algorithm optimized for networks with large number of internal BGP routers

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.

3.5.2 Backbone-like ISP networks

The IGP topologies of a large ISPs consist of a set of points-of-presence (PoPs) spread across the ISP's area of coverage [20]. 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 ISP's network.[3] A route reflector configuration is formed over the IGP topology by configuring the backbone routers in a PoP as route reflectors and all the access routers in the PoP as clients of those route reflectors. The route-reflectors at the top of the hierarchy need to be configured in a full-mesh to ensure that all route reflectors learn of all routes coming into the AS. In practice, however, a route reflector hierarchy is used instead of a full-mesh if there are a large number of backbone routers.

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

To solve this problem, we propose a variant `BGPSep-Backbone` (shown in Algorithm 3) that is better suited for ISP-like backbone networks. The idea is simple: run `BGPSep` on the backbone routers alone, to construct a route reflector hierarchy that emulates the case of a fully-meshed backbone. Then configure the backbone routers in each PoP as route reflectors of the access routers in the PoP, as is done in practice today. Lastly, fully mesh the access routers in each PoP. The last step is needed because configuring a route reflector hierarchy according to `BGPSep` only ensures that every shortest path between two access routers in different PoPs passes through a graph separator and the same property does not hold for shortest paths between access routers in the same PoP.

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

`BGPSep-Backbone`;

Input: IGP $G = (V, E)$ of all BGP routers of an ISP

Output: Set I of iBGP sessions

$I = \emptyset$;

`/* Run the BGPSep algorithm over the backbone routers */`

$G_b \leftarrow$ subgraph of backbone routers in G;

$I_b =$ `BGPSep`(G_b);

$I = I \cup I_b$;

$P_1, \ldots, P_p \leftarrow$ PoPs of the ISP;

foreach P_i **do**

  $A_{i1} \ldots A_{ia} \leftarrow$ access routers in P_i;

  $B_{i1} \ldots B_{ib} \leftarrow$ backbone routers in P_i;

end

`/* Configure the access routers in each PoP as route reflector clients of each backbone router of the PoP */`

foreach $A_{ij}, B_{ik} \in P_i$ **do**

  $I = I \cup \{(A_{ij}, B_{ik}, \text{client})\}$;

end

`/* Fully-mesh the access routers in each PoP */`

foreach $A_{ij}, A_{ik} \in P_i$ **do**

  $I = I \cup \{(A_{ij}, A_{ik}, \text{peer})\}$;

end

return I;

Algorithm 3: A variant of the `BGPSep` algorithm optimized for backbone-like ISP networks

Chapter 4

Proof of Correctness

In this chapter, we rigorously prove that the iBGP configurations output by `BGPSep` satisfy the properties of complete visibility, loop-free forwarding and robustness to IGP changes (referred to as P1, P2, and P3 respectively in Chapter 1).

Let V denote the set of all BGP routers in the network[1] and let G be the IGP subgraph induced by the routers in V. Let d denote any destination. Let $E_d \subseteq V$ denote the set of egresses that have routes to d which are equally good up to the step of comparing the IGP cost to the egress. For every router $A \in V$, let $\xi_d^A \in E_d$ denote the egress router from amongst the routers in E_d that has the shortest IGP path cost to A.

The following lemma proves a fundamental property of shortest path routing.

Lemma 4.0.1 *If A and ξ_d^A are not adjacent to each other in G, then for every router C on the shortest path from A to ξ_d^A, $\xi_d^C = \xi_d^A$.*

Proof We prove the lemma by contradiction. Let $B = \xi_d^A$. Suppose that for some router C on the shortest path between A and B, $\xi_d^C = B'$ and $B' \neq B$. Then B' is closer to C than B (IGP cost-wise), which means that B' is also closer to A than B contradicting the fact that $B = \xi_d^A$. Hence $\xi_d^C = \xi_d^A$. ■

[1]Our proofs assume that all BGP routers in the network are egress routers *i.e.,* there are no internal BGP routers. The proofs in the case of networks with internal BGP routers are similar

4.1 Complete visibility

Definition 4.1.1 *An iBGP configuration satisfies complete visibility if every router $A \in V$ chooses the route via egress ξ_d^A as its best route to destination d.*

Observe that if A learns of the route through ξ_d^A, it always chooses that route as its best route to d. In order to show that complete visibility holds, it is enough to show that every router A learns of the route via egress ξ_d^A.

We now prove that the iBGP configuration produced by **BGPSep** satisfies complete visibility. We begin by defining a *signaling chain.*

Definition 4.1.2 *A signaling chain between two routers $A, B \in V$ is defined as a set of routers $A(= R_0), R_1, R_2, \ldots, R_r, B(= R_{r+1}) \in V, r \geq 1$, such that for $i = 1 \ldots r$, (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 4.1.1 *In the iBGP configuration produced by **BGPSep**, for every $A \in V$, there either exists an iBGP session or a signaling chain on the shortest path between A and ξ_d^A.*

Proof Let $B = \xi_d^A$. If A and B are in the same component when the recursion terminates in **BGPSep** then A and B have an iBGP session between them. Otherwise, consider the shortest path between A and B in G. From the construction in **BGPSep**, we know that this shortest path passes through a set of recursively produced graph separators. Because the route reflectors in a graph separator produced in one recursive iteration of the algorithm are clients of the route reflectors produced in all previous iterations, it follows that there exist route reflectors $R_1, \ldots R_r \in V$ $(r \geq 1)$ on the shortest path (in that order) such that at least one of R_{i+1} or R_{i-1} is a route reflector client of R_i. Hence $A(= R_0), R_1, R_2, \ldots, R_r, B = (R_{r+1})$ is a signaling chain. Note that $R_1, \ldots R_r$ need not be adjacent to each other on the shortest path. ■

Lemma 4.1.2 *If there exists a signaling chain on the shortest path between routers A and ξ_d^A, then A learns of the best route via ξ_d^A to destination d.*

Proof Let $B = \xi_d^A$ and let $A(= R_0), R_1, R_2, \ldots, R_r, B(= R_{r+1}) \in V, r \geq 1$, be the signaling chain on the shortest path between A and B. We first claim that R_i propagates the best route to d learned from R_{i+1} to R_{i-1} for $i = 1 \ldots r$.

To see why, recall that in the signaling chain, at least one of R_{i+1} or R_{i-1} is a route reflector client of R_i. If R_{i+1} is a route reflector client of R_i, then R_i propagates the best route 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 a best route learned on any of its iBGP sessions to all its clients.

Now, by Lemma 4.1.1, $\xi_d^{R_i} = B$ for $i = 1 \ldots r$ *i.e.,* every router on the signaling chain between A and B also chooses B as its egress router. Hence the route to d via $R_{r+1} = B$ propagates "from left to right" along the signaling "chain" to $R_r, R_{r-1} \ldots$ and eventually reach $R_0 = A$. ■

The theorem below follows from Lemmas 4.1.1 and 4.1.2.

Theorem 4.1.1 *The iBGP configuration output by* ***BGPSep*** *satisfies the property of complete visibility.*

Proof Consider any router $A \in V$ and let $B = \xi_d^A$. By Lemma 4.1.1, there either exists an iBGP session or a signaling chain on the shortest path between A and B. If there exists an iBGP session between A and B, then A learns of the best route to d via B. On the other hand, if there exists a signaling chain, then by Lemma 4.1.2, A learns of the best route to d via B. Hence A always chooses the best route to d via B. ■

4.2 Loop-free forwarding

Theorem 4.2.1 *An iBGP configuration that satisfies complete visibility also satisfies the property of loop-free forwarding if the IGP implements shortest path routing.*

Proof Consider the forwarding path within the AS of a packet to d from some router $A \in V$. By Theorem 4.1.1, A chooses $B = \xi_d^A$ as its egress router to d. If the IGP implements shortest path routing, the forwarding path within an AS is simply the shortest path from A to B. By Lemma 4.0.1, for every router C on the shortest path from A to B, the route through B is the best route to d (*i.e.*, $\xi_d^C = \xi_d^A = B$) and by Theorem 4.1.1 again, C learns of the route to d via ξ_d^C. Thus every router on the shortest path between A and B consistently forwards the packet destined to d towards B. Therefore, there are no deflections when packets are forwarded along the shortest path from A to B, guaranteeing loop-free forwarding. ■

Because we know that iBGP configurations output by `BGPSep` satisfy complete visibility, Theorem 4.2.1 implies that these iBGP configurations also satisfy the property of loop-free forwarding (assuming that the IGP implements shortest path routing, which is true of most IGPs). Note that it may also be possible to construct iBGP configurations which satisfy loop-free forwarding alone without satisfying the stronger condition of complete visibility; we have not addressed this question in this paper.

4.3 Robustness to IGP changes

Lemma 4.3.1 *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. Note that a graph separator of a graph depends only on the connectivity of nodes in a graph and not on the edge weights between the nodes. ■

Lemma 4.3.2 *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'. In simple terms, a graph separator of a graph remains a separator on any subgraph of the original graph. This property ensures that properties P1 and P2 hold even in the face of IGP router and link failures in the iBGP configurations produced by **BGPSep**. No reconfiguration is required to cope with these failures. ■

Note that the proofs of this section assume that the IGP has converged to a stable topology following a link cost change or failure.

4.4 Caveats and limitations

Not robust to iBGP failures: Though **BGPSep** is robust to IGP failures, there is another class of failures which break the correctness properties of our algorithm: iBGP failures, where only the iBGP configuration changes without changing the underlying IGP topology (*i.e.*,, 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). When such failures occur, we can no longer assume that the nodes in the graph separators are all route reflectors, and so our correctness guarantees break down. Note that in the case of iBGP failures, the correctness properties of the full-mesh iBGP also do not hold.

Not incremental: **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. However, at no other time is the re-running of the algorithm required for the correctness properties to hold. In particular, as explained earlier, the algorithm does not need to be re-run on failures or removals of links and routers.

Chapter 5

Evaluation

In this chapter, we describe the implementation of BGPSep and the evaluation of our implementation on various real-world and synthetic network topologies[1]. We first explain our implementation in Section 5.1. We then describe the network topologies used in our evaluation in (Section 5.2). We investigate how the number of iBGP sessions in the configurations produced by BGPSep compares with that in the full-mesh iBGP[2] (Section 5.3) and how this number of iBGP sessions scales with the number of BGP routers in the network (Section 5.4).

5.1 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. Our algorithm cannot be used in a distributed setting—we assume that a network operator will run the algorithm with the complete IGP graph as input, and configure the iBGP according to the iBGP sessions output by the algorithm.

We implemented the $O(n^3)$ spectral partitioning algorithm from [19] to find graph

[1] A network topology is the IGP graph of the network, which is the input to the BGPSep algorithm.

[2] 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.

separators. Our implementation is efficient—`BGPSep` runs in under 5 seconds on an Intel Xeon 3GHz processor for real network topologies having between 80 and 300 nodes.

Because our implementation is efficient and iBGP configurations produced by `BGPSep` need not be regenerated on link cost changes or node failures, we believe that `BGPSep` is a practical alternative for iBGP configuration today.

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.

5.2 Network topologies

We use both real-world and synthetic topologies in our evaluation of `BGPSep`. For real-world topologies, we use the backbone topologies of 6 ISPs annotated with inferred link costs obtained from the Rocketfuel project [14]. The ISP backbone topologies are summarized in Table 5.1. We also evaluate `BGPSep` on synthetic topologies generated using GT-ITM [4], with the GT-ITM parameters set according to the suggestions in [12].

5.3 `BGPSep` vs. full-mesh iBGP

We run the `BGPSep` algorithm on the network topologies described in Section 5.2 to produce iBGP configurations. We compare the number of iBGP sessions in these configurations to the number of iBGP sessions that would have been required had the networks used a full-mesh iBGP. We assume (conservatively) that all the nodes in

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 5.1: ISP topologies used for the evaluation of **BGPSep**.

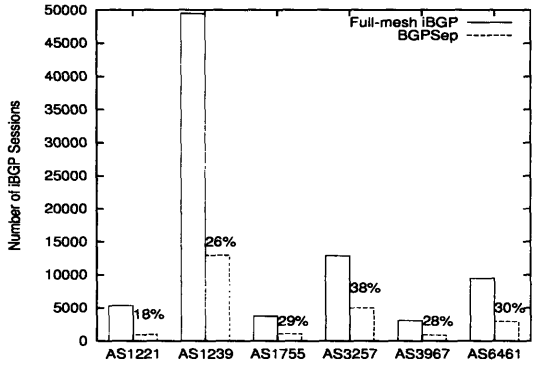

Figure 5-1: **BGPSep** vs. full-mesh iBGP: real-world network topologies.

the topology are egress routers. The results of the comparison are shown in Figures 5-1 (real-world topologies) and 5-2 (synthetic topologies). We observe that the iBGP configuration produced by **BGPSep** results in a 2.5× to 5× reduction in the number of iBGP sessions on real-world topologies, and a 5× to 10× reduction on synthetic topologies compared to the full-mesh iBGP.

We also observe from Figures 5-1 and 5-2 that the reduction in the number of iBGP sessions is more significant in the case of synthetic network topologies. The reason for the better performance of **BGPSep** on synthetic topologies could be that synthetic topologies are produced using well-defined structured rules (a certain num-

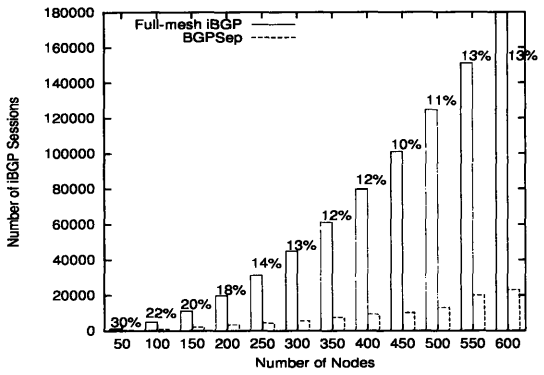

Figure 5-2: `BGPSep` vs. full-mesh iBGP: synthetic network topologies.

ber of central cores, a certain number of stubs hanging off each core, etc.) and have smaller-sized separators. Real-world topologies do not seem to have as much structure and so have bigger separators.

Another key aspect of the iBGP configurations produced by `BGPSep` is the number of route reflectors, the number of top-level route reflectors (*i.e.*, the number of route reflectors at the highest level of the hierarchy), and the number of levels in the resulting route reflector hierarchy. These numbers measured for the iBGP configurations of the Rocketfuel ISP topologies are listed in Table 5.2. The number of top-level route reflectors is an important metric because it is the top-level route reflectors that usually have the most clients and hence the most complex configurations. These results show that although a substantial number of nodes are route reflectors, the number of top-level route reflectors is relatively small.

5.4 Scaling

We know that the number of iBGP sessions in the full-mesh iBGP scales quadratically as the number of nodes—*i.e.*, if n is the number of eBGP routers and N_{ibgp} the number of iBGP sessions in a network, then N_{ibgp} scales as n^2 in the full-mesh iBGP configuration. We now measure empirically how N_{ibgp} varies with n in the iBGP

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 5.2: Number of route reflectors, top-level route reflectors, and number of levels in the hierarchy of route reflectors in the iBGP configurations generated by **BGPSep**.

configurations output by **BGPSep**.

To conduct this evaluation, we need network topologies with varying number of BGP routers n. While it is easy to generate synthetic topologies with varying number of routers using GT-ITM, it is tougher to find real-world ISP topologies with different values of n. We therefore turn to constructing subgraphs of the Rocketfuel ISP topologies to emulate real-world ISP topologies with varying number of BGP routers.

We generate subgraphs of five different sizes from each Rocketfuel ISP topology as follows: if n_{orig} is the number of BGP routers in the original network topology, we generate a random subgraph of n_i nodes, where $n_i = \frac{n_{orig}}{2^i}, i = 0 \ldots 4$. For each i (except when $i = 0$), we generate 10 subgraphs, choosing a different random subset of n_i nodes each time. We also generate synthetic GT-ITM topologies with the number of BGP routers n varying from 50 to 600 in steps of 50, generating 10 topologies for each value of n. We then run **BGPSep** on each of the above network topologies and compute the number of iBGP sessions in the resulting iBGP configurations. We plot the mean and standard deviation of number of iBGP sessions for each value of the number of BGP routers, as shown in Figure 5-3. In the case of the real-world topologies, we show the results only for one ISP (AS1239)—the results in the case of

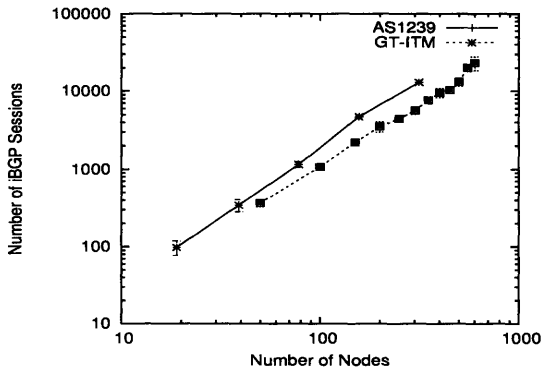

Figure 5-3: Number of iBGP sessions vs number of BGP routers: AS1239 and GT-ITM. The curves show the mean value and the error bars show one standard deviation.

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

Table 5.3: Measured values of k where the number of iBGP sessions scales as n^k.

the other ISPs are similar.

If we plot the number of iBGP sessions N_{ibgp} in the iBGP configurations output by **BGPSep** and the number of BGP routers in a network n on a log-log scale, the slope of the best-fit line of the plot gives the scaling behavior N_{ibgp} with n; 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 5.3. 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.

In addition to the number of iBGP sessions, we also compute the number of route reflectors and the number of top-level route reflectors for each **BGPSep** iBGP configuration generated above. The mean and the standard deviation of the number of route reflectors and the number of top-level route reflectors for different values of

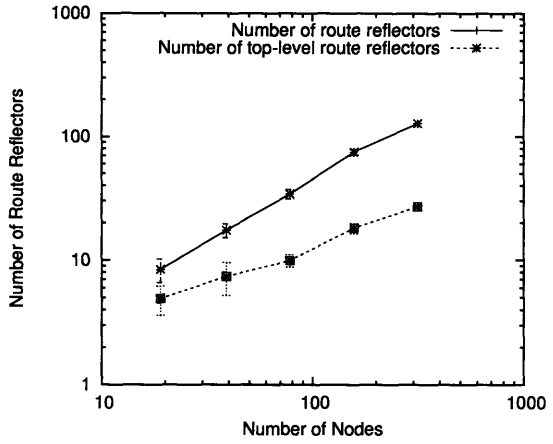

Figure 5-4: Number of route reflectors vs. number of BGP routers: AS1239. The curves show the mean value and the error bars show one standard deviation.

the number of BGP routers n is shown in Figure 5-4. Again, the results are shown for one representative ISP (AS1239). The slopes of the best-fit lines in this log-log plot are 0.95 and 0.53, indicating that the number of route reflectors and the number of top-level route reflectors scale in proportion to $n^{0.95}$ and $n^{0.53}$ respectively for the network topology under consideration.

Chapter 6

Conclusion

Perhaps the most complex 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 forwarding loops and sub-optimal paths. These problems are hard to diagnose and debug, and networks with these problems are hard to manage. The two common approaches to disseminating external routes within a network—full-mesh iBGP and iBGP with route reflection—are both flawed. While the full-mesh is not scalable, route reflection does not provide any correctness guarantees.

We proposed the `BGPSep` algorithm to construct an iBGP configuration that satisfies the properties of complete visibility, loop-free forwarding, and robustness to failures. An evaluation of `BGPSep` 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× and 5× 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. The algorithm admits an efficient and practical implementation, and can easily

be integrated into tools that produce router configuration code. In addition, deploying `BGPSep` is easy because it does not require any changes to existing route reflectors. We believe that `BGPSep` can eliminate some hard-to-diagnose network problems that network operators face.

Bibliography

[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, 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.

[10] 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.

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

[12] 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.

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

[14] 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.

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

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

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

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

[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. on Networking*, 12(1):2–16, 2004.

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