Network-Wide Prediction of BGP Routes

Nick Feamster
Georgia Tech
feamster@cc.gatech.edu

Jennifer Rexford
Princeton University
jrex@cs.princeton.edu

***Abstract*— This paper presents provably correct algorithms for computing the outcome of the BGP route-selection process for each router in a network, without simulating the complex details of BGP message passing. The algorithms require only static inputs that can be easily obtained from the routers: the BGP routes learned from neighboring domains, the import policies configured on the BGP sessions, and the internal topology. Solving the problem would be easy if the route-selection process were deterministic and every router received all candidate BGP routes. However, two important features of BGP—the Multiple Exit Discriminator (MED) attribute and route reflectors—violate these properties. After presenting a simple route-prediction algorithm for networks that do not use these features, we present algorithms that capture the effects of the MED attribute and route reflectors in isolation. Then, we explain why the interaction between these two features precludes efficient route prediction. These two features also create difficulties for the operation of BGP itself, leading us to suggest improvements to BGP that achieve the same goals as MED and route reflection without introducing the negative side effects.**

I. Introduction

To control the flow of traffic through their networks, operators need to know how configuration changes affect the routes that each router in the network selects. The outcome of the route-selection process depends on the routes advertised by neighboring domains, the internal topology, the interdomain routing policies, and the intradomain link weights. Ordinarily, computing the outcome would require a complex simulation of routing-protocol dynamics. Instead, we present efficient algorithms that compute the outcome of the BGP route-selection process without backtracking. In designing our algorithms, we grapple with two features of the Border Gateway Protocol (BGP) [1]: limited visibility into the available routes for each destination and non-deterministic ranking of these routes.

A. Backbone Network Engineering

The flow of traffic through a backbone network depends on the interactions between three routing protocols, as shown in Figure 1:

External BGP (eBGP): Routers in the AS use eBGP to receive route advertisements from neighboring ASes. For example, the routers W, X, and Y each have eBGP sessions with neighboring ASes. The routers may apply an import policy to modify the attributes of the routes learned from the neighbor, with the goal of influencing the selection process in Table I that each router applies to select a single best BGP route for each destination prefix.

Internal BGP (iBGP): The routers use iBGP to disseminate the routes to the rest of the network. In the simplest case, each router has an iBGP session with every eBGP-speaking router, forming an "full mesh" configuration. If two routes are equally good through the first four steps in Table I, the router favors an eBGP-learned route over an iBGP-learned one. In Figure 1, router Z receives three iBGP routes, from routers W, X, and Y. Upon learning routes with the same local preference, AS path length, origin type, and MED values, router Z uses the IGP to break ties between the remaining routes.

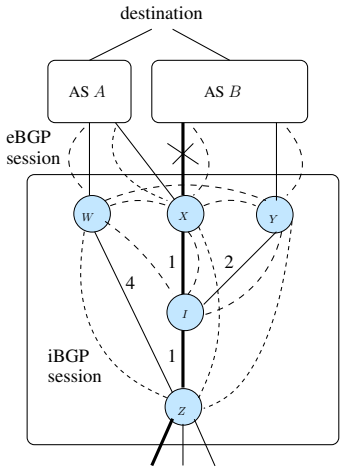


Fig. 1. Network with three egress routers connecting to two neighboring ASes: Solid lines correspond to physical links (internal links are annotated with IGP link weights) and dashed lines correspond to BGP sessions. Thick lines illustrate the shortest path from one router to its closest egress point for reaching the destination.

Interior Gateway Protocol (IGP): The routers run an Interior Gateway Protocol (IGP) to learn how to reach each other. Two common IGPs today are OSPF [2] and IS-IS [3], which compute shortest paths based on configurable link weights. The routers also use the IGP path costs in the sixth step of the BGP route-selection process in Table I. In Figure 1, router Z selects the route with the smallest IGP path cost of 2, learned from router X.[1]

After selecting a route to each destination, each router combines the BGP and IGP information to construct a forwarding table that maps the destination prefix to the outgoing link along the shortest path. In Figure 1, the forwarding path consists of the thick lines from the ingress link at router Z to the egress link at router X.

If the link from X to AS B becomes persistently congested, the network operator may need to adjust the configuration of

[1]If two routes have the same IGP path cost, the router performs an arbitrary tiebreak in the seventh step in Table I.

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

TABLE I

STEPS IN THE BGP ROUTE-SELECTION PROCESS.

the routing protocols to direct some of the traffic to other egress routers. For example, the operator could modify the import policy at router X for the routes it learns from AS A and AS B to make the BGP routes for some destinations look less attractive than the routes received at other routers [4]. Changing the import policy in this way causes the route that X readvertises via iBGP to carry a smaller *local preference*, which influences the routes that other routers in the network select. For example, changing the import policy at X has the *indirect* effect of directing some of the traffic entering at router Z to egress router Y (the next-closest egress point, in terms of the IGP path costs), thereby alleviating the congestion on the link connecting X to AS B. Network operators make similar kinds of configuration changes for a variety of other reasons, such as exploiting new link capacity, preparing for maintenance on part of the network, or reacting to equipment failures.

Operators must predict the effects of changes to the routing protocol configuration before modifying the configuration on a live network. Human intuition is not sufficient for understanding the complex interactions between three routing protocols running on a large, dynamic network. Experimenting on a live network runs the risk of making disruptive configuration changes that degrade performance. Instead, we believe that operators should have an accurate and efficient tool that computes the effects of configuration changes on the flow of traffic through the network. This tool should allow a network operator (or automated configuration algorithm) to efficiently explore the large space of possible configurations.

B. Problem Statement and Challenges

Our goal is to compute the *outcome*—the routing decision for each router—once the protocols have converged. Accordingly, we present algorithms that accurately and quickly determine how the network configuration and the eBGP-learned routes affect the flow of traffic through an AS. Some existing tools simulate BGP's behavior [5–7] and even use simulation as an "inner loop" for optimizing BGP policy configuration [8]. On the other hand, this work is the first to develop algorithms that determine the outcome of the BGP route-selection process at each router in an AS without simulating the dynamics of the protocol.

Predicting the route that each router ultimately selects is challenging because the route selected by one router often depends on the routes selected by other routers. Consider Figure 2, where router R_1 receives two routes via eBGP, while

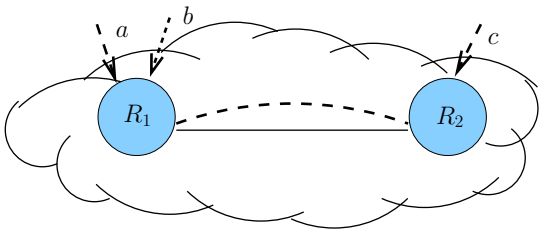


Fig. 2. Route prediction requires resolving circular dependencies. Determining the route that R_2 ultimately selects (*i.e.*, a, b, or c) first requires determining whether R_1 selects route a or b. Ultimately, R_1's selected route could depend on whether it learns route c from R_2, which requires revisiting R_1.

R_2 receives a single route via eBGP. To determine the route that each one of these routers ultimately selects, we must first determine the candidate routes available to each router. Of course, the set of candidate routes available to each of these routers depends on the route that the other selects! This circular dependency seems to imply some "back and forth" reasoning (*i.e.*, determining the route that R_1 selects depends on the route that R_2 selects, which in turn depends on the route that R_1 selects, etc.). Efficiently resolving these types of circular dependencies is the focus of this paper. In particular, we solve the following problem:

> **Problem:** Given only a static snapshot of the routing configuration for the routers in an AS and the BGP routes learned from neighbor ASes, determine the route that each router selects for each destination, while considering each available candidate route only once.

Solving this problem would be easy if (i) the route-selection process in Table I allowed each router to form a deterministic ranking of all candidate routes and (ii) the dissemination of routes in iBGP ensured each router received the best route for a destination from every eBGP-speaking router. If these two properties held, then a simple algorithm that considered which route each router would select from all of the eBGP-learned routes would correctly compute the outcome of BGP route selection without having to revisit any routers. Unfortunately, two features of BGP cause these properties to be violated, thus making route prediction more challenging:

The BGP selection process does not form a deterministic ranking of the routes, due to the Multiple Exit Discriminator (MED) attribute. An eBGP neighbor can set the MED attribute of route advertisements on different BGP sessions to influence the decisions in a neighboring AS. For example, in Figure 1, AS B may send a route with a MED of 10 to router Y and a route with a MED of 20 to router X; as a result, Z would select the route from Y with the smaller MED, even though the IGP path to X is shorter. The MED comparison in Table I applies only to routes learned from the *same* next-hop AS. When MEDs are used in this fashion, a router does not necessarily have a deterministic ranking of the BGP routes. In other words, the choice of one route over another may depend

on the presence or absence of a third route [9]. An accurate route-prediction algorithm must resolve these dependencies.

The dissemination of routes within an AS does not necessarily satisfy visibility, due to the use of route reflectors. The quadratic scaling of a full-mesh iBGP configuration forces large networks to distribute routes in a hierarchical fashion. A router configured as a route reflector selects a single best route and forwards the route to its clients. Using route reflectors reduces the number of iBGP sessions, as well as the number of routes the clients need to receive and store. Because each route reflector forwards only its *best* route to its iBGP neighbors, the candidate routes available at one router depend on decisions at other routers. In particular, a route reflector may make a different choice in step 6 of the route-selection process (*i.e.*, shortest IGP path to the next-hop IP address) than its clients because it is located at a different place in the IGP topology.

In general, these two features of BGP cannot be ignored because operators use them often to satisfy important policy and scalability goals. To illustrate the extent to which these artifacts of BGP complicate route prediction, we present the "ideal" route-prediction algorithm, before considering more sophisticated algorithms that capture the effects of these two artifacts. The paper makes three main contributions:

Simple algorithm for predicting BGP route selection, when determinism and visibility are satisfied: Rather than analyzing BGP dynamics, we present efficient algorithms to compute the outcome of the distributed route-selection process using only static inputs. Our algorithms exploit the following observation: when a routing system converges, the outcome does not depend on the order and timing of the messages, allowing our algorithms to apply a message ordering that efficiently computes the outcome of BGP route selection. Section II presents practical constraints that enable efficient computation of network-wide BGP route selection and decomposes the route-prediction problem into three stages. After we introduce some notation in Section III, Section IV presents an algorithm that computes the outcome of BGP route selection for the simple case, with a full-mesh iBGP configuration and no MED attribute.[2]

Algorithms that capture the influence of the MED attribute and route reflectors: Most of the rest of the paper deals with route prediction in networks that employ MED, route reflection, or both. We first present algorithms that handle MED and route reflectors in isolation. We then discuss why the interaction between these two features precludes efficient route prediction. Section V focuses on algorithms that capture the effects of the MED attribute, assuming a full-mesh iBGP configuration. In Section VI, we consider iBGP configurations that use route reflection. In a longer version of this paper, we suggest ways to improve the design and operation of BGP to avoid the harmful effects without sacrificing the policy semantics of MEDs and the scalability provided by route reflectors [10].

[2]Throughout the paper, we often describe BGP "without MED". Network configurations "without MED" could also be viewed as a configuration that compares the MED attributes across all routes (*e.g.*, in Cisco IOS, this behavior can be enabled with `always-compare-med` setting).

Our route-prediction algorithms have been implemented in a traffic-engineering tool for network operators [11, 12]. We tested our prototype on a large tier-1 ISP to measure the speed of the algorithms on realistic inputs and to validate the correctness of the results. The study showed that the tool provides fast, accurate answers to "what if" questions about the effects of configuration changes on the flow of traffic through the network.

II. Modeling Constraints and Overview

In this section, we impose three constraints that the routing system must satisfy to enable efficient and accurate route prediction. Next, we describe how these constraints enable us to decompose the algorithm into three stages—applying the import policy to eBGP-learned routes, selecting the best BGP route at each router, and computing the forwarding path. The algorithm takes as input the router configuration and a static snapshot of the routes learned via eBGP and outputs the route that each router in the AS selects, for each destination. Because the first and third stages of the algorithm are relatively simple, the rest of the paper focuses on the second stage of computing the best BGP route at each router for each destination prefix.

A. Modeling Constraints

Efficiently computing the effects of a configuration or topology change is possible when three important conditions hold. Imposing the constraints we outline in this section frees our prediction algorithms from needing to consider whether different orderings of routing messages will produce different results. This property allows us to focus on designing algorithms that emulate a particular message ordering that prevents the algorithm from having to revisit routers where it has already made a prediction. The rest of this section explains how these constraints help simplify the prediction algorithms.

First, the inputs to the algorithm must be stable.

Constraint 1 (Slowly changing inputs): The eBGP-learned routes change slowly with respect to the timescale of network engineering decisions.

If the eBGP-learned routes change frequently, the internal routing system does not have time to propagate the effects of one eBGP advertisement before the next one arrives. In practice, most BGP routes are stable for days or weeks at a time [13], and the vast majority of traffic is associated with these stable routes [14]. This allows the routing algorithm to operate on a static snapshot of the eBGP routes. Any eBGP routing change can be treated as a separate problem instance.

Second, the routers must ultimately converge to stable outcome.

Constraint 2 (Safety and uniqueness): Given stable eBGP-learned routes and fixed iBGP and IGP topologies, each router inside the AS converges to a unique routing decision.

If the routers continually change the routes that they select, accurately predicting the flow of data traffic becomes significantly more challenging. Fortunately, previous work [15] has identified sufficient conditions for an internal routing configuration to satisfy Constraint 2. We describe these conditions

in more detail in Section VI when we address the challenges introduced by route reflectors.

Third, the routing decisions at each router should not depend on message ordering or timing.

Constraint 3 (Independence of message ordering): The routing decision at each router depends only on the routes received from its neighbors and *not* the order or timing of these routing messages.[3]

Common BGP implementations have two configurable features that, if enabled, would violate Constraint 3. First, some router vendors have an additional step in the BGP route-selection process that favors the "oldest" route before the final tie-breaking step of comparing the router IDs. Configuring the router to skip this step in the route-selection process avoids the problem. Second, the MED attribute can also cause violations of Constraint 3 in two cases: (1) if the router compares a new route announcement to only the current best route, rather than rerunning the entire route-selection process; or (2) in certain pathological cases, such as the "mashed potato" configuration described in previous work [17]. To handle the first case, router vendors recommend enabling the "bgp deterministic-med" option to ensure that route selection does not depend on which routes were learned first; we discuss this feature in greater detail in the Appendix of our previous study of interdomain traffic engineering [4]. The second case results from the fact that MED prevents the set of routes to any destination from forming a total ordering.

Constraint 2, which guarantees that the routing system will converge to a unique outcome, and Constraint 3, which guarantees that this outcome does not depend on the ordering of routing messages, allow us to make the following observation:

> *Observation 1:* If a routing system is guaranteed to converge to a unique outcome, that outcome is independent of the order in which routers exchange routes and apply the route-selection process.

This observation implies that the algorithm can consider the evolution of the routing system under *any particular message ordering*, without the risk of arriving at the wrong answer.

Although our algorithms require Constraints 2 and 3 to hold, we note that existing static analysis tools (*e.g.*, *rcc* [18]) can easily check that the routing configuration satisfies these constraints. It is also worth noting that, although our algorithms assume that the routing protocol configuration satisfies the above properties, they do not assume that the routing protocol is configured "correctly", as defined in previous work [16] (*e.g.*, the resulting BGP routes may give rise to forwarding loops). The algorithms in this paper are only concerned with predicting the outcome of BGP selection process, not whether the resulting routes actually produce the desired results.

[3]In previous work, we incorporated this concept into the definition of determinism [11, 16]; in this paper, we have separated the terms for clarity.

B. Problem Decomposition

Following the approach applied in other recent work [19, 20], the algorithms in this paper compute the effects of a particular message ordering using an *activation sequence*, an *offline* analysis technique that "activates" one or more routers at each discrete step. When activated, a router applies the route-selection process in Table I and propagates the best route to its iBGP neighbors. In an actual network, routers may be activated in any order and may change their best route many times before the network converges. Capitalizing on Observation 1, our algorithms are based on an activation sequence that allows us to decompose route prediction into three distinct stages, as shown in Figure 3:

1. Receiving the eBGP routes and applying import policy. This stage takes as input all of the eBGP-learned routes at each router and applies the appropriate import policies at each router *before exchanging any iBGP update messages* and outputs the set of eBGP-learned routes after these import policies have been applied. This stage activates all of the edge routers at the same time. Each eBGP-learned route has attributes (such as the destination prefix and the AS path) and is associated with an eBGP session. The import policy may filter the route or set certain attributes such as local preference, origin type, and multiple-exit discriminator (MED), according to attributes in the advertised route (*e.g.*, based on ASes in the AS path). Because applying the import policy is a local operation for each eBGP-learned route at each router, the first stage emulates the operations a real router would perform upon receiving each of the eBGP routes. These routes, with modified attributes, are the input to the second stage.

2. Computing the best BGP route at each router. Many routes from the first stage could never be selected by any router as the best route. For example, an eBGP-learned route with a local preference of 90 would *never* be selected over another route with a local preference of 100. In addition, different routers in the AS may select different best BGP routes because they have different IGP path costs to the egress router. Also, a router can only consider the routes advertised by its iBGP and eBGP neighbors, which may influence the final decision at that router. This stage takes as input the set of eBGP-learned routes after the import policies of each router have been applied and outputs a single best egress router for each ingress router and destination prefix. Constructing an efficient activation sequence for this stage is challenging and is the focus of the next four sections of the paper.

3. Computing the forwarding path through the AS: The third stage computes the effects of the IGP link weights on the construction of the forwarding path through the AS from an ingress router toward a destination prefix. Given the selected BGP route, the ingress router forwards packets along the outgoing link (or links) along shortest paths to the egress router, and the process repeats at the next router. Ideally, the traffic flows along the shortest path (or paths) all the way from the ingress router to the selected egress router. However, in practice, routers along the shortest path may have selected a *different* egress router. These violations can occur if the iBGP session configuration limits the BGP routing options at the

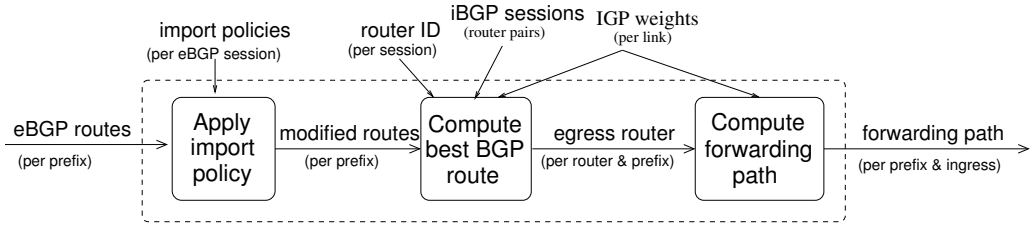


Fig. 3. Our algorithms decompose network-wide BGP route selection into three independent stages. The algorithms take as input the eBGP-learned routes from neighboring ASes, the router IDs of each BGP session, and the routing configurations from all of the routers in the AS, which provide information about the IGP topology, the iBGP topology, and the import policies (*i.e.*, rankings) of each router.

routers [15]. By considering one router at a time, the third stage can compute an accurate view of the forwarding path(s) even when deflections occur.

While all three steps are necessary for determining the flow of traffic through a network from a static snapshot of the network state, the rest of this paper focuses on the second step (*i.e.*, computing the best BGP route at each router), since performing this step correctly and efficiently is considerably more difficult than either of the other two steps.

III. Preliminaries

We first introduce some notation. Table II lists the notation we use for the remainder of this paper and summarizes where this notation is introduced. We assume that the AS has a set E of N eBGP-learned routes for a given destination prefix, which it learns at R routers. E contains the eBGP-learned routes after each router in the AS has applied its local import policy (which may filter some set of the routes it receives and set or modify the route attributes of others). For convenience, we define $E_r \subseteq E$ as the set of eBGP-learned routes at router $r \in R$. At any given time, a router r also has zero or more iBGP-learned routes $I_r \subseteq E$. We define two useful functions:

- λ_r, which takes a set of routes at router r and produces the best route at router r according to the BGP route-selection process in Table I.

The subscript on λ_r is necessary because different routers can apply the BGP route-selection process to the same set of routes and obtain different results based on the BGP session from which they learn the route and their location in the topology. For example, in Figure 1, router X would treat the route learned from AS B as an eBGP-learned route with the router ID of the eBGP session with B. On the other hand, Z sees an iBGP-learned route with an IGP path cost of 2 and the router ID associated with the iBGP session to X.

- γ, which takes a set of BGP routes, C, and produces $C' \subseteq C$, such that routes in C' are the network-wide best routes based on the first four steps in Table I.

Unlike λ_r, γ has *global* (*i.e.*, network-wide) context; that is, its context is not router-specific. When the routers' λ_r functions do not satisfy determinism, each router's best route is not guaranteed to be in the set $\cup_r \lambda_r(E_r)$. In Sections V and VI, we will apply γ to a set of routes when it is safe to eliminate all routes that could *never* be the best route at any router. In these sections, we will see that as long as all routers have either complete visibility of the routes that the AS learns via eBGP or λ_r functions that satisfy determinism, every router will ultimately select a route from $\gamma(E)$.

Symbol	*Description*	*Section*
	Functions on Routes	
λ_r	Takes a set of routes received at router r and outputs the best route at router r, according to the BGP route-selection process applied at router r	III
γ	Takes a set of routes and extracts the subset whose attributes are equally good up through the first four steps of the route-selection process	III
σ	Takes a set of routes and extracts the subset whose attributes are equally good up through the first *three* steps of the route-selection process	V-B
	Sets of Routes or Routers (Initial Inputs)	
R	routers in the AS	III
A	routers that have been activated	VI-B
E	eBGP-learned routes	III
E_r	eBGP-learned routes at router r	III
N	number of eBGP-learned routes (*i.e.*, $\|E\|$)	III
	Sets of Routes (Intermediate and Final Outputs)	
I_r	iBGP-learned routes at router r	III
P_r	All routes learned at router r	III
b_r	The best route that router r selects.	III
C	The set of candidate routes at some intermediate activation. A subset of E.	IV
C_r	The set of candidate routes at router r at some intermediate activation. A subset of E_r.	V-B
B	The set of best routes computed by the algorithm. A subset of C.	IV
L	The set of routes eliminated at some activation step.	V-B
	iBGP Topology	
S	iBGP sessions.	VI-B
G	iBGP session graph. $G = (R, S)$.	VI-B

TABLE II
Description of the notation used in this paper, and the sections where each piece of notation is introduced.

IV. BGP with Determinism and Full Visibility

In this section, we describe an algorithm that predicts the outcome of BGP route selection when a network employs a full mesh iBGP topology and the MED attribute is compared across all routes (which we also refer to as "no MED"

Algorithm: Full Mesh, No MED

SELECTBEST_EBGP(E, R)
// Build the set of locally best routes at each router.
// This set is the set of candidate best eBGP routes.
$C \leftarrow \cup_r \lambda_r(E_r)$

// Eliminate all routes from C that
// do not have highest local preference,
// shortest AS path length, lowest origin type,
// lowest MED
$B \leftarrow \gamma(C)$

Fig. 4. Algorithm for computing the best route at eBGP routers, assuming that MED is compared across all routes (*i.e.*, that there exists a total ordering of routes at each router).

or "without MED").[4] After describing the route-prediction algorithm and proving its correctness, we explain two basic properties that hold in this case that make the prediction algorithm quite simple and explain why two artifacts of BGP—MED and route reflection—can cause these properties to be violated.

A. Algorithm: Full Mesh, No MED

A full-mesh iBGP topology provides full visibility of BGP routes at each router: every router learns the set of routes selected by every eBGP-speaking router in the AS. Furthermore, when the MED attribute is compared across all routes (as opposed to just those from the same neighboring AS) a router's ranking over the set of routes it learns form a total ordering, which implies that determinism is satisfied. These characteristics allow us to devise a relatively simple algorithm to compute the outcome of BGP route selection at each router in the AS.

In this case, the algorithm for computing the best route at every eBGP-speaking router is shown in Figure 4. The algorithm takes as input the set E of all eBGP-learned routes and the set R of all eBGP-speaking routers, and produces the set B of best eBGP routes. E_r refers to all eBGP-learned routes learned by router r, and C represents the set of candidate routes after each router selects the best route from the set of its eBGP-learned routes. The output of this algorithm is $B = \gamma(C)$, the set of all best routes to this destination, such that $b_r = \lambda_r(B)$. The algorithm proceeds in two steps. The first step computes the locally best BGP route at each eBGP-speaking router; this step guarantees that each router selects no more than one eBGP-learned route. The second step eliminates any route from this set for which a router would select another router's iBGP route over its own locally best route.

The first step of the algorithm scans all N eBGP-learned routes and selects the best eBGP-learned route at each router, if any; at most $|R|$ routes remain after this step. The second step selects, for each router $r \in R$, the best route from R. Thus, the running time will be $O(N + |R|^2)$, where N is the number of eBGP-learned routes, and $|R|$ is the number of routers in the system (a full mesh iBGP configuration will have $|R|(|R|-1)$ iBGP sessions). When $|R| > N$, the N term is dominated, so the running time is $O(|R|^2)$. When $N > |R|$, however, a simpler approach to the algorithm would simply be to apply $\lambda_r(E)$ at each router, which has $O(N|R|)$ running time. Thus, the computational complexity for route prediction is proportional to the total number of routes in the system.

To prove that this algorithm is correct, we must show that this algorithm accurately emulates *one* activation sequence; Observation 1 guarantees that as long as the algorithm correctly emulates some activation, it will correctly emulate BGP route selection.

Theorem 1: When each router can produce a total ordering over all possible candidate routes, the algorithm in Figure 4 correctly computes the outcome of the route-selection process for all routers that select an eBGP-learned route as their best route.

Proof. We prove this theorem constructively, by showing that the algorithm correctly emulates an activation sequence and message ordering that could occur in BGP. Consider the following ordering:

1) All routers receive routes to the destination via eBGP. Then, every router is activated simultaneously.
2) Every router advertises its locally best route via iBGP. After all iBGP messages have been exchanged, every router is activated simultaneously.

In the first phase, each router r computes $\lambda_r(E_r)$, resulting in a set of candidate routes $C = \cup_r \lambda_r(E_r)$, as in the first line of the algorithm in Figure 4. Then, each router learns these routes. Note that $B \subseteq C$ by definition, which means that each router that learns a route to the destination via eBGP has either zero or one route in B. We consider both cases. If a router r has a route in C but not in B, then r's eBGP-learned route $b_r = \lambda_r(E_r)$ must have been worse according to the first four steps of the route-selection process than some other route, $b_s = \lambda_s(E_s)$ in C (otherwise, $\gamma(C)$ would not have eliminated it). But in a full mesh iBGP topology, r would learn a route via iBGP that is at least as good as b_s, so b_r would also be eliminated in phase 2 of the activation. Of course, if a router has a route in C, then that must be the route that it would select after phase 2 of activation: it is equally good as all routes in $\gamma(C)$ through the first four steps of the route-selection process (by construction), and it prefers its own best route over any iBGP-learned route (by step 5 of the route-selection process). ■

B. Importance of Determinism and Visibility

The simple algorithm in Figure 4 works because two properties hold. First, when MED is compared across all routes, every router that selects a route from the set of eBGP-learned routes will select its locally best route. Second, when the iBGP topology is a full mesh, each BGP-speaking router ultimately

[4]This scenario may be the case for many small stub ASes that do not have customers of their own: a network that does not have many routers will typically configure its iBGP topology as a full mesh, and a stub AS typically does not receive (or honor) MEDs from the ASes from which it buys transit. In practice, some transit ISPs even configure their routers to compare the MED attribute across all candidate routes (often to avoid problems with oscillation), and most small networks do not use route reflection.

§	*MED*	*RR*	*Running Time*	Prop. 1	Prop. 2
IV	No	No	$O(N + \|R\|^2)$	•	•
V-B	Yes	No	$O(N \log N + N\|R\|)$		•
VI-B	No	Yes	$O(N + \|S\|)$	•	•
VI-C	Yes	Yes	—		

TABLE III

PROPERTIES OF THE BGP ROUTE PREDICTION ALGORITHMS IN EACH OF THE THREE CASES.

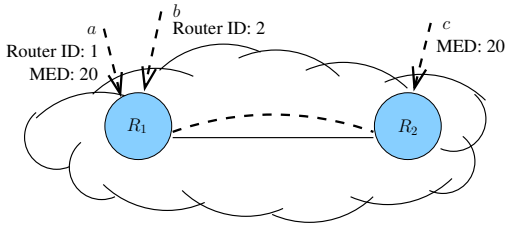


Fig. 5. With MED, a router may select a route that is no router's best eBGP route, thus violating Property 1.

selects a route in $\gamma(E)$; that is, every router ultimately selects a route that has the maximum local preference, minimum AS path length, lowest origin type, and lowest MED (assuming MEDs are compared across all routes). Table III summarizes when these two properties hold, for all possible combinations of MED and route reflection (the rest of this section treats defines these two properties more formally). The table also indicates the computational complexity for computing the best route at each router, for each scenario. We now formalize these two properties, explain why they make route prediction simple, and present cases where BGP violates each of them.

Every best route is some router's locally best eBGP route. This property holds only if every router's $\lambda_r()$ function satisfies determinism. We now formalize this property, prove that determinism is required to ensure that it holds, and show an example where this property is violated if BGP does not satisfy determinism.

Property 1: If determinism is satisfied, then each router ultimately either selects its own best eBGP-learned route or some iBGP-learned route. Formally, $b_r \in E_r \Rightarrow b_r = \lambda_r(E_r)$.

Proof. By definition, each router r applies the route-selection process to the union of the routes it learns via eBGP and iBGP: $b_r = \lambda_r(E_r \cup I_r)$. Therefore, either $b_r \in E_r$ or $b_r \in I_r$. Furthermore, because determinism is satisfied, the router r's preferences over routes in $E_r \cup I_r$ form a total ordering, so either $b_r = \lambda_r(E_r)$ or $b_r = \lambda_r(I_r)$. But, if $b_r \neq \lambda_r(E_r)$, then $b_r = \lambda_r(I_r)$, so $b_r \in I_r$ and $b_r \notin E_r$. ■

Property 1 makes it possible to propagate the effects of route selection at each router only once, because each router ultimately selects either its locally best eBGP-learned route or some other router's locally best route.

Unfortunately, when the MED attribute is only compared among routes from the same AS, BGP does not satisfy determinism, so this property no longer holds. Figure 5 shows an example where this property is violated. In this example, router R_1's ranking between a and b depends on whether it learns route c. Thus, even though route a is R_1's locally best route (by the router ID tiebreak), R_2 ultimately selects route b (c eliminates a due to MED, and R_1 selects b over a because b is learned via eBGP), which is *no* router's best eBGP route. As such, the simple algorithm that selects each router's best route from the set of best eBGP routes does not work: a naïve algorithm would result in precisely the "back and forth" behavior described in Section I-B. Section V describes an alternate route prediction algorithm that handles this case.

Every best route is in $\gamma(E)$. This property states that every router selects a route that is equally good up through the MED comparison step of the route-selection process. Intuitively, it might seem that this property would always hold—why would a router ever select a route with a lower local preference, longer AS path, higher origin type, or higher MED value if it had a better route available? In fact, in certain iBGP configurations, a route reflector can prevent a router from *learning* a route with a lower MED value than the one it selects. Property 2 holds if either the iBGP topology is a full mesh or determinism is satisfied. We now formally state the conditions when this property holds, show an example where a BGP configuration can violate this property, and briefly discuss its implications for route prediction.

Property 2: If (1) every router in the AS receives the best eBGP-learned route from every other router in the AS or (2) all route attributes are compared across all routes (*i.e.*, it is possible to construct a total ordering over all routes) and every router receives at least one route in $\gamma(E)$, then every router r will ultimately select a route, $b_r \in \gamma(E)$, where E is the set of all eBGP-learned routes.

Proof. Define $P_r \subseteq E$, the set of routes that router r learns (*i.e.*, $P_r = E_r \cup I_r$). Assume that some router r selects $b_r = \lambda_r(P_r) \notin \gamma(E)$. This property implies that $P_r \cap \gamma(E) = \phi$ (*i.e.*, that P_r contains *no* routes in $\gamma(E)$; otherwise, b_r would be better than all routes in $\gamma(E)$, which contradicts the definition of γ. But, if $P_r \cap \gamma(E) = \phi$, then the iBGP topology is such that r does not learn all routes, because at least one router $s \in R$ selects a route from $\gamma(E)$, and router r would have learned that route from s. If path visibility is satisfied and $b_r \notin \gamma(E)$, this also implies that some route attribute is not compared across all routes (*i.e.*, it is not possible to form a total ordering): otherwise, given a total ordering, if one router selects a route from $\gamma(E)$, then every router either learns that route and selects it, or selects its own route (which must be in $\gamma(E)$, by total ordering) and propagates that route. ■

Property 2 makes it possible to compute the route that each router r selects by applying λ_r to the set of *all* locally best routes, B (*i.e.*, $b_r = \lambda_r(B)$), thus eliminating other routes.

Unfortunately, this property is not guaranteed when determinism is violated and every router does not learn every eBGP-learned route. Consider the example shown in Figure 6. The network learns routes to some destination at routers X, Y, and Z that are equally good up to MED comparison. All three routers are clients of the route reflector RR. The routes

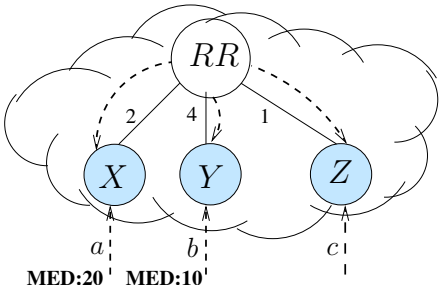


Fig. 6. When an AS's iBGP topology uses route reflectors and MED, a router may not always select a route in $\gamma(E)$.

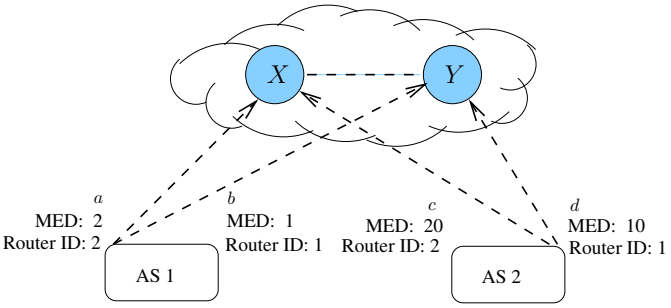


Fig. 7. Interaction between MED and router ID in the BGP route-selection process. X and Y are routers, each with direct eBGP sessions to ASes 1 and 2. a, b, c, and d are routes learned via eBGP.

at X and Y are learned from the same next-hop AS, and r_Y has a lower MED value. One might think that router X would never select route a, since, after all, it has a higher MED value than route b, but that is not the case in this figure: RR learns routes a, b, and c, and selects route c as its best route, because c has the shortest IGP path cost. As a result, X never learns route b.

When Property 2 is not satisfied, route prediction must essentially resort to simulation. The problem in this case is that it is impossible to know when activating any given router that it is safe to eliminate *any* route that it learns via eBGP. We discuss this problem in more detail in Section VI.

V. Route Prediction without Determinism

In this section, we present how to model path selection when the MED attribute is compared only across routes learned from the same AS, rather than across all routes for a destination prefix. MED prevents each router from having a total ordering over all possible candidate routes, so it is actually possible to have $b_r \in E_r$ without $b_r = \lambda_r(E_r)$. In Section V-A, we describe this problem in more detail and describe why the simple approach presented in Section IV fails; then, we present an algorithm that accurately computes the outcome of BGP path selection when MED is compared only across routes from the same AS.

A. Problems Introduced by MED

The algorithm from Section IV assumes that each router's ranking between two routes is independent of whether other routes are present (*i.e.*, $\lambda_r(\{a,b\}) = a \Rightarrow \lambda_r(\{a,b,c\}) \neq b,\ \forall a,b,c$). When MED is only compared across routes from the same AS, the algorithm cannot simply select the locally best route at each router, because a router may ultimately select a best route that it learned via eBGP that was not its locally best route. This point has serious implications, because we can no longer assume that if a router selects an eBGP-learned route to a destination, that eBGP-learned route will be that router's locally best route; rather, the route that the router ultimately selects may be worse than the "best" route at that router when compared only against routes learned via eBGP at that router. Thus, the approach from Section IV, which computes b_r by taking the locally best route at each router from $\gamma(E)$, may not compute the correct result. Using the example in Figure 7, we explain why two seemingly-natural approaches to computing the routes do not work:

- *Local route elimination is not correct.* The algorithm in Figure 4 would first apply $\lambda_r(E_r)$ at each router. In Figure 7, given the choice between the two eBGP-learned routes a and c, router X prefers c, because c has a smaller router ID. Between routes a, c, and d (which it learns via Y), however, router X prefers route a, because route d eliminates route c due to its lower MED value. Thus, router X's preference between routes a and b depends on which route Y selects. The algorithm in Figure 4 would compute $\lambda_X(\{a,c\}) = c$ and $\lambda_Y(\{b,d\}) = d$ (resulting in $C = \{c,d\}$), and ultimately compute $B = \{d\}$ because d has a smaller MED value than c. In reality, though, router X would select route a over d, because a is an eBGP-learned route from a different neighboring AS.
- *Global route elimination is not correct.* It might also seem reasonable to apply γ globally, followed by applying λ_r locally at each router. In Figure 7, a global comparison of the routes (*i.e.*, applying $\gamma(\{a,b,c,d\})$), would first eliminate a and c based on MED, and then router X would select route d (because d is preferred to b based on the router ID comparison applied at router Y). This conclusion is incorrect, because X would *always* prefer route a over route d, because a is learned via eBGP (step 5) and a and d are equally good up through step 4 (recall that a router does not compare the MEDs of routes with different next-hop ASes).

The crux of the problem is that the MED attribute makes it impossible to produce an ordering of the routes at X that is independent of the presence or absence of other routes.

B. Algorithm: Full Mesh, MED

To correctly handle the interaction between the MED and router ID attributes, the algorithm emulates a message ordering that propagates the effects of MED on each router's best route. Figure 8 summarizes this algorithm. For this algorithm, we define a new function, σ, which takes a set of routes and returns all routes equally good up through the first three steps of the BGP route-selection process (*i.e.*, local preference, AS path length, and origin type). When applied to the network in Figure 7, the algorithm starts with all routes in $\sigma(E)$ and proceeds as follows:

Algorithm: Full Mesh, MED

```
SELECTBEST_EBGP_MED(E, R)
    // Eliminate all routes from C that
    // do not have highest local preference,
    // shortest AS path length, lowest origin type
    C ← σ(E)
    B_0 ← φ; i ← 0
    do
        B_{i+1} ← ∪_r λ_r(C_r ∪ B_i); i ← i + 1
    while B_{i+1} ≠ B_i
    return B_i
```

Fig. 8. Algorithm for computing the best route at eBGP routers, assuming that MED is only compared across routes *from the same neighboring AS.*

1) B_1 gets the locally best routes from X and Y: c and d, respectively. That is, $B_1 = \{c, d\}$.
2) On the second iteration, X compares the routes from C that it learns via eBGP, a and c, along with route d from B_1, so $\lambda_X(\{a, c, d\}) = a$. Similarly, $\lambda_Y(\{b, c, d\}) = d$. Thus, $B_2 = \{a, d\}$.
3) On the third iteration, the process repeats, and $B_3 = \{a, d\}$, at which point the algorithm terminates.

This algorithm computes the correct routing decision for each router: a at router X and d at router Y. At router Y, d is better than a (step 5), b (step 7) and c (step 4). At router X, a is better than d (step 5); a is not better than b, but this does not matter because router Y does not select b, and a is not better than c, but this does not matter because c is always worse than d (step 4).

Theorem 2: When MED is compared only across routes from the same neighboring AS, the algorithm from Figure 8 accurately emulates the results of one activation sequence and message ordering for all routers that select an eBGP-learned route as their best route.

Proof. Computing $\sigma(E)$ produces the set C, which is simply the set of eBGP-learned routes, E, minus the routes that could *never* be the best route at any router (*i.e.*, because they have a lower local preference, longer AS path length, or higher origin type). Because the iBGP topology forms a full mesh, as long as there is a route in E at *any* router that is better in the first three steps of the route-selection process, no router will select a route that is not in $\sigma(E)$. The remainder of the algorithm evaluates a routing system with the routes in $\sigma(E)$.

The remainder of the algorithm follows an activation sequence where each phase (or iteration of the loop) activates all of the routers simultaneously. The proof proceeds by induction. After the first iteration of the loop, $B_0 = \phi$ and $b_r = \lambda_r(C_r)$, where C_r is all of the routes learned at router r via eBGP with the highest local preference, shortest AS path length, and lowest origin type. By definition, $\lambda_r(C_r)$ returns each router's locally best route according to the BGP route-selection process, which is the same as that which the BGP route-selection process would select for each router after phase 1 of the activation sequence. In a network with a full mesh iBGP configuration, each router r then sends its locally best route, b_r, to every other router.

Suppose the algorithm correctly computes the outcome of the BGP route-selection process for the first i iterations of the activation sequence. Suppose that there is some router r for which the algorithm, at iteration $i + 1$, computes $b'_{r,i+1}$, the element in B_{i+1} that is the best route at router r, such that $b'_{r,i+1} \neq b_{r,i+1}$. Then, it must be the case that $b_{r,i+1} \notin C_r \cup B_i$; otherwise, $\lambda_r(C_r \cup B_i)$ would also have selected $b_{r,i+1}$. Either $b_{r,i+1}$ is an eBGP-learned route or it is an iBGP-learned route. If it is eBGP-learned, then it must be in C_r, as we previously established. If it is iBGP-learned, then it must be in B_i, because every iBGP-learned route is the best route of some other router in the AS. But if either $b_{r,i+1} \in C_r$ or $b_{r,i+1} \in B_i$, then $b_{r,i+1} \in C_r \cup B_i$, which is a contradiction.

The algorithm terminates when $B_i = B_{i+1}$; that is, when activating all of the routers in the AS does not cause any router to select a new best route and generate a new BGP update message. We have shown that the algorithm correctly predicts the outcome of BGP route selection after k iterations for any k. Further, we assumed that the routing system satisfies safety; that is, given a stable topology, it is guaranteed to converge to a path assignment where no router changes its best route. When the BGP routing system converges to this path assignment, no router changes the route it selects and, hence, no new routing messages are generated. Since, after i iterations, the algorithm correctly predicts the outcome of BGP route selection and the algorithm activates every router in the AS on every iteration, then it will terminate precisely when it has reached the BGP path assignment when no new BGP messages are generated (*i.e.*, the unique solution). ■

The algorithm in Figure 8 is correct, but it is not efficient: each iteration of the loop repeatedly considers routes that have been "eliminated" by other routes. A more efficient algorithm would eliminate routes from consideration at each iteration if we know that they could never be the best route at any router—such is the spirit of applying $\sigma(E)$ across the initial set of routes. Unfortunately, because the MED attribute is not comparable across all routes, it is possible for a route that is not in the set B_i to emerge in the set B_j for some $j > i$. We now formally define a condition under which routes may be eliminated, which will allow us to devise a more efficient prediction algorithm.

Lemma 1: Suppose there exist two routes: (1) $s \in C_r$ at router r and (2) $t \in C_{r'}$ at router $r' \neq r$. If $t \in B_i$, $\lambda_r(s, t) = t$, and router r learns route t (*e.g.*, as in a full mesh iBGP configuration), then $s \notin B_j \ \forall j > i$.

Proof. First, note that as long as $t \in B_j$, then $s \notin B_j$ because route t is preferable to s Also note that because all routes in C are equally good up the MED comparison and eBGP-learned routes are preferred over iBGP-learned routes, we know that $\lambda_r(s, t) = t$ because $\text{MED}(t) < \text{MED}(s)$. Now, suppose there exists some $j > i$ for which $t \notin B_j$. Call the best route at router r' at step i, $v = \lambda_{r'}(C_{r'}) \neq t$; again, we know that $\text{MED}(v) < \text{MED}(t)$. But this means that $\text{MED}(v) < \text{MED}(s)$, $\lambda_r(s, v) = v$, and, thus, $s \notin B_j$. ■

Algorithm: Full Mesh, MED (Efficient Algorithm)

SELECTBEST_EBGP_MED(E, R)

// Eliminate all routes from C that
// do not have highest local preference,
// shortest AS path length, lowest origin type
$C \leftarrow \sigma(E)$

// Keep track of the best routes at each router.
do
$B \leftarrow \cup_r \lambda_r(C_r)$
$L \leftarrow B \setminus \gamma(B)$
$C \leftarrow C \setminus L$
while $L \neq \phi$

Fig. 9. Computationally efficient algorithm for computing the best route at eBGP routers, assuming that MED is only compared across routes *from the same AS* (*i.e.*, that there is no total ordering of routes).

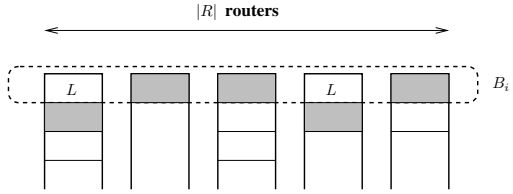


Fig. 10. Implementation of the route computation algorithm from Figure 9. Each stack represents one of $|R|$ total routers, and each stack element represents one of L routes. The top elements of the $|R|$ stacks represent B_i, the elements marked L represent routes that are worse than the routes at the top of the remaining stacks according to the first four steps of the route-selection process (*i.e.*, local preference, AS path length, origin type, MED), and the shaded routes represent B_{i+1}. The algorithm terminates when no routes are marked L.

We can use this result to devise a more efficient route prediction algorithm that eliminates, at every iteration, a router's locally best route if it has a higher MED value (and same next-hop AS) than some other router's locally best route. This algorithm is described in Figure 9 and shown conceptually in Figure 10; it can also be thought of in terms of an activation sequence: (1) each router learns routes via eBGP, selects a locally best route, and readvertises via iBGP; (2) each router compares its locally best route with all other routes learned via iBGP, and *eliminates* its own locally best route from the system if it is worse than some other locally best route at another router; (3) the system is restarted (from phase 1) with the eliminated routes removed. This algorithm is computationally more efficient than the one in Figure 8; we now analyze its running time complexity.

Computational Complexity. Understanding the running time of the algorithm in Figure 9 is easiest when we consider the implementation of the algorithm shown in Figure 10. In this figure, the eBGP-learned routes at each router are represented as a stack and are sorted *locally* (*i.e.*, compared only to other routes learned at the same router). The top of the stack represents the best route learned at that router; the route that is second from the top is the second best route, and so forth. Then, the algorithm from Figure 9 can be interpreted as follows:

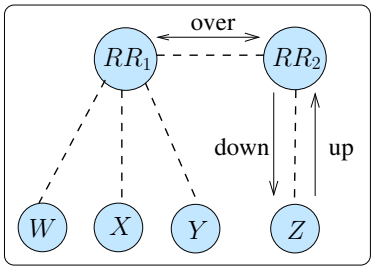


Fig. 11. Example iBGP signaling graph.

- $B \leftarrow \cup_r \lambda_r(C_r)$ is the union of all of the elements at the top of the stack and does not need to be computed explicitly, assuming each stack is sorted. The complexity of sorting N routes distributed across $|R|$ stacks is $O(N \log N)$. Each of N routes may be inserted into as many as $|R|$ stacks, so the complexity of this step is $O(N \log N + N|R|)$.
- $L \leftarrow B \setminus \gamma(B)$ marks a route at the top of a stack if that route is worse than any route at the top of another stack, according to the first four steps of the BGP route-selection process. This process takes at most two scans of the routes at the top of the $|R|$ stacks, so the running time is $O(|R|)$.
- $C \leftarrow C \setminus L$ "pops" the marked routes from the top of the stacks, where appropriate. This process requires a single scan through $|R|$ stacks and at most $|R|$ pop operations, so the running time is $O(|R|)$.

In the worst case, the above three steps repeat until $N-1$ routes are popped from the stacks, and each iteration only pops a single route. Thus, in the worst case, the running time for the algorithm is $O(N \log N + N|R|)$.

VI. Route Prediction without Full Visibility

A full mesh iBGP topology does not scale to large networks because a network of $|R|$ routers requires $O(|R|^2)$ iBGP sessions. Network operators use a technique called *route reflection*, which improves scalability by introducing hierarchy but complicates route prediction. First, we define an iBGP signaling topology, expound on problems introduced by route reflection, and describe constraints on iBGP configuration that must hold for route prediction to be possible. Next, we propose an algorithm that efficiently computes the outcome of BGP path selection in a network with route reflection; we then present a minor modification to the algorithm that is necessary if MED is only compared across routes from the same neighboring AS.

A. Problems Introduced by Route Reflection

A router does not normally forward iBGP-learned routes over other iBGP sessions, but it can be configured as a *route reflector* (RR), which forwards routes learned from one of its route-reflector clients to its other clients. The routers in an AS form a directed graph, $G = (R, S)$, of iBGP sessions called a *signaling graph*. Each edge $a = (u, v) \in S$ where $u, v \in R$ corresponds to an iBGP session between a pair of routers. We

then define three classes of edges: (1) $a \in$ **down** if v is a route-reflector client of u; (2) $a \in$ **up** if u is a route-reflector client of v; and (3) $a \in$ **over** if u and v have a regular iBGP session between them. Figure 11 shows an example signaling graph. In a full-mesh configuration, every eBGP-speaking router has an edge in **over** with every other router in the AS, and both the **up** and **down** sets are empty.

Previous work has shown that iBGP satisfies safety as long as the structure of the signaling graph satisfies certain sufficient conditions [15]. Accordingly, we refine Constraint 2 in terms of these sufficient conditions to guarantee that an iBGP topology with route reflection satisfies safety (at least when the MED attribute is not used or compared across all routes):

Constraint 4: (1) $\forall \quad u, v, w \quad \in \quad R, \quad ((u, v) \quad \in$ **down** and $(u, w) \notin$ **down**$) \Rightarrow \lambda_u(\{\rho_v, \rho_w\}) = \rho_v$, where ρ_v represents any route learned from v and ρ_w is any route from w; and (2) the edges in **up** are acyclic.

Part (a) is satisfied when routers do not change the attributes of iBGP-learned routes and each router has a lower IGP path cost to its clients than to other routers. The common practices of applying import policies only on eBGP sessions and placing route reflectors and their clients in the same point-of-presence (*i.e.*, "PoP") ensure that these conditions hold. Part (b) states that if a is a route reflector for b, and b is a route reflector for c, then c is not a route reflector for a, consistent with the notion of a route-reflector *hierarchy* (rather than an arbitrary signaling graph).

Even a route reflector configuration that converges can wreak havoc on the algorithms from Sections IV and V. A route reflector hides information by advertising only a *single best route to its iBGP neighbors*. For example, in Figure 11, if W and Z have eBGP-learned routes, router Y learns a single route from its route reflector RR_1. Suppose that RR_1 selects the eBGP route advertised by Z. Then, Y would pick Z's route as well, even if Y would have preferred W's route over Z's route. Note that Y makes a different routing decision than it would if it could select its best route from all the eBGP routes (*i.e.*, from both W and Z). In large networks, route reflection reduces the number of routing messages and iBGP sessions, which helps scalability, but it complicates route prediction in the following ways:

1) A router will not typically learn every route that is equally good up through the first four steps of the route-selection process. That is, it is possible (and likely) that some routers will not learn every route in $\gamma(B)$. In Section VI-B, we describe an algorithm that handles this case.
2) If a network uses route reflectors, *and* MED is only compared across routes from the same AS, the routes that some routers ultimately select may be *worse* than some eBGP-learned routes, according to the first four steps of the route-selection process. That is, it may be the case that $b_r \notin \gamma(E)$ for some router r. This characteristic creates problems not only for efficient route prediction, but also for safety. We discuss this case in Section VI-C.

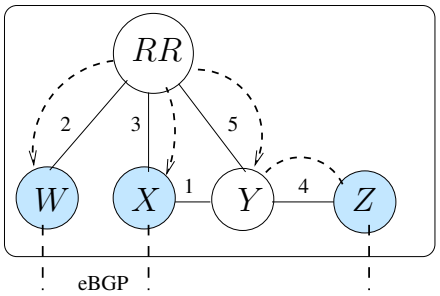


Fig. 12. When an AS's iBGP topology uses route reflectors, a router may not always discover the route corresponding to its closest egress router.

B. Algorithm: Route Reflection, No MED

Route reflection obviates the need for routers in an AS to form a full mesh topology, but it also means that some routers may not learn all routes in $\gamma(B)$. This artifact has two implications. First, the algorithm cannot simply assign a non-eBGP-speaking router the route from the "closest" eBGP-speaking router, because the former router may never learn the route. Thus, applying $b_r \leftarrow \lambda_r(B)$ may not always be correct. For example, consider the network shown in Figure 12. W, X, and Y are clients of route reflector RR, and Z is a regular iBGP peer of Y. X and Y have a short IGP path between them, but they are *not* directly connected by an iBGP session. Routers W, X, and Z have eBGP routes that are equally good through the first four steps of the route-selection process, and have thus selected their own eBGP-learned routes. In this network, Y's closest egress point is X, but Y selects W, because RR's closest egress router is W.

Second, often there is *no consistent ranking of possible egress routers* from some non-eBGP-speaking router. For example, in Figure 12, RR prefers egress router W because its IGP path cost to W is the shortest. Router Y's preferences over possible egress routes depends on the presence or absence of other routes. If the AS learns routes for some destination via eBGP sessions at routers X and Z, then router Y prefers using X as an egress router. On the other hand, if the AS learned routes at W, X, and Z, then Y prefers using Z, which implies that Y prefers egress Z over X and is inconsistent with Y's choice when only X and Z are available egress routers.

To account for the fact that some routes are not visible at some routers, we design an algorithm that emulates a certain activation sequence, making route assignments at each router where possible and propagating the effects of these decisions to other routers, without ever having to revisit any assignment. This algorithm is shown in Figure 13. The algorithm first activates the routers from the bottom of the route-reflector hierarchy upwards, which guarantees that each router selects a *down* route where possible, as required by Constraint 4(a). Because the algorithm moves upwards from the bottom of the hierarchy, it performs computations for each route reflector after all of the routes from its clients become known; computations for these routers never need to be revisited, since, by Constraint 4, a router always prefers routes from its "children" (*i.e.*, clients) over routes from its peers or parents. Visiting the routers in the *down* direction ensures that the

Algorithm: Route Reflection, No MED

SELECTBEST_EBGP_RR(E, R)

// Proceed up the hierarchy, assigning best routes.
// Find a router for which all children are activated.
$A \leftarrow \phi$
while $\exists r \in R$ s.t. $r \notin A$ and $c \in A \; \forall c \in \text{DOWN}(r)$
  $I_r \leftarrow \cup_{c \in \text{DOWN}(r)} b_c$
  $b_r \leftarrow \lambda_r(I_r \cup E_r)$
  $A \leftarrow A \cup r$

// Proceed down the hierarchy.
// Find a router for which all parents are activated.
$A \leftarrow \phi$
while $\exists r \in R$ s.t. $r \notin A$ and $c \in A \; \forall c \in \text{UP}(r)$
  $I_r \leftarrow \cup_{c \in \text{UP}(r) \cup \text{OVER}(r)} b_c$
  $b_r \leftarrow \lambda_r(I_r \cup b_r)$
  $A \leftarrow A \cup r$

Fig. 13. Algorithm for computing the best route at each router in a network with route reflection but no MED.

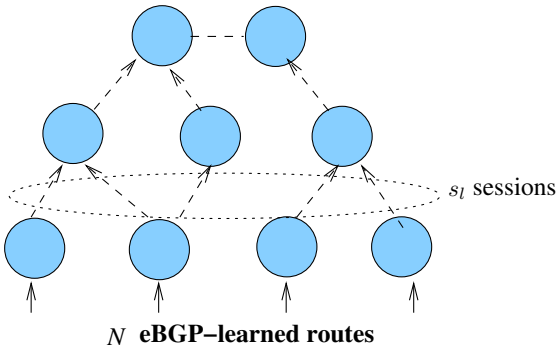


Fig. 14. Running time analysis of an iBGP graph walk for the algorithm in Figure 13.

algorithm performs computations for the remaining routers using all available routes from the **up** and **over** sets. The algorithm defines two partial orderings of the routers based on the elements of the **up** and **down** sets. We can define these two partial orderings because Constraint 4(b) requires that the signaling graph does not have any cycles of these edges, so each partial ordering must have a top and bottom element.

Applying this algorithm to the example in Figure 12, the shaded routers select best routes in the first step, because each of those routers is at the bottom of the hierarchy and, thus, all of their neighbors in **down** have been activated (because they have none). Y is activated, but it does not select a route at this point because it has no neighbors in **down**. Because these four routers are at the same level in the hierarchy, they can be activated in any order. Then RR is activated; it applies $\lambda_{RR}(\{r_W, r_X\})$ and selects r_W because it has the smallest IGP path cost. The routers are all activated again in the downward direction; Y receives r_W from RR and compares it with r_Z, which is its best route to the destination. X and Z also receive r_W but continue to select their own route, because λ_r prefers eBGP routes over iBGP routes. We now prove that the algorithm shown in Figure 13 is correct.

Theorem 3: If each router can form a total ordering over the set of all candidate routes, then the algorithm in Figure 13 correctly computes the outcome of the BGP route-selection process, b_r, for all routers $r \in R$.

Proof. Assume that some router r selects a route, b_r, that is different from the route assigned by the algorithm in Figure 13, b'_r. The mismatch can occur in one of two cases: (1) when b_r is learned from a session in **down**, or (2) when b_r is learned from a session not in **down** (*i.e.*, in either **up** or **over**).

Consider Case 1, where b_r is learned from a session in **down**. Call b'_r the first case of an incorrect computation (*i.e.*, the algorithm has correctly computed the best route for all routers below r in the hierarchy); because we examine the first such mismatch, I_r is correct. If b'_r is also in **down**, then $b'_r = \lambda_r(I_r \cup E_r)$ when the algorithm proceeds up the hierarchy, which implies that b'_r is better than b_r according to the BGP route-selection process, and r would have actually selected b'_r. If b'_r is in **up** or **over**, then it must have been the case that it was better, according to the BGP route-selection process, than the displaced route b_r in **down**. But then, by definition of λ_r, router r would have also selected b'_r in BGP. Thus, the algorithm correctly computes b_r for all routers r that select a best route from **down**.

Consider Case 2, where b_r is learned from a session in **up** or **over**. From the first half of the proof, we know that the algorithm correctly computes b_r for all routers that select a route from **down**, so call b'_r the first instance of a mismatch for some router that selects a best route from **up** or **over** (*i.e.*, the algorithm correctly assigns b_r for all routers higher in the hierarchy than r). Again, because we consider the first such mismatch, we know that I_r is correct. If the route that the algorithm selects, b'_r, is in **down**, then, by Constraint 4(a), BGP could not have selected b_r, so we have a contradiction. If both b_r and b'_r are learned from sessions in **up** and **over**, then both are in I_r, and, according to the $\lambda_r(I_r \cup b_r)$ step in the algorithm and by definition of λ_r, both the algorithm and the BGP route-selection process would select the same route. ■

This theorem relates to one from earlier work [20] on sufficient conditions for stable BGP routing *at the AS level*; this work provides a constructive proof showing that the sufficient conditions guarantee safety. In subsequent work, Griffin *et al.* discovered that the sufficient conditions for stable eBGP routing were analogous to those for stable iBGP routing with route reflection [15]. The algorithm from this section applies the iBGP analog of the constructive proof from the work on stable interdomain routing to develop an algorithm for *computing* that stable path assignment.

Computational Complexity. This algorithm traverses the route-reflector hierarchy exactly twice. The running time of this algorithm is $O(N+|S|)$, where N is the number of eBGP-learned routes, and $|S|$ is the number of iBGP sessions. To see why this is the case, consider the l-level route-reflector hierarchy pictured in Figure 14. Starting from the bottom of the hierarchy, the algorithm must perform comparisons over N routes to determine the routes that the M routers at the

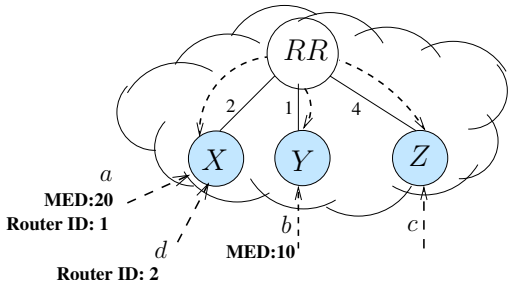
bottom of the hierarchy select (the number of routers at the bottom of the hierarchy is inconsequential: these comparisons can be performed by constructing a subset of M routes from the original N routes, which can be performed in a single scan of the N routes). The algorithm then propagates the selection of these M routes to the next level of the hierarchy, where s_l comparisons must be performed across the routers at the next highest level, where s_l is the number of iBGP sessions at level l. Repeating this process up the hierarchy yields a total running time of $O(N + |S|)$.

Recall from Section IV that the running time for the algorithm in the case of full-mesh iBGP, was $O(N+|R|^2)$, or $O(N + |S|)$. Note that the algorithm for the case with route reflection has the *same* running time complexity as before; the running time for computing the outcome of BGP route selection is no more complex, even though the process for computing the outcome is more involved. In an iBGP topology with route reflection, the number of sessions, $|S|$, will actually be less than $|R|^2$; thus, the running time of the algorithm benefits from the fact that route reflectors reduce the number of sessions in the iBGP topology.

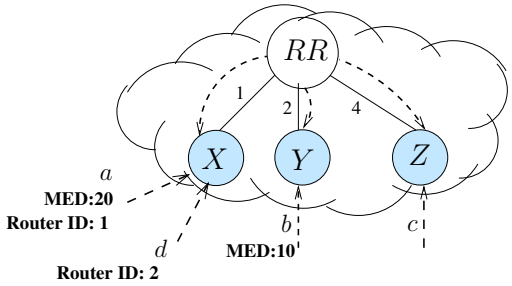
C. Algorithm: Route Reflection, MED

When a network uses both route reflection and MED, the graph walk algorithm in Figure 13 no longer works, because it relies on the fact that all routers will ultimately select a route in $\gamma(E)$. In a network with route reflection *and* MED, this is not always true, because when a router selects a locally best route, a route with a lower MED value might not be visible to that router. As a result, some router in the AS might select an eBGP-learned route that is *worse*, according to the first four steps of the BGP route-selection process, than eBGP-learned routes selected by other routers! Figure 15 shows an example of exactly this scenario.

Note that applying the algorithm from Figure 13 does not always correctly compute the outcome of the BGP route-selection process. Consider the operation of the algorithm from Figure 13 on the topology and route announcements shown in Figure 15(a). Proceeding up the hierarchy: (1) routers X, Y, and Z would select routes a, b, and c, respectively; (2) RR selects route b because b has a lower MED value than a and a shorter IGP path to the egress than c. Proceeding down the hierarchy, X prefers b because it has a lower MED value than a, and ultimately picks d over b because it prefers an eBGP-learned route. At this point, the algorithm in Figure 13 would terminate successfully. But, depending on the IGP topology X's selection of d could have caused RR to select a new best route. Suppose, instead, that the IGP topology were such that RR were closer to X than to Y, as in Figure 15(b). In this case, proceeding up the route-reflector hierarchy a second time would cause RR to change its selected route from b to d; subsequently proceeding down the hierarchy would cause X to change its selected route from d to a. In fact, as described in a similar example in recent work [16], BGP does not satisfy safety in this example—therefore, *no* number of progressions up and down the iBGP hierarchy would cause the algorithm to predict the correct outcome.



(a) When Y is closer to RR than X, the routing system satisfies safety.



(b) When X is closer to RR than Y, the routing system violates safety and the algorithm in Figure 13 is incorrect.

Fig. 15. A BGP configuration where the algorithm in Figure 13 may produce the incorrect result.

In this situation, any router in the AS might ultimately select a route that is not in $\gamma(E)$; as a result, the route-prediction algorithm cannot eliminate a route from the set of candidate routes C_r at any router r, as was done in the case where determinism did not hold but every router was guaranteed to learn every eBGP-learned route (Section V, Figure 9). As we have seen, the fact that a router may select a route that is not in $\gamma(E)$ as its best route, the algorithm (and BGP route selection, for that matter) is no longer guaranteed to terminate.

It might initially seem reasonable to impose constraints on the iBGP and IGP topologies that guarantee safety and can easily be checked with a tool like *rcc* [18]. Unfortunately, as the example in Figure 15 shows, any condition that guarantees safety would require knowledge of the MED attributes of every eBGP-learned route to a destination, not just the iBGP and IGP topologies. Further, the simplicity of this example demonstrates that any condition that guarantees safety for any combination of eBGP routes would be overly restrictive (*i.e.*, it would essentially require not using route reflectors). Thus, in the case where a BGP configuration uses route reflection and only compares the MED attribute across routes from the same AS, the most efficient algorithm for determining the outcome of BGP route selection (and detecting safety violations) is actually a simulator. In other words, there are no conditions on the topology that can be enforced to guarantee that an algorithm would never have to visit each router in the AS more than once, or even that BGP would satisfy safety.

VII. Conclusion

This paper has presented route prediction algorithms that predict the outcome of BGP route selection based on only a static snapshot of the network state. In addition to helping network operators accomplish traffic engineering tasks, these algorithms provide useful insight into the subtleties of network-wide BGP route selection and suggest several directions for improvements to the Internet routing system. For instance, network-wide BGP route prediction could be combined with traffic measurements to help network operators select BGP configuration changes that achieve various traffic engineering goals. In addition, the emulator could be combined with higher-level mechanisms that spot misconfiguration or check that other constraints are satisfied [18].

Although the diagram in Figure 3 shows only three stages, we envision that network operators could incorporate other phases. For example, another phase could combine the predicted forwarding paths with traffic data to predict the load on each link in the network. Using the model for traffic engineering assumes that traffic volumes are relatively stable, and that they remain stable in response to configuration changes. In previous work, we found that prefixes responsible for large amounts of traffic have relatively stable traffic volumes over long timescales [4]. Operators could use the routing model to test configuration changes on reasonably slow timescales that affect prefixes with stable traffic volumes. A network operator could also combine measurements or estimates of the traffic arriving at each ingress router for each destination prefix [21] with the link-level paths to predict the load on each link in the network. Another phase might evaluate the optimality of the these link-level paths in terms of propagation delay or link utilization and could search for good configuration changes before applying them on a live network.

Finally, we note that modeling BGP routing is more difficult than it should be. In the future, we hope that routing protocol designers will consider predictability as a design goal; some of these simplifications that aid protocol modeling also fix problems with protocol *operation*. Routing protocols that are easy to model and reason about will make everyday network engineering tasks more tractable.

Acknowledgments

We thank Hari Balakrishnan, David Clark, Ramesh Johari, M. Frans Kaashoek, and Renata Teixeira, and the anonymous reviewers, for helpful comments and suggestions. Thanks also to Jared Winick for his work on the prototype implementation.

References

[1] Y. Rekhter, T. Li, and S. Hares, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006.

[2] J. Moy, *OSPF Version 2*, Mar. 1994, RFC 1583.

[3] D. Oran, *OSI IS-IS Intra-domain Routing Protocol*, Internet Engineering Task Force, Feb. 1990, RFC 1142.

[4] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for Interdomain Traffic Engineering," *ACM Computer Communications Review*, vol. 33, no. 5, pp. 19–30, Oct. 2003.

[5] "BGP++ Home Page," http://www.ece.gatech.edu/research/labs/MANIACS/BGP++/, 2005, BGP simulator for ns-2.

[6] "C-BGP," http://cbgp.info.ucl.ac.be/, 2005.

[7] "SSFNet," http://www.ssfnet.org/, 2003.

[8] T. Ye and S. Kalyanaraman, "A Recursive Random Search Algorithm for Large-Scale Network Parameter Configuration," in *Proc. ACM SIGMETRICS*, San Diego, CA, June 2003, pp. 196–205.

[9] "How BGP Routers Use the Multi-Exit Discriminator for Best Path Selection," http://www.cisco.com/warp/public/459/37.html.

[10] N. Feamster and J. Rexford, "Network-Wide Prediction of BGP Routes," georgia Tech TR GT-CSS-2006-003.

[11] N. Feamster, "Proactive techniques for correct and predictable internet routing," Ph.D. dissertation, Massachusetts Institute of Technology, Feb. 2006.

[12] N. Feamster, J. Winick, and J. Rexford, "A Model of BGP Routing for Network Engineering," in *Proc. ACM SIGMETRICS*, New York, NY, June 2004, pp. 331–342.

[13] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental Study of Internet Stability and Wide-Area Network Failures," in *Proc. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, Washingon, DC, June 1999, p. 278.

[14] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP Routing Stability of Popular Destinations," in *Proc. ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002, pp. 197–202.

[15] T. Griffin and G. Wilfong, "On the Correctness of IBGP Configuration," in *Proc. ACM SIGCOMM*, Pittsburgh, PA, Aug. 2002, pp. 17–29.

[16] N. Feamster and H. Balakrishnan, "Correctness Properties for Internet Routing," in *Forty-third Annual Allerton Conference on Communication, Control, and Computing*, Monticello, Il, Sept. 2005.

[17] T. Griffin, F. B. Shepherd, , and G. Wilfong, "The Stable Paths Problem and Interdomain Routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 1, pp. 232–243, 2002.

[18] N. Feamster and H. Balakrishnan, "Detecting BGP Configuration Faults with Static Analysis," in *Proc. 2nd Symposium on Networked Systems Design and Implementation (NSDI)*, Boston, MA, May 2005, pp. 43–56.

[19] N. Feamster, R. Johari, and H. Balakrishnan, "The Implications of Autonomy for Stable Policy Routing," in *Proc. ACM SIGCOMM*, Philadelphia, PA, Aug. 2005, pp. 25–36.

[20] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking*, pp. 681–692, Dec. 2001.

[21] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True, "Deriving Traffic Demands for Operational IP Networks: Methodology and Experience," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 257–270, June 2001.

Nick Feamster Nick Feamster is an assistant professor in the College of Computing at Georgia Tech. He received his Ph.D. in Computer science from MIT in 2005, and his S.B. and M.Eng. degrees in Electrical Engineering and Computer Science from MIT in 2000 and 2001, respectively. His research focuses on many aspects of computer networking and networked systems, including the design, measurement, and analysis of network routing protocols, network security, anonymous communication systems, and adaptive streaming media protocols. His honors include award papers at the NSDI 2005 conference (fault detection in router configuration), Usenix Security 2002 (circumventing web censorship using Infranet), and Usenix Security 2001 (web cookie analysis).

Jennifer Rexford Jennifer Rexford is a Professor in the Computer Science department at Princeton University. From 1996-2004, she was a member of the Network Management and Performance department at AT&T Labs–Research. Jennifer is co-author of the book *Web Protocols and Practice*. She serves as the chair of ACM SIGCOMM, and as a member of the ACM Council and the CRA Board of Directors. Jennifer received her BSE degree in electrical engineering from Princeton University in 1991, and her MSE and PhD degrees in computer science and electrical engineering from the University of Michigan in 1993 and 1996, respectively.