Atomic Routing Theory: Making an AS Route Like a Single Node

Rui Zhang-Shen, Yi Wang, Jennifer Rexford
Department of Computer Science, Princeton University
{rz, yiwang, jrex}@cs.princeton.edu

ABSTRACT

The hierarchical structure of the Internet separates routing into two subproblems: routing between Autonomous Systems (ASes) and routing within each AS. The first problem has been studied extensively, assuming that the routers in an AS collectively act as a single node, even though there is no such guarantee today. In this paper, we study how an AS as a whole makes interdomain routing decisions. We introduce Atomic Routing Theory (ART)—a model that captures how a distributed collection of routers can correctly realize an AS's routing policy—and analyze the fundamental tradeoff between protocol overhead and policy flexibility. We identify three reasons why today's BGP-speaking ASes are not atomic: (i) a router cannot assign different routes to different links, (ii) route selection is too tightly coupled with route dissemination, and (iii) routers may deflect data packets from the chosen interdomain route. Our proposed solutions to these problems, which result in an enhanced protocol called *atomic BGP*, are easily implementable, introduce minimum overhead, and, since the changes are local to an AS, can be deployed incrementally. Atomic BGP guarantees policy correctness, simplifies router configuration, and closes the gap between interdomain routing assumptions and realities.

1. INTRODUCTION

The separation of inter- and intradomain routing—an important founding principle of the Internet—enables the distributed management of resources by different Autonomous Systems (ASes), each with its own *routing policy*. These uncoordinated, possibly conflicting policies, are reconciled by the inter-AS routing protocol. However, how the many routers *within* an AS can realize a policy is not well understood. This paper presents a precise model of routing policy and conditions for an AS to realize its policy in a distributed fashion. Applying our theory to today's BGP-speaking ASes, we identify several problems that can cause an AS to violate its policy, and present practical solutions that guarantee correctness.

1.1 "Atomic" Autonomous Systems

Despite many years of implementing and managing policy-based interdomain routing, the precise definition of an AS remains elusive. An AS is often loosely defined as a collection of routers and links managed by a single administrative entity, but this says nothing about routing. Having two levels in the routing hierarchy means that the upper (inter-AS) level should be able to ignore the lower (intra-AS) details and treat an AS as a node. The behavior of this "node" is specified by the AS's policy, which dictates how the AS selects and propagates routes. In fact, the very first attempt to formally define an AS requires it to have "a SINGLE and CLEARLY DEFINED routing policy" [1]. We say an AS is *atomic* if it realizes its policy, and therefore can be described by the policy alone, without details of its internal structure.

What does it mean for an AS to realize its policy? Loosely speaking, a policy is a combination of objectives, which reflect the AS's goals (such as security and balanced load) and obligations (typically specified by business contracts). An AS has a set of edge links that connect to other ASes. The AS learns a set of routes and solves a *route assignment problem* by assigning a subset of these routes to each edge link. We say that an AS *realizes* its policy if the route assignment does not violate any of its policy objectives and the AS forwards packets according to the route assignment. For example, if the AS has an objective to avoid routes that traverse a known hostile AS, then such a route cannot be assigned to any link; if an objective is to use only the shortest routes, then longer routes should not be assigned.

How can a distributed collection of routers realize a policy? Suppose for a moment that an AS consists of a single router. That router would learn all the routes, and assign routes to all its links according to the AS's policy. The landscape changes in several interesting ways when an AS has multiple routers:

- A router may not be able to learn every route, due to scalability concerns for route dissemination.
- Multiple routers may connect to the same neighbor AS, which may have expectations about the routes advertised at different locations.
- Data packets may traverse multiple routers in the AS, and some of these routers may have selected different routes.

These challenges make it difficult for a distributed collection of routers to realize even some of the most basic AS-level policies correctly, despite considerable flexibility in specifying how each *individual router* selects and advertises routes.

In this paper we study intra-AS routing by first precisely defining ASes and policies, and then examining what it takes for an AS to realize its policy. We present a mathematical model of an *AS*, which solves the *route assignment problem* (RAP), as well as a mathematical model of a *policy*, which acts as constraints on RAP. In addition, we study atomicity in a distributed setting, and explore the tradeoff between protocol overhead and policy flexibility. Our Atomic Routing Theory (ART) is not only theoretically interesting, but also makes the following practical contributions: it can discover potential policy violations, especially when new mechanisms and new policy objectives are introduced; it gives the necessary conditions for realizing a given policy in a distributed setting; and it provides the theoretical foundation to prove the atomic properties of a newly proposed protocol. In this paper we focus on the first two applications of ART and propose changes to fix today's ASes.

1.2 Making BGP-Speaking ASes Atomic

BGP was created to exchange network reachability information among ASes, and it does this job reasonably well. But in an Internet with a variety of business relationships (such as customer-provider, where the customer pays for connecting to the Internet through the provider, and peer-peer, where they carry traffic between each other's customers for free [2]), providing connectivity alone is not enough. Network operators need more control over which routes are selected and which are advertised to neighboring ASes. As more and more features were added to BGP, not enough attention was given to whether BGP can *simultaneously and correctly* realize all the policy objectives [3, 4]. Today, BGP *cannot* jointly realize some common policy objectives, as we will show in the paper, and ART helps us trace the cause to two protocol peculiarities, one in route selection, and another in intra-AS route dissemination.

First, a BGP router selects only one best route for a destination, and then for each link, decides whether or not to assign the chosen route. This precludes a router from assigning different routes to different edge links, and artificially constrains the route assignment when two links terminate on the same router. Now an AS's behavior cannot be explained by its policy alone, and its internal structure starts to matter—a sign that it has lost atomicity. This limitation can cause policy violations that can only be addressed by terminating certain links on different routers. We argue that *a router should independently assign routes to each of its links*. This means that links to other ASes can terminate on any router, with no risk of policy violations.

Second, a router today can propagate only the route it has selected, even to other routers in the same AS. This limitation becomes a problem when the policy does not impose a *total ordering* of routes, i.e., a router's relative ranking of two routes can change depending on what other routes have been learned. So a route considered inferior, and thus not propagated, by one router might affect route selection at another router, which can lead an AS to violate its policy. We argue for decoupling route dissemination and route selection because they have rather different goals. Route selection is for the router *itself* to make correct decisions, and route dissemination needs to provide enough information for *other routers* to make correct decisions.

These two limitations of today's BGP protocol can lead to subtle interactions between different policy objectives in certain network topologies, which can result in violation of some objectives and even traffic blackholes. Techniques commonly used today for making route dissemination more scalable (e.g., route reflectors [5]) can make the matter worse, causing routing oscillations and forwarding loops. We propose to fix the problem by making a few small changes to today's BGP protocol and its implementation. The resulting protocol *atomic BGP* achieves desired policy flexibility at low cost. Atomic BGP eliminates all policy violations and their undesirable manifestations, and guarantees that policies are realized correctly. This also dramatically simplifies the task of configuring routers. The operators need only configure each router with the AS's policy, knowing that the routers collectively will realize the policy. This is in contrast to today's practice, where policies are specified and configured on a per-router basis. Not only the physical topology often has to be constrained in order to minimize the effect of policy violations, but changes to the network topology (such as failures) could unintentionally cause new violations.

The two proposed changes—letting a router assign different routes to different links and decoupling route dissemination from route selection—affect the control plane, where routes are propagated and selected. Both changes enable route assignments that ASes cannot have today. However, more flexible route assignments make it hard for hop-by-hop forwarding to deliver packets to the correct exit points, and tunneling is the most straightforward solution to ensure that packets are forwarded according to control-plane decisions.

The rest of the paper is organized as follows. Section 2 uses realistic examples to illustrate how ASes violate atomicity today and proposes solutions to make ASes atomic. Section 3 presents Atomic Routing Theory, which includes the mathematical model for atomic routing and conditions for achieving atomic routing in a distributed fashion. ART guides us to propose minimum changes to BGP so as to guarantee atomicity, and the resulting protocol—atomic BGP—is described in Section 4. Section 5 presents related work, and Section 6 concludes the paper.

2. POLICY VIOLATIONS IN TODAY'S BGP

In this section we examine route assignment and route dissemination in today's BGP, and identify the reasons why ASes violate atomicity. Since we are interested in defin-

Stage	BGP route selection step
i	1. Highest local_pref 2. Lowest AS path length 3. Lowest origin type
ii	4. Lowest MED (with same next-hop AS)
iii	5. Closest exit point 6. Lowest Router ID

Table 1: The six-step BGP route-selection process divided into three stages.

ing and analyzing "correctness" in steady state, we ignore transient behaviors and only consider the final route assignments. We draw principles which will motivate the atomic routing theory, and propose three types of solutions to eliminate policy violations: restricting the policy, restricting the topology, and enhancing the protocol.

2.1 Route Selection

In BGP routes are imported and exported at the edge routers. An edge router may perform *import actions* to modify or filter routes when they are first learned from other ASes, with the goal of influencing the route selection process. For each destination prefix, a router selects at most one route by comparing its learned routes in the *route selection* process. Then an edge router performs *export actions*, which may modify or filter the selected route based on the neighbor a link connects to and the route's own attribute. After import actions, route selection, and export actions, if a route is assigned to a link, it is exported to the corresponding neighbor. What routes are exported and where they are exported should comply with the AS's policy. Import actions are straightforward in BGP, and we next describe route selection and export actions in detail.

A router learns a set of routes from other routers either in the AS or in other ASes, and then proceeds through a sequence of six *route selection* steps that compare the routes based on their attributes (Table 1), ultimately selecting a single "best" route for each destination prefix [6]. In the first three steps, the router identifies the routes with the highest local preference, breaking ties based on AS-path length and then origin type. The fourth step is peculiar because the Multi-Exit Discriminator (MED) attribute is compared *only* among routes with the same next-hop AS. A neighboring AS uses the MED attribute to indicate its preference for where it wants to receive traffic (i.e., at the location(s) that announce routes with the smallest MED value). The MED-comparison step, which compares only subsets of routes, makes it impossible to define a total ordering on the routes, because the relative ranking of two routes may depend on the presence or absence of a third route. In the first four steps all routers in the AS would make the same decisions when presented with the same set of routes. Step five allows different routers to make different decisions, and each router will direct traffic to the "closest" exit point to implement *hot-potato* routing. If there is still a tie, the router uses router ID as the tie-breaker, so that a router selects at most one route. Step five and six will result in different choices of routes among routers.

We divide the six steps in the BGP selection process into three stages based on which routes are compared against each other and whether the comparison is router-specific: (i) AS-wide path pruning by comparing all routes, (ii) AS-wide route pruning by comparing the MED attribute among routes with the same next-hop AS, and (iii) router-specific tie-breaking (Table 1).

After selecting a single best route, an edge router must decide whether to export the route (perhaps after some modification) on each link by performing the *export actions*. Export actions often reflect the *business relationship* between a pair of ASes. Common business relationships include customer-provider and peer-peer [2]. A router typically exports any best route to its customers, but does not export routes learned from one peer or provider to another. A provider may also let its customers use the BGP community attribute [7] to have "remote control" over the handling of a route. The so called *redistribution communities* affect a router's export actions. Examples include "no-export" and "prepend," which instruct the provider not to export the route to other ASes, or to export it after prepending the AS-path with additional hops to make the AS path artificially longer. Other than these common export actions, commercial routers allow network operators to filter and modify routes based on regular expressions on the AS-path attribute as well (e.g., an AS can filter or de-preference routes that pass through a particular AS, possibly because the two ASes have a bad relationship).

Peering contracts often require the two networks export *consistently*, i.e., exporting routes with the same AS path lengths—to each other at all peering points so as not to influence the routing decisions of the other network [8, 9, 10, 11, 12, 13]. In addition, peering contracts often require the peers to export "full customer routes," so that they can reach each other's customers. Both of these requirements can be violated due to the interaction between route selection and export actions.

Consistent export is frequently violated in practice [14, 15], and Figure 1 shows an example where two routes considered equal are treated differently by export actions. The AS learns two routes r_1 and r_2 that are equally good in the first four steps of the route selection process. Due to hot-potato routing, router A_1 selects r_1 and router A_2 selects r_2. Suppose r_1 triggers no export actions but r_2 does. This is possible in two cases: i) if r_2 has a community attribute specifying "no export" or "prepending," and 2) if r_1 is learned from a customer, r_2 is learned from a peer, but the AS equally prefers customer- and peer-learned routes during route selection. In either case, the AS exports inconsistently on the two links to the same neighbor. When the filtering is due to business relationship, if routers A_1 and A_2 both select route r_2, e.g., because r_2 is closer to both of them than

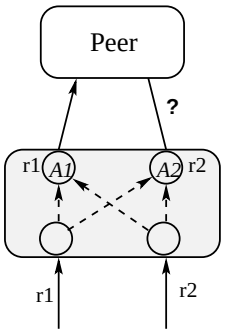

Figure 1: Two routes considered equal during route selection are treated differently by export action.

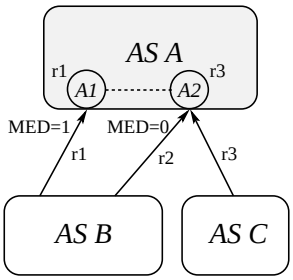

Figure 2: Violation of the MED semantics. Router A_2 prefers r_3 so router A_1 never learns route r_2, which has the lowest MED value. Router A_1 selects r_1, violating MED semantics.

r_1 is, then both routes will be filtered. So the peer cannot reach the destination through the AS even though there is a customer-learned route, and the AS has violated the "export full customer routes" requirement.

Lesson learned: Routes that are considered equal during selection should not be treated differently during export. In addition, if an AS would like to assign different routes to links connecting to different neighbors, the distinction should be made during route selection rather than during export, i.e., the AS should have different route selection processes for different types of neighbors (customers vs. peers and providers).

The following are solutions for solving these problems.

Restricting the policy: The AS needs to strictly prefer customer-learned routes over peer-learned routes, so that customer routes will always be exported to peers. There cannot be export actions based on communities, so communities have to be either ignored or handled at import.

Restricting the topology: The AS has to ensure that a router connects to only one type of neighbors. This way each router can select routes for the particular neighbor type, and routers connected to peers can be configured to select only customer-learned routers. To configure today's BGP so that each router selects routes differently is not straightforward, and is omitted from this paper. But restricting the topology alone cannot solve the problems caused by communities, so they still need to be either ignored or handled at import.

Enhancing the protocol: Two changes are needed: First, a router should select routes for each neighbor type independently, so a router may need to run several route selection processes in parallel, and route selection is per link instead of per router. Second, route modification and filtering should be done *before* route selection, since now that routes are selected for a particular customer, there is no need for export actions.

2.2 Route Dissemination

Upon selecting a best route, a router informs other routers in the AS via internal BGP (iBGP) sessions. The iBGP sessions in the AS together form the *signaling graph*. Throughout this paper, we assume that the signaling graph connects all the routers, so the AS is not partitioned.

If the routes have total ordering, disseminating only the selected route can ensure that routers select routes according to the AS's policy. But the MED attribute destroys the total ordering of routes because it is compared only among routes with the same next-hop AS. This can cause many problems [16, 17] and we show one example in Figure 2, where the semantics of MED are violated. All three routes are equal in the stage i comparisons (Table 1), but r_2 has a lower MED value than r_1. Between routes r_2 and r_3, assume router A_2 picks r_3 in the final router ID tie-breaking step. So router A_1 learns r_1 and r_3, and picks r_1 due to hot-potato routing. So a route with a higher MED value (r_1) is selected when the AS learns another route with a lower MED value (r_2), violating the MED semantics. The problem becomes worse when route reflectors [5] are used, forming a hierarchy among routers inside an AS, because routers will learn even fewer routes. In fact, there can be routing oscillations in this case.

Lesson learned: If routes do not have total ordering, each router disseminating only the selected route is not sufficient.

The following are solutions for solving these problems.

Restricting the policy: Disallowing the use of the MED attribute will restore the total ordering of routes and solve all problems, though this may cause a network to lose customers who demand MED.

Restricting the topology: A router which terminates a link from a neighbor who uses MED should not terminate any links from other neighbors, and full-mesh iBGP, where every router has an iBGP session with every edge router, should be used. This ensures that a route which is among the best routes after the MED comparison step be propagated to all the routers.

Enhancing the protocol: To ensure that attributes which are not compared among all routes (like MED) are treated correctly in all topologies, each router needs to be able to disseminate more than one route. How to select routes to disseminate will become clear after we present ART.

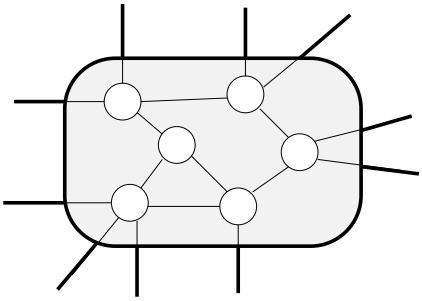

Figure 3: An autonomous system. The nodes represent routers and the lines represent links. The dark circle is the AS boundary.

3. ATOMIC ROUTING THEORY

This section presents the atomic routing theory (ART). Given a set of routes, an AS solves the *route assignment problem* (RAP) to assign a subset of routes to each edge link, according to a *policy* that constrains the solution. We give mathematical representations of RAP and policy, and derive the conditions for distributed atomic routing.

3.1 The Route Assignment Problem

Suppose the AS has N links connecting to other ASes (Figure 3). For a destination, the AS learns a set of *routes* R, each specifying a path to forward the packets. We do not constrain how the routes are learned. For example, routes can be learned by the edge routers or the AS's routing server, from neighboring ASes's border routers as in today's BGP, from a neighbor's routing server, or even from an Internet-wide centralized routing server. A route r has an *exit point*, which is an edge link of the AS, and potentially other information. For convenience we define the *empty route* ϵ, which does not have information for reaching the destination, and make it a convention that $\epsilon \in R$. Thus, if no routes are learned, we write $R = \{\epsilon\}$. A packet following a route will exit the AS at the corresponding exit point, and if the route is empty the packet is dropped.

We can generalize the grouping packets by destination to the notion of a *traffic unit*—a group of packets which the AS does not differentiate when making forwarding decisions. In destination-based routing like today's Internet, packets belonging to a traffic unit would have the same destination or destination prefix, and request the same type of service within the AS. Other network architectures such as source routing may define a traffic unit differently. Thus in destination-based routing, if the source would like different packets for the same destination to receive different types of service, it would send the packets as different traffic units. But if multiple routes are assigned to an edge link for a traffic unit, the router receiving a packet on the edge link for the traffic unit can arbitrarily decide which of the assigned routes to forward the packet on. Without loss of generality, we focus on policy and routing for a single traffic unit.[1]

The *route assignment problem* (RAP) is to assign a *set* of routes $e_l \subset R$ to edge link l, for $l = 1, 2, \ldots, N$,[2] and the assignment is advertised to the corresponding neighbors, either via border routers or routing servers. If e_l is empty then no route is assigned to the link and we write $e_l = \{\epsilon\}$. Let $E = \{(l, e_l), l = 1, 2, \ldots, N\}$ represent the *route assignment solution* of the AS. Our route assignment model is more flexible and more general than today's BGP. It is more flexible because routes are assigned to each *edge link*, and multiple links on a router are allowed to have independent assignments; it is more general because a single edge link can be assigned multiple routes, which can be used for, e.g., multi-path routing.

3.2 AS-Wide Routing Policy

An AS's policy is rooted in the AS's business goals, such as using more profitable routes, providing reliable service to customers, conserving its own resources by traffic engineering, adhering to contracts with other ASes, etc. These business goals can translate into two types of *objectives*: preferences and constraints on the route assignment solution. A *preference* objective *ranks* different route assignment solutions, e.g., "using more profitable routes" can mean assigning routes whose next hop is a customer whenever possible, because a customer typically pays to receive traffic, and "conserving resource" can mean routers use exit points as close to them as possible. A *constraint* objective *disallows* certain route assignments. It can be a route filtering requirement either for all or some of the edge links, e.g., "avoiding known insecure routes" means such routes cannot be assigned to any edge link. A constraint can also examine multiple links together and require that the routes assigned to them have certain joint properties, such as routes assigned to links connected to the same peer having the same AS path length, common in today's networks. In the latter case, we say that the route assignments of the links are *dependent*. A policy is a combination of these preferences and constraints, and a way to weigh them relative to each other.

These objectives and their relative weighing are distilled into the AS-wide policy function $P(\cdot)$, which takes in the set of candidate routes R, and returns all the route assignment solutions that satisfy the policy, i.e.,

$$\mathcal{E}_P = P(R),$$

where $\mathcal{E}_P$ is the set of route assignment solutions satisfying the policy $P(\cdot)$. The set $\mathcal{E}_P$ should not include any route assignment that is disallowed by a constraint, and should only include the most preferred route assignments among the rest. An AS *realizes* its policy, or is *atomic*, if it chooses a route

[1] Policies that involve multiple traffic units can be distilled into a policy for each traffic unit.

[2] In general, we allow multiple routes to be assigned to a link, but can easily add the restriction that $|e_l| \leq 1$.

assignment that satisfies its policy, i.e.,

$$E \in P(R); \tag{1}$$

otherwise, it *violates* its policy. We let the policy function $P(\cdot)$ to return a set of route assignment solutions instead of one particular solution, because it is often the case that an AS equally prefers multiple route assignment solutions, and we should leverage this flexibility for efficient implementation, especially in a distributed setting. An AS need not calculate the whole set $\mathcal{E}_P$; it only needs to find *one* route assignment that satisfies the policy. For an AS to be guaranteed atomic, the AS-wide policy should not depend on how links and routers are connected inside the AS, and the AS should realize its policy regardless of its topology. But the particular route assignment that an AS picks may depend on its topology.

When the AS assigns routes to a link, other links that are tied to this link, directly or indirectly through some constraints in the policy, have to be considered at the same time. For example, if link 1 and 2 have a constraint, and link 2 and 3 have another constraint, all three links need to be considered together. Thus we divide the AS's edge links into non-overlapping *groups*, so that links across groups are independent. Suppose the N edge links are divided into M groups and let G_i represent group i, i.e., $\bigcup_{i=1}^{M} G_i = \{1, 2, \ldots, N\}$ and $G_i \bigcap G_j = \phi$ for $i \neq j$. Now the route assignment solution, which includes the route assignment for each link, can be decomposed according to the link groups. Let the assignment for group i be $E_i = \{(l, e_l), l \in G_i\}$, and we have $E = \bigcup_{i=1}^{M} E_i$. The AS's policy, which includes the objectives involving all the edge links, can also be decomposed into a set of policies, one for each link group. Let $P_i(\cdot)$ be the policy for link group i, then the requirement that an AS realizes its policy, $E \in P(R)$, becomes

$$E_i \in P_i(R), \ \ \forall i. \tag{2}$$

Note that the division of links into groups allows independent links to be in the same group, but sometimes we may want to have link groups as small as possible, so it is clear where coordination is and is not needed. Having many link groups is not a problem, because in practice the calculation for $P_i(\cdot)$ can usually be amortized.

The devision of edge links into independent groups illustrates the importance of decoupling the multiple links of a router. The policy may dictate that different links of a router belong to different groups and should therefore be independent. But today's BGP routers artificially places a constraint on the route assignment so that the links on the same router are dependent. When these artificial constraints conflict with the policy constraints, there may be no route assignment that satisfies all constraints, causing policy violation.

3.3 Partial Ordering of Routes

Having some ordering of routes is convenient because then only the highest ranking routes need to be considered for assignment and dissemination. Routes may not have a total order, e.g., when the MED attribute is used in BGP, but we can define a *partial order*. Once the partial order is defined, the relationship between two routes is a property of the two routes only, and does not depend on the presence and absence of other routes. For two routes a and b, we have $a \geq b$, $b \geq a$, or a and b not comparable. The most trivial partial order is that all routes are not comparable, thus we can always define a partial order for the routes.

We represent the partial order with the *route preference function* $B_i(\cdot)$, which, given any set of routes S, returns the most preferred routes $B_i(S) \subset S$. If an AS learns a set of routes R, only routes in $B_i(R)$ can be assigned to any link, thus $B_i(\cdot)$ effectively *prunes* routes which do not need to be considered in route assignment. The rest of the policy is expressed by the *combinatorial preference function* $C_i(\cdot)$, which, given the set of routes $B_i(R)$, returns the set of route assignments that the AS prefers. Thus $C_i(\cdot)$ can represent preferences involving combinations of routes and links. We have $C_i(B_i(R)) = P_i(R)$, and the AS is atomic if

$$E_i \in C_i(B_i(R)), \ \ \forall i. \tag{3}$$

If $B_i(\cdot)$ is trivial, i.e., it returns all routes given to it, then we have $C_i(\cdot) = P_i(\cdot)$.

Since $B_i(\cdot)$ defines a partial order, it has the following property: if for a route $a \in S$ we have $a \notin B_i(S)$, then we *cannot* have $a \in B_i(S \bigcup S')$ for any set of routes S'. Another way to state this property is, if $T \subset S$, then $B_i(T) \supset B_i(S) \bigcap T$. We call the partial order of routes defined by $B_i(\cdot)$ the *dominance* relationship. For two routes a and b, we say a *strictly dominates* b for link group i, or $a >_i b$, if $B_i(\{a, b\}) = \{a\}$; we say a *dominates* b, or $a \geq_i b$, if for any route c such that $b >_i c$, we also have $a >_i c$. If we have $a \geq_i b$ but not $a >_i b$, then a *weakly dominates* b. Note that a route weakly dominates itself. If we have $a \geq_i b$ and $b \geq_i a$, then a and b are *equal*, and we write $a =_i b$. Dominance is transitive in that if $a \geq_i b$ and $b \geq_i c$ then $a \geq c$. When two routes are not comparable, neither route dominates the other one. But if all routes are comparable, then dominance is a *total order*. We use the convention that the empty route is always comparable to another route, and a route strictly dominated by the empty route is not in $B_i(R)$.

For two routes a and b, if $a >_i b$, then $b \notin B_i(S)$ for any set of routes S containing a and b. Thus, a route strictly dominated by another route will not be assigned to a link or be able to influence the route selection by any router (as long as the dominant route is present), and can therefore be safely discarded. For any route $r \in R \backslash B_i(R)$, there must exist a route $s \in B_i(R)$ such that $s >_i r$. For two routes a and b, if $a \geq_i b$, we can have $B_i(\{a, b\}) = \{a, b\}$, i.e., both a and b are candidates for route assignment. But knowing a is enough to prune all the routes that b would help prune, so even if b is assigned to a link, it does not needed to be disseminated. Thus, although only routes in $B_i(R)$ are considered for assignment and dissemination, *the decisions*

to assign and disseminate routes should be made independently.

The route pruning by $B_i(\cdot)$ is done based on comparing individual routes, and does not require knowledge of all the learned routes, so whether it is performed centrally or distributedly, it is a simple operation requiring no coordination. The complexity of implementing the policy lies in the combinatorial preference function $C_i(\cdot)$. This is especially true in a distributed setting, because not only the computation, but also the communication overhead depend on $C_i(\cdot)$. Thus when designing a protocol or specifying a policy, there is incentive to factor as much of the policy into $B_i(\cdot)$ as possible, and there is a tradeoff between the complexity of $C_i(\cdot)$ and the ease of implementation.

3.4 Distributed Atomic Routing

If the AS solves the route assignment problem centrally, a routing server would learn all the routes, and then for each link group, filter routes according to $B_i(\cdot)$ and select a route assignment that satisfies $C_i(\cdot)$. The communication required is only for the central server to learn all the routes and to let the routers and neighbor ASes know the route assignment it picks. The computation to find a route assignment that satisfy the AS's policy only needs to be done once by the server. So a centralized solution is perhaps the most efficient in terms of communication overhead and total computation required.

On the other hand, finding a route assignment that satisfies the policy in a *distributed* fashion is non-trivial. In fact, BGP cannot always achieve this today. In distributed routing, typically each router learns a subset of the routes in R, and assigns routes to the edge links connected to it. Thus the communication among routers should provide each router sufficient information to make correct decisions, and the minimum amount of communication required depends on the policy. In order to reduce communication overhead, the route preference function $B_i(\cdot)$ should prune as many routes as possible, because pruned routes do not need to be disseminated.

Among the routes that are not pruned, the combinatorial preference function $C_i(\cdot)$ determines which ones need to be disseminated. We can easily imagine policies where a router has to know *all* the routes in $B_i(R)$ in order to make correct decisions. For example, if a combinatorial preference function calculates a score for each of the possible route assignments, sorts them, and picks the one in the middle, then it is likely that every router needs to know all the routes in $B_i(R)$ in order to pick the correct route assignment. If there are many routes in $B_i(R)$, the communication overhead can be too high to make a distributed protocol attractive. In the other extreme case, we can imagine a trivial combinatorial preference function $C_i(\cdot)$ which prefers all route assignments consisting of the routes in $B_i(R)$. In this case the dissemination only needs to ensure that routes not in $B_i(R)$ are pruned and not assigned to any link, so we say the combinatorial preference function $C_i(\cdot)$ is *inactive*. When $C_i(\cdot)$ is inactive, the condition for an AS to be atomic reduces to

$$e_l \subset B_i(R),\ \forall l \in G_i,\ \forall i. \tag{4}$$

Given an arbitrary $C_i(\cdot)$, the dissemination requirement is somewhere between the two extremes.

Suppose $C_i(\cdot)$ is inactive. The dissemination only needs to ensure that any route *not* in $B_i(R)$ is pruned and *not* assigned to any edge link. We have the following theorem:

THEOREM 1. **(Distributed Atomic Routing)** *In distributed routing, for an edge link group, if the combinatorial preference function is inactive, then in order to realize the AS's policy for the link group, each router must ensure that for every route it has learned, there is a route it disseminates that dominates the route.*

PROOF. By contradiction. Let D be the set of routes disseminated by router 1. Suppose route $a \notin D$ known to router 1 is not dominated by any route in D, i.e., there exists a route b (not necessarily known to router 1) that is strictly dominated by a but not strictly dominated by any route in D. It is possible that another router, router 2, learns only route b and the routes in D. So router 2 does not know any route that dominates b, and may assign it to an edge link. But b is strictly dominated by a and therefore $b \notin B_i(R)$, so assigning b to a link violates the AS's policy. □

Thus a router must disseminate a route if the route is not dominated by any route known to the router. On the other hand, a route does not need to be disseminated if it is dominated by a disseminated route. In order to minimize the dissemination overhead, a router should find the smallest set of routes that dominate all routes. Since a route can only dominate other routes comparable to it, the minimum number of routes disseminated is equal to the minimum number of disjoint subsets we can divide $B_i(R)$ into so that routes within a subset are comparable. If all routes are comparable, then disseminating one best route is sufficient (and necessary because otherwise no route is disseminated).

Suppose $C_i(\cdot)$ is inactive, then Theorem 1 gives the dissemination required to achieve policy correctness. If multiple link groups share the same route preference function $B_i(\cdot)$, then route pruning and route dissemination only need to be done once for these link groups. So the edge links can be divided into a few *types* according to their $B_i(\cdot)$ function, and each type may need separate route pruning, selection, and dissemination processes. Typically, there are only two types of edge links in today's ASes: those connected to a customer, and those connected to a peer or a provider.

Theorem 1 gives the condition for achieving atomicity in a distributed fashion when the combinatorial preference function $C_i(\cdot)$ is inactive. It is also the condition for correctly pruning routes according to the route preference function $B_i(\cdot)$. If the combinatorial preference function $C_i(\cdot)$ is *active*, then possibly more routes need to be disseminated. Therefore there is a tradeoff between policy flexibility and

dissemination overhead. Policies that are too complex may require a lot of coordination among routers and too high a dissemination overhead to make distributed routing practical. For such policies, a centralized solution where a server collects all routes and makes route assignment decisions for the entire AS is more sensible. A centralized solution is much more flexible than a distributed one [18]. As the ASes' objectives become more subtle and more complex, centralized application of routing policy may someday become necessary.

3.5 Router-Specific Preference

The pruning of routes according to $B_i(\cdot)$ is common for all routers in the AS, i.e., all routers will arrive at the same set $B_i(S)$ given the same routes S. When $C_i(\cdot)$ is inactive, a router can assign any routes in $B_i(R)$ to a link in group i. Thus a router can implement its preferences such as choosing routes with the closest exit points (hot-potato routing). Let $H_l(\cdot)$ be the *router-specific preference* function for link l, which takes the set of best routes $B_i(R)$ ($l \in G_i$) and returns the routes most preferred by the AS to assign to link l, i.e.,

$$e_l = H_l(B_i(R)), \;\; l \in G_i, \;\; \forall l. \tag{5}$$

If we want to implement hot-potato routing, then $H_l(\cdot)$ will return the route with the closest exit point, or a set of routes with exit points close to the router.

The purpose of hot-potato routing is to conserve internal resource, but today it does not take into account congestion, etc., and can cause problems like unintended traffic shifts due to changes in the network. So if $H_l(\cdot)$ returns multiple routes with exit points close to the link, then using any of the routes or load-balancing among them is likely sufficient to achieve the goal of conserving resource, and at the same time provides flexibility to, say, avoid congestion.

In distributed atomic routing, the signaling graph, and how closely it resembles the physical network topology, determines how closely the AS can realize hot-potato routing. If the signaling graph overlaps the physical topology then we can implement exact hot-potato routing. This is possible if each router always disseminates the route it chooses to use (which has the closest exit point). Since there are usually choices among equal routes as to which one to disseminate, this additional requirement is unlikely to increase the number of routes disseminated. If the signaling graph only loosely resembles the physical topology, then doing so will only implement *approximate* hot-potato routing, i.e., every router may use a *nearby* (but not necessarily the closest) exit point. But as discussed above, approximate hot-potato routing is likely to be sufficient, allowing some flexibility in building the signaling graph.

4. ATOMIC BGP

Guided by ART, we identify limitations of today's interdomain routing protocol BGP, and suggest modifications to make BGP-speaking ASes atomic regardless of the topology. We propose changes in route selection and route dissemination that can be implemented using existing and emerging technologies like virtual routing and forwarding (VRF) and the BGP ADD-PATH extension [19]. The resulting protocol—atomic BGP—not only guarantees atomicity, but also simplifies network management.

We first model today's BGP with ART. The set R includes all the routes that an AS learns for a destination prefix, plus the empty route. An AS performs *import actions* to filter and manipulate routes when they are first learned. Import filtering typically eliminates routes that the AS will not assign to any link even if no other routes are learned, such as routes that are known to be insecure, or routes that would cause loops. Import route manipulation modifies some route attributes to influence the route selection process, e.g., setting local_pref, ignoring MED (by resetting them to zero) for neighbors who are not supposed to use it. The import actions and the first four steps of the route selection process (Table 1) are captured by the AS-wide partial ordering $B_i(\cdot)$. The fifth step in route selection (hot-potato routing) can be represented by the local preference function $H_l(\cdot)$.

In today's policies, the combinatorial preference function $C_i(\cdot)$ typically requires that routes exported to a particular neighbor have the same AS-path length. AS-path length is compared in $B_i(\cdot)$, so $C_i(\cdot)$ should be inactive. But a router may also modify the attributes of a route before exporting it to a neighbor. We can capture export actions in $C_i(\cdot)$, i.e., $C_i(\cdot)$ can check the consistency among routes *before or after* export actions. If $C_i(\cdot)$ checks routes after export actions, then $B_i(\cdot)$ should rank routes *after* the export actions in order for $C_i(\cdot)$ to be inactive. We say two routes x and y are *consistent* if $B_i(\{x, y\}) = \{x, y\}$, because they automatically satisfy $C_i(\cdot)$.

4.1 Route Selection

We first describe changes to route selection, assuming that each router has learned enough routes to make correct decisions, deferring the discussion of route dissemination to the next subsection. From the discussion in Section 2.1, export actions are the main cause of policy violations in today's route selection, because they are applied *after* routes are selected, based on properties not considered during route selection. There are two types of export actions: route filtering and attribute manipulation.

Export filtering is performed today because a router selects a single best route, but may not want to export the route to all neighbors. For example, an AS would export a route learned from one customer to another customer, but not from one peer or provider to another peer or provider. For the neighbors that the AS cannot export the selected route, the policy may dictate that a different route be exported (e.g., when a router selects a peer or customer route, and does not export it to a peer or provider, the policy may dictate that a customer route be exported to a peer or provider). This is

not possible in today's networks because a router selects a single route, hence the policy violations. Export filtering allows an AS to export differently to different neighbors, but it provides very limited flexibility, because an edge links can either export the route selected by the router it is attached to, or export no route. In ART, *we allow routes to be selected for each edge link independently*, whether the links connect to the same router or not. This allows the AS to have a separate preference function $B_i(\cdot)$ for each edge link, and eliminates the need for export filtering, because a route that cannot be exported is not in $B_i(R)$, and is therefore not selected.

Attribute manipulation is a tool for ASes to influence how *other ASes* use a route. For example, the AS can make a route appear longer to other ASes (by "prepending" the AS path with repeated AS hops) so that less traffic uses this route. However, if attribute manipulation is based on information not considered during route selection, then it is possible that only some of the selected routes have their attributed manipulated, and the selected routes can become inconsistent. Thus *for neighbors that expect consistency, attribute manipulation should be done before selection*. This will result in different route assignments from today, but is necessary for atomicity.

Allowing each link group to have its own route selection process may sound expensive, but in practice, a router needs to implement at most two selection processes in order to realize today's common policies. We can classify the edge links of an AS based on the type of neighboring AS a link connects to: customers, peers, and providers. Routers today are typically configured not to export a route learned from a peer or provider to another peer or provider. Routes with the "no export" community are filtered, and routes with the "prepend" community are modified, before being exported. In addition, the routes exported to a peer are typically expected to have the same AS-path length. Thus a router should consider all routes when selecting for a customer, but should consider only customer-learned routes when selecting for a peer or provider. A router should also apply attribute manipulations first when selecting for a peer. Thus, an AS needs to have a route selection process for neighbors, and another one for peers and providers, as it can export to providers the same routes it exports to peers.

In order to support Virtual Private Networks (VPNs), today's commercial routers commonly implement *virtual routing and forwarding* (VRF) [20, 21], which allows a router to run multiple decision processes. Each packet interface, the smallest unit of a router's data-plane, can subscribe to any of the decision processes in the router, and be populated with the corresponding forwarding table. We can utilize this existing feature to implement atomic BGP. For memory considerations, an interface may support only one forwarding table, in which case neighbors requiring different forwarding tables may not be able connect to the same interface, i.e., a customer cannot connect to the same interface as a peer or provider. But different kinds of neighbors can always connect to the same *router*, and we expect this memory limitation to go away with the advance in memory technologies.

4.2 Route Dissemination

A router will only select routes that are not strictly dominated by any route in R, i.e., the routes in $B_i(R)$. Even though a router may not learn all the routes, route dissemination should ensure that every router learns enough routes to make a correct decision. Theorem 1 states that the routes disseminated should (strictly or weakly) dominate all routes. In order to minimize the number of disseminated routes, $B_i(R)$ can be divided into the minimum number of disjoint subsets such that routes within each subset are comparable. Then disseminating one best route from each subset is sufficient to realize $B_i(\cdot)$ (ties can be arbitrarily broken, e.g., according to hot-potato routing).

Today's BGP implements $B_i(\cdot)$ via a series of attribute comparisons. The first three steps compare all routes, and they form a total order of routes. The fourth step compares the MED attribute only among routes with the same next-hop AS, thus it destroys the total order, but the four steps still form a partial order. If we were to add any more step in the route selection process, a step that compares all routes must be added *before* the MED step, as adding it after will destroy the partial order. In other words, the set of routes that are compared against each other cannot expand from one step to the next. The final step of comparison in the partial order determines how routes can be divided into subsets so that routes are comparable within each subset. Routes with different next-hop ASes that use MED are not comparable, but a route with a next-hop neighbor which does not use MED is comparable to all routes.

Among the routes in $B_i(R)$, the total number of next-hop ASes that use MED determines the minimum number of routes disseminated. Applying Theorem 1, we have the following rule for dissemination. For each neighbor type, among the routes that are left after MED comparison, a router needs to *disseminate at least one route, and at least one route for each next-hop AS that uses MED*. Thus every router should be configured with the list of all neighboring ASes that use MED, and disseminate routes according to the above rule for *each* neighbor type, regardless of its connections. But for each next-hop AS that uses MED, one best route dominates all the routes with the same next-hop, for all neighbor types. Thus the number of routes that need to be disseminated is upper-bounded by the number of neighbor ASes that use MED.

The changes in route selection described in the previous subsection do not affect hot-potato routing, i.e., a router can break ties among the best routes known to it by choosing the one with the closest exit point. Thus, dissemination needs to ensure that every router learns the route in $B_i(R)$ with the closest exit point to it. This can be achieved by having the signaling graph matches the AS's physical topology, and a router disseminates the routes it selects. Thus a router may

need to disseminate more routes than those required for correctly implementing the partial order $B_i(\cdot)$. But if *approximate* hot-potato is acceptable, i.e., a router can select any nearby route, then the signaling graph only needs to loosely resemble the physical topology. The precise tradeoff is left for future study.

Atomic BGP requires two changes to the iBGP protocol. First, a router today can advertise at most one route to another router, but atomic BGP requires that a router disseminate multiple routes when necessary. The recently proposed ADD-PATH feature of BGP [19], which allows multiples routes advertised per iBGP session, can solve this problem. Second, in today's iBGP, routers (that are not route reflectors) only advertise externally-learned routes, requiring the signaling graph to be richer than necessary. Atomic BGP allows routers to disseminate any route, and therefore only require the signaling graph to be connected (hot-potato routing may impose further constraints). We need to ensure that routes are not propagated in loops, and techniques similar to BGP Scalable Transport (BST) [22] and internal BGP downloader (iBGPd) [23] can achieve this.

A concern for disseminating more routes is the increase in communication overhead and storage (because a router stores all learned routes). Since multiple routes need to be disseminated *only* when there are routes in $\bigcup_i B_i(R)$ with multiple next-hop ASes that use MED, the increase in overhead is mainly a problem for large ISPs that learn many routes *and* have many neighbors who use MED.

An earlier study on a large ISP gives a loose upper bound on the increase of dissemination overhead. The study took the BGP table dumps from the edge routers of the AT&T backbone, and analyzed the best routes after the first three steps of the selection process (Table 1). It showed that the AT&T backbone learns best routes from a single next-hop AS for about 50% of the prefixes, and from two next-hop ASes for about 20% of the prefixes [24]. Based on their data, we estimate that the AT&T backbone learns best routes from an average of two next-hop ASes for each prefix. If all of these next-hop ASes use MED, then about twice the number of routes need to be disseminated in atomic BGP compared to today's BGP. If not all the next-hop ASes use MED, then fewer routes need to be disseminated.

4.3 Data-Plane Tunneling

A packet that entered the AS should follow one of the routes assigned to the edge link where it enters. However, if the packet is forwarded via multiple hops within the AS, the routers along the way may forward the packet along a different route. This is called *packet deflection*, and is hard to avoid in hop-by-hop forwarding [25].

Atomic BGP allows more flexibility in route assignment so that all today's common policies can be realized correctly, but this makes packet forwarding even more complex. Thus the best solution to guarantee that packets are forwarded correctly is to use *tunneling*. Router-to-router tunneling was in fact one of the solutions proposed in the early stages of adopting BGP [26], but routers did not support tunneling efficiently until the MPLS [27] technology matured.

In atomic BGP, multiple edge links on the same router are independent of each other. Packets entering the AS at the same router but via different links may follow different routes to exit the AS, and packets exiting at the same router may need to exit on different links. So, tunneling from *edge link to edge link* is needed. Fortunately, IP-in-IP tunneling is readily available at line rate in many commercial routers. A packet can be encapsulated at ingress link and decapsulated at egress link before being forwarded to neighboring ASes. The path taken by the packet inside the AS can be determined by the underlying interior gateway protocol, such as OSPF, therefore no signaling overhead is needed to set up the path explicitly.

5. RELATED WORK

Most work on interdomain routing has focused on protocol convergence, giving conditions for inter-AS stability and to avoid intra-AS routing anomalies [28, 2, 29, 15, 25, 16]. Although we draw examples from previous work, our focus is not on what BGP does today, but on what an intra-AS routing protocol *should* do—guarantee that an AS correctly realizes its policy regardless of the topology—rather than what it does today. We propose the atomic routing theory for policy-based routing. Applying it to today's BGP-speaking networks, we suggest practical enhancements to make ASes *intrinsically atomic*.

The most related work to ours is by Basu et. al. [17]. Interestingly, even though they started from modeling the iBGP protocol, and we started from examining the problem intra-AS routing should be solving, our solutions are similar: they propose to disseminate all routes in $B_i(R)$, while we point out that only a subset of $B_i(R)$ need to be disseminated, as long as the disseminated routes dominate all other routes. In addition to new route dissemination schemes, we propose dividing the edge links into groups and allowing each group to have its own route selection processes. Similar to ART, Morpheus [18] also proposes to assign different routes to different edge links and use link-to-link tunnels, but the focus is on policy flexibility with a centralized routing server, rather than policy correctness in a distributed setting.

Metarouting [30, 31] shares the view with ART that the routers in an AS should select routes so that they collectively provide some policy guarantee. The two pieces of work are complementary as metarouting provides a formal way to specify *policies* that can be decomposed into routing algebras, and gives sufficient conditions on policies to ensure convergence, while our work allows policy to be general, and provides the *mechanism* for an AS to realize its policy.

A practical motivation of our work is to specify an AS's policy at a high level and to automatically configure routers according to the policy. This vision is shared by previous

work. The Routing Policy Specification Language (RPSL) [32] was invented to specify policy so as to automatically generate router configurations, but the language is vague and hard to use. Boehm et. al. [33] created a system that translates a given policy objective (which they call an atomic unit), described in a high-level language, into vendor-specific router configurations. As we have shown, due to the interactions between policy objectives and router limitations, configuring all routers correctly does *not* ensure that the routers *collectively* realize the AS's policy. But, using their system in a network running atomic BGP will produce automated configurations that guarantee to correctly realize an AS's policy—an ideal situation for operators.

Our work is motivated in part by BGP protocol extensions, e.g., [34], and proposals for other interdomain routing architectures, e.g., [35, 36, 37], which assume that an AS routes like a single router. This simplification begs the question of what would happen if different routers in an AS select different routes. There has been heuristic techniques to model an AS (to a limited precision) by a few routers derived from observation [38], but a precise AS model is needed for theoretical analysis. ART models an AS as a single node, and guarantees that it acts like one, so future work on modeling and analyzing interdomain routing, and designing new protocols can utilize ART to precisely model an AS.

6. CONCLUSION AND FUTURE WORK

Despite the importance of policy-based interdomain routing, an AS is not guaranteed to realize its policy today. In this paper, we present the atomic routing theory, which precisely defines what it means for an AS to realize its policy, and gives conditions on a distributed protocol to achieve atomicity. We also propose a practical protocol atomic BGP, with two simple enhancements from today's protocol to make ASes atomic—having multiple route-selection processes in a router and disseminating multiple routes in an iBGP session. Atomic BGP is incrementally deployable and the changes are easily implementable in a single AS using existing and emerging technologies. Once deployed, atomic BGP can dramatically simplify network management, as it leads naturally to automatic router configuration and imposes no constraints on the network topology.

This paper deals with how to realize a policy $P(\cdot)$ once it is synthesized from a set of objectives, while how to synthesize $P(\cdot)$ is left for future work. But we believe that the process should check for fundamental conflicts among the objectives, i.e., whether all the policy objectives can be satisfied at the same time. Also, in order to guarantee Internet-wide stability, the policy of an AS should satisfy some conditions similar to the way today's policies do [2]. Since an AS's policy is precisely represented by the function $P(\cdot)$, and since ART and atomic BGP ensure that an AS realizes its policy correctly, looking for policy conflicts and deriving stability conditions can be more systematic and free of intra-AS details. Thus, in addition to making today's ASes atomic, the ART model can facilitate the design of interdomain routing protocols that provide atomicity guarantee and global stability.

Atomic routing theory is not confined to interdomain routing. It can be used whenever a routing hierarchy exists and a collection of nodes need to appear as a "single node." Examples are:

- Cluster-based routers. The capacity of single-chassis routers is constrained by space and power, but one can build routers with high capacity by interconnecting multiple smaller routers. The many small routers together need to select and propagate routes and forward traffic the way a single large router would do.
- Multi-AS networks. An institution may have multiple ASes for historical or scalability reasons; e.g., a large ISP may divide its global network into ASes, one for each continent. Even though each AS implements its own policy, the institution may want its whole network to have some common policy.

Atomic routing theory builds theoretical foundations for policy-based interdomain routing, from which we can derive the conditions for atomicity. The simple enhancements to BGP to make ASes atomic show the power of ART—it would be very hard to find these enhancements through trial-and-error, but theory helps find the simplest and most effective solutions in a systematic way.

Acknowledgments: We sincerely thank Gordon Wilfong for his invaluable feedback on earlier drafts.

7. REFERENCES

[1] J. Hawkinson and T. Bates, "Guidelines for creation, selection, and registration of an Autonomous System (AS)." RFC 1930, March 1996.

[2] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 681–692, 2001.

[3] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," *IEEE Network*, vol. 19, pp. 5–11, Nov.-Dec. 2005.

[4] R. Zhang-Shen, "Making an AS Route like a Single Node," *NANOG 43 Presentation*, June 2008.

[5] T. Bates, R. Chandra, and E. Chen, "BGP Route Reflection - An Alternative to Full Mesh Internal BGP (IBGP)." RFC 4456, April 2006.

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

[7] R. Chandra, P. Traina, and T. Li, "BGP Communities Attribute." RFC 1997, August 1996.

[8] "AOL Transit Data Network Settlement-Free Interconnection Policy." `http://www.atdn.net/settlement_free_int.shtml`.

[9] "AT&T Global IP Network Settlement-Free Peering Policy." `http://www.corp.att.com/peering/`.

[10] "Verizon Business Policy for Settlement-Free Interconnection with Internet Networks." http://www.verizonbusiness.com/terms/peering/.
[11] "Qwest's North America IP Network Peering Policy." http://www.qwest.com/legal/peering_na.html.
[12] "Qwest's International IP Network Peering Policy." http://www.qwest.com/legal/peering_int.html.
[13] "Settlement-Free Peering Policy in USA for Autonomous System 3561." http://www.savvis.net/corp/savvis/peering.
[14] A. Flavel, A. Shaikh, T. Scholl, and M. Roughan, "Peer Dragnet: Analysis of BGP Peering Policies," *University of Adelaide Technical Report*, 2008.
[15] N. Feamster, Z. M. Mao, and J. Rexford, "BorderGuard: Detecting cold potatoes from peers," in *IMC '04*, pp. 213–218, 2004.
[16] T. Griffin and G. T. Wilfong, "Analysis of the MED Oscillation Problem in BGP," in *ICNP '02*, pp. 90–99, 2002.
[17] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong, "Route oscillations in I-BGP with route reflection," in *Proc. ACM SIGCOMM*, vol. 32, pp. 235–247, 2002.
[18] Y. Wang, I. Avramopoulos, and J. Rexford, "Morpheus: Making routing programmable," in *INM '07: Proceedings of the 2007 SIGCOMM workshop on Internet network management*, pp. 285–286, 2007.
[19] D. Walton, A. Retana, and E. Chen, "Advertisement of Multiple Paths in BGP." Internet draft, February 2006. http://tools.ietf.org/html/draft-walton-bgp-add-paths-05.
[20] "Introduction to Cisco MPLS VPN Technology." http://www.cisco.com/en/US/docs/net_mgmt/vpn_solutions_center/1.1/user/guide/VPN_UG1.html. Cisco Support Document.
[21] "JUNOS Software VPNs Configuration Guide, Release 8.5." http://www.juniper.net/techpubs/software/junos/junos85/swconfig85-vpns/frameset.html. Juniper Support Document.
[22] V. Jacobson, C. Alaettinoglu, and K. Poduri, "BST - BGP Scalable Transport," *NANOG 27 Presentation*, Feb 2003.
[23] I. Kouvelas, "Internal BGP Downloader," May 2004. International Patent Number WO 2004/040865 A1.
[24] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communications Review*, October 2003.
[25] T. G. Griffin and G. Wilfong, "On the correctness of IBGP configuration," in *Proc. ACM SIGCOMM*, pp. 17–29, 2002.
[26] Y. Rekhter and P. Gross, "Application of the Border Gateway Protocol in the Internet." RFC 1655, July 1994.
[27] C. Metz, C. Barth, and C. Filsfils, "Beyond MPLS ... Less is More," *IEEE Internet Computing*, vol. 11, no. 5, pp. 72–76, Sept.-Oct. 2007.
[28] T. Griffin, F. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 232–243, Apr 2002.
[29] N. Feamster and H. Balakrishnan, "Correctness properties for Internet routing," in *Annual Allerton Conference on Communication, Control, and Computing*, September 2005.
[30] T. G. Griffin and J. L. Sobrinho, "Metarouting," in *Proc. ACM SIGCOMM*, pp. 1–12, 2005.
[31] A. J. T. Gurney and T. G. Griffin, "Rethinking BGP Metrics," *unpublished report*, March 2008.
[32] C. Alaettinoglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, and C. Villamizar, "Routing Policy Specification Language (RPSL)." RFC 2622, June 1999.
[33] H. Boehm, A. Feldmann, O. Maennel, C. Reiser, and R. Volk, "AS-Wide Inter-Domain Routing Policies: Design and Realization," *NANOG 34 Presentation*, May 2005.
[34] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: Improving BGP convergence through root cause notification," *Computer Networks and ISDN Systems*, vol. 48, no. 2, pp. 175–194, 2005.
[35] L. Subramanian, M. Caesar, C. T. Ee, M. Handley, M. Mao, S. Shenker, and I. Stoica, "HLP: A next generation inter-domain routing protocol," in *Proc. ACM SIGCOMM*, pp. 13–24, 2005.
[36] X. Yang, "NIRA: A new Internet routing architecture," in *FDNA '03: Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pp. 301–312, 2003.
[37] D. Zhu, M. Gritter, and D. R. Cheriton, "Feedback based routing," in *Proc. ACM SIGCOMM*, vol. 33, pp. 71–76, 2003.
[38] W. Mühlbauer, A. Feldmann, O. Maennel, M. Roughan, and S. Uhlig, "Building an AS-topology model that captures route diversity," *SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 195–206, 2006.