Toward the Design of Robust Interdomain Routing Protocols

Aaron D. Jaggard, Tulane University
Vijay Ramachandran, International Computer Science Institute

Abstract

The Border Gateway Protocol, the interdomain routing protocol for the Internet, allows for a wide variety of routing policies that may interact in unintended and unstable ways. Recent work on BGP and related protocols has begun to incorporate formal protocol models, which have enabled rigorous descriptive analyses of BGP. More recently, such models have been used to give prescriptive guidelines for the design of new protocols. These guidelines include both sufficient conditions for good routing behavior and limitations on what can be achieved without coordination between routers. Here we review potential routing problems, various formal protocol models, and the design guidelines they have been used to prove.

The Border Gateway Protocol (BGP) [1] establishes best effort interdomain connectivity for the Internet. BGP is unique among IP routing protocols because routes are computed using fairly complex policies configured locally at each router. The interaction of these local policies can produce global routing anomalies [2, 3]. Because the domains or networks that form the Internet, called autonomous systems (ASes), are independently administered, policies are provided with little coordination. This often makes global anomalies hard to predict and debug because they involve policies across several networks.

BGP is the classic example of a *path-vector protocol*, in which routes are established as information about reachable destinations is disseminated hop by hop through the network. Figure 1 depicts the dynamics of the BGP route selection procedure: for each destination, the route choices of neighbors are examined; then from these possibilities a best path is chosen based on local policy; finally, the chosen path is shared with neighbors so that they can establish routes using the same process. (Information about a route first enters the network when a router directly linked to the destination announces its availability.) Best routes may change when new routes are learned or existing routes are withdrawn; this allows BGP to respond to network changes. By repeating the route choice process and sharing routes through update messages, BGP attempts to *converge*, in time, to a stable set of consistent routes.

Policies are only constrained by router programming languages and are not addressed in the BGP specification [1]. They can depend on various factors, such as security holes in distant ASes or business relationships with neighboring ASes. An improved understanding of how local policies affect convergence will lead to increased network stability. Ideally, we want to describe the protocols and policies that are *robust*, that is, those that converge to a predictable set of consistent routes even after link and node failures. Unfortunately, no complete characterization currently exists. (In this article we are only concerned with the effects of policies between ASes, i.e., *external BGP* [eBGP]. BGP is also used within an AS; for further details about BGP and anomalies in this setting, see [4].)

Initial work inspired by specific problems with BGP has led to recent work that better characterizes the behavior of path-vector protocols without involving the details of particular network configurations or protocol-specific implementations. As an application of theory to networking, this work uses formal models to prove that certain constraints guarantee good behavior, even in worst case scenarios; identifying these constraints is instrumental toward any characterization of stable protocols and policies. Note, however, that this rigorous treatment of protocol behavior is not a part of existing protocol adoption standards.

In this article we trace the development of this work and review best-known principles toward the design of robust path-vector protocols. First, we give canonical examples of BGP anomalies, motivating the need for robustness and other design goals. We then discuss a mathematical representation of policies that is useful in evaluating a network for robustness. After that, we discuss constraints on protocols that guarantee robustness on any network.

Motivation

Potential Problems with BGP

As network nodes write their routing policies and share data using BGP, the interaction between these policies can have undesirable effects: nodes in a network may not settle on a best route. An example of this was first given in [3] and is shown in Fig. 2a. We call this combination of a small network (or network fragment) and policies *bad gadget*, adopting the name of a similar example given in [5]. Every pair of nodes is connected by a link, and nodes 1, 2, and 3 are trying to select routes to node 0. Next to each node in Fig. 2a are the *permitted paths* each node will consider, listed in order of prefer-

0890-8044/95/$05.00 1995 © IEEE

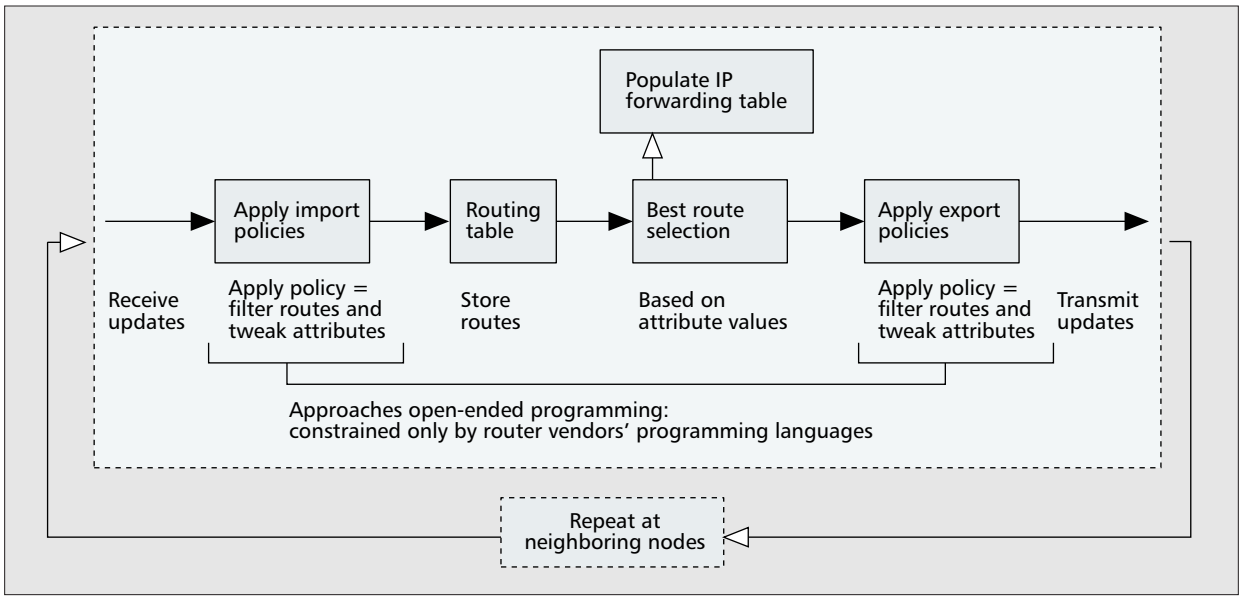

Figure 1. *Dynamics of BGP's route-selection procedure.*

ence: node 1 prefers the path through node 2 to 0 over the path directly from 1 to 0; these paths are denoted 120 and 10, respectively. We assume that node 1 does not learn about other routes because of routing policies (e.g., node 3 might not share route information with node 1, and paths containing loops are filtered out). Similarly, nodes 2 and 3 prefer routes through 3 and 1, respectively, over their direct routes to 0 and do not learn other routes. If no links fail, the direct paths to 0 are always known. Suppose that these direct paths are chosen at nodes 1–3. Following BGP dynamics (Fig. 1), these choices are advertised to neighbors, making available the more preferred indirect paths. Once the indirect paths are chosen, the direct paths are no longer advertised; withdrawal of these routes makes the indirect paths unavailable, and all nodes choose the direct paths again. This process repeats *ad infinitum*, never converging to a choice of routes. As shown in [3], this oscillation of path selections does not depend on the timing of updates in the network. Note that if any one of the outer nodes' policies were changed to prefer the direct path to 0, this oscillation would not occur, and the nodes would converge to a set of consistent routes.

We call a stable set of consistent route choices a *solution*. In a solution every node is assigned a path such that all paths are valid extensions of neighbors' paths, and no node is able to choose a more preferred path given its neighbors' choices. (This resembles a Nash equilibrium condition.) A solution may or may not be unique in a given network. The routing configuration shown in Fig. 2b, originally given in [5] and called *disagree*, has two solutions:

- 10 and 210
- 20 and 120

Either set of routes remains stable because no node can learn of a more preferred route. Unfortunately, the protocol does not have to converge to either of these solutions. If both nodes begin by choosing direct routes and advertising those to the other, the protocol can oscillate indefinitely, although this depends on the timing of various router operations.

Routing configurations with multiple solutions are not *predictable*: delays in BGP updates or different orderings of link failures and recoveries can result in different choices of routes for the same set of routing policies. In these cases routing might appear nondeterministic to network operators, making problem diagnosis difficult. We thus aim for routing configurations with unique solutions.

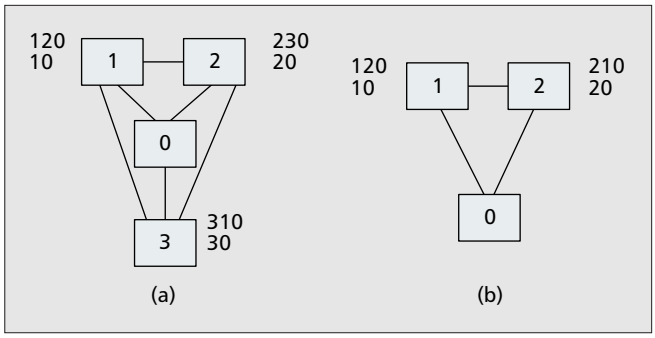

Figure 2. *The routing configurations (SPP instances): a) bad gadget; b) disagree.*

Design Goals for Path Vector Protocols

Anomalies due to no solution (e.g., *bad gadget*) or multiple solutions (e.g., *disagree*) are difficult to debug, because they involve policies in different ASes. Avoiding these situations, however, is only one goal of designing a routing system. Additional design goals were identified and rigorously defined in [6]; as we discuss later, there are inherent trade-offs among achieving multiple goals in any one protocol. These design goals include the following.

Robustness — Each network should have a unique stable set of paths to a given destination. A routing configuration is robust if it and every subnetwork, obtained by deleting some subset of edges and nodes from the original network, all have unique solutions. (This ensures stable routing behavior even after network failure.)

Expressiveness — It may be the case that networks running a certain protocol are guaranteed to be robust, but the protocol may achieve this by allowing a restricted set of routing policies (i.e., few network routing configurations may be compatible with the protocol). Ideally, we would like a protocol that guarantees robustness without overly constraining the types of policies allowed; we are thus interested in designing *expressive* protocols: those that can model as many different (robust) networks as possible.

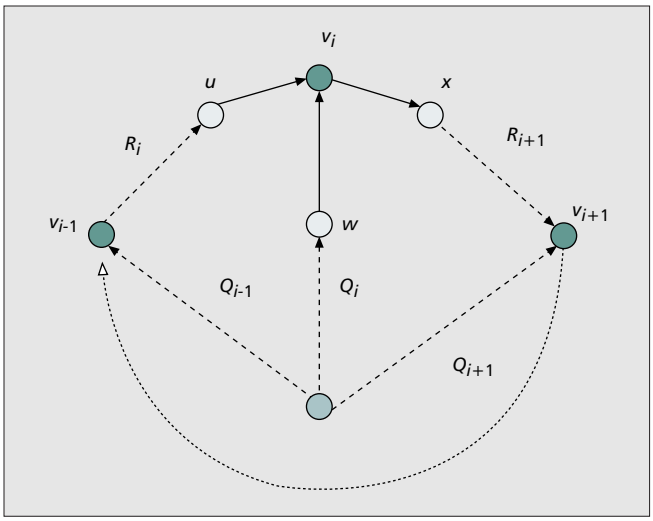

Figure 3. *A generic dispute wheel.*

Autonomy — As we model routing policies, not just their net effects, we care about how router operators can write these policies. While some amount of coordination might help achieve robustness, we expect policies to be written independent of other nodes' policies. We refer to this general goal as *autonomy*; in looking at particular classes of protocols, we may investigate specific types of autonomy that characterize different degrees of freedom.

There are additional protocol design goals that might be considered, but they involve more technical details about policy syntax and protocol implementation; see [6] for examples of these.

Studying Policies on Individual Networks

Initial work investigating the impact of policies on protocol convergence studied how BGP-like protocols ran on example networks. Varadhan, Govindan, and Estrin [3] outlined a basic view of routing dynamics and used this to characterize the timing-independent oscillations that could occur in simple classes of network topologies. Their motivating example was essentially the bad gadget of Fig. 2a, which was the first such oscillation shown for BGP. They also suggested that only shortest path routing, in which each node prefers the route with the fewest hops, might be provably safe in arbitrary network topologies. (Note that both bad gadget and disagree are inconsistent with shortest path routing.)

Griffin, Shepherd, and Wilfong [5] proposed the Stable Paths Problem (SPP) as a formal model for the underlying problem that BGP tries to solve–a problem of finding paths given complex routing policies, not just path length. An *instance* of SPP contains essentially the information given in Figs. 2a–2b above: a graph corresponding to the network and a set of permitted paths at each node; a node assigns each permitted path a *rank* — some positive number — corresponding to that node's level of preference for the path. (Individual path ranks are determined independently, i.e., not based on the rank of other paths; when this is not the case, as with BGP's multi-exit discriminator [MED] attribute, the modeling and analysis of protocols becomes much more difficult. The details of the MED attribute are outside the scope of this overview; see, e.g., [7] for a discussion of MED in BGP.) Which paths are permitted and which ranks are assigned result from routing policies across the network. Thus, an SPP essentially captures the static semantics of a network's routing policy configuration. A solution to an instance corresponds to a stable set of consistent routes, as described earlier.

This rigorous definition of the routing problem led to several insights in [5]. First, it was shown that solving the routing problem (i.e., determining whether or not a routing configuration has at least one solution) is *NP*-complete, even if a centralized algorithm is used. Checking a specific set of routing policies for a stable route assignment is thus presently impractical. Griffin *et al.* did show that the suggestion of Varadhan *et al.* concerning shortest path routing worked: in general topologies, if routers choose paths with the fewest hops, a solution is always reached. But, using their formalism, they could also prove much broader sufficient conditions on network policies that guarantee robustness.

Begin by considering generalizations of shortest path routing. Suppose that network edges could be assigned costs (e.g., based on traffic load or payment for usage); then the cost of a path is the sum of the costs of its component edges. When positive costs are used, lowest cost routing is essentially shortest path routing and, indeed, always converges to a solution. Although BGP does not support assigning edge costs in this way, in the design and study of a broader set of routing protocols we might envision several that allow this feature. In doing so, we can continue generalizing this notion of policies based on edge costs to understand the level of expressiveness at which our policies might cause routing anomalies. (This is useful because more complex policies can often be modeled by transforming them to cost assignments, given the network topology.) For example, if we allow the assignment of negative costs to edges, there may be a cycle of negative total cost, and following that cycle would artificially reduce the cost of any path through it. This could be done infinitely, and thus prevents finding a reasonable lowest cost path.

Griffin *et al.*, however, directly proved in [5] that if edges have negative costs, but every cycle in the graph has a positive cost, lowest cost routing is indeed robust. Such a cost assignment is called *coherent*. Coherence also precludes the divergent examples above. However, the authors of [5] gave an example of a convergent routing configuration in which policies are not consistent with coherent costs, suggesting that one might write a broader sufficient condition for robustness, the enforcement of which would permit a broader range of good routing policies.

Proving the coherence result involved characterizing a necessary condition for divergence. Griffin *et al.* captured this condition with a *dispute wheel*, which is a generalization of bad gadget. It is an abstraction of a network fragment in which policies are configured such that they might induce an oscillation. They described a procedure that, given an SPP, attempts to construct a solution; an unsuccessful attempt implies the existence of a dispute wheel in the SPP. They also showed that multiple routing solutions imply the existence of a dispute wheel. The combination of these results proves that SPPs without dispute wheels are robust, an observation central to subsequent work in this area.

A generic dispute wheel is shown in Fig. 3, in which specific edges are shown by solid lines, and generic paths are shown by broken lines. It comprises a *rim* (the outer circle) and *spokes* (the inner paths), allowing nodes and edges to appear multiple times in a single wheel. The rim and spokes are paths in the network graph such that routing policies at the *active nodes*, where spokes connect to the rim, conflict to allow bad routing behavior. In particular, each of these nodes v_i learns a path Q_i to the destination from its neighbor w down the spoke but would prefer to use the path R_i that follows the rim counter-clockwise through u to v_{i-1} and then goes down the next spoke Q_{i-1}. (It is easy to see that a three-node version of this network configuration is bad gadget; the dispute wheel in bad gadget is shown in Fig. 4.)

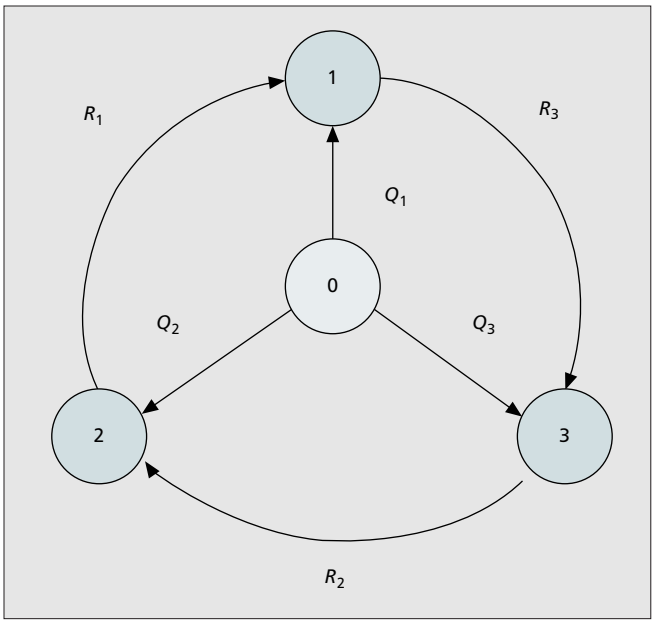

Figure 4. *The dispute wheel in bad gadget.*

Suppose all active nodes start by only knowing (and thus selecting) spoke paths. As time progresses, extensions of these routes may be further propagated through the network so that at each active node, a path counterclockwise through the rim to the next active node and then down that node's spoke becomes available. Because these routes are preferred, they will all be selected. Once this happens, active nodes can no longer advertise their spoke paths because they are not selected, and the spoke paths will be withdrawn. This eventually makes the extended paths through the rim unavailable, reverting all choices back to the direct spoke paths, at which point this sequence can start again.

Such an oscillation is not guaranteed to occur if a network contains a dispute wheel; for example, other paths might be preferred over all of the paths in the wheel, or, in the case of multiple solutions, timing of updates might induce temporary nondeterministic convergence to one of the solutions. But a dispute-wheel-free network will never oscillate and is always predictable. The condition of dispute wheel freedom is not a *local* condition but instead a restriction on how the local routing decisions of nodes may interact *globally*. This has guided much of the recent analysis of path-vector routing, as it has allowed researchers to specify local restrictions that help prevent dispute wheels from appearing in a network. In the next section we discuss frameworks that model protocols, not only network instances, and how they are used to prove that local conditions alone can be used to guarantee convergence, albeit while sacrificing some other protocol design goals.

Studying Protocol-Level Constraints

Previous work has suggested various constraints for protocols that guarantee robustness without examining specific networks. In this section we begin by discussing one set of assumptions about Internet hierarchy that does this. We then discuss two frameworks that can be used to abstractly model protocol behavior and policy constraints, and their results for guaranteeing robustness. Finally, we discuss a practical application of the frameworks, that of generalizing hierarchical constraints, to analyze a broad range of next-hop routing preferences commonly used for most policies today.

HBGP

Gao and Rexford [8] were the first to discuss the role of local constraints defined in terms other than cost increments assigned to links. They showed that an assumption about the Internet AS graph structure and a combination of simple rules for nodes' policies are enough to guarantee BGP's convergence. Fortunately, these rules and assumptions are consistent with, and naturally enforced by, common Internet economics.

Two connected ASes usually view their relationship as between either a customer and a provider of network connectivity or two equals; in the second case, these *peers* may use their connection to provide backup connectivity, connect their customers, or shortcut expensive or longer routes through provider links. In this Hierarchical BGP (HBGP) model, every AS assigns one of the labels (customer, provider, or peer) to each of its neighbors such that this view is consistent with that of other ASes (e.g., an AS's customers view it as a provider).

Nodes' routing policies usually satisfy certain rules, defined in terms of these labels, on the relative ranking of routes and with whom routes are shared; these restrictions are natural given the traffic agreements usually made with the three types of neighbors. (For example, routes learned from customers must be preferred to routes learned from providers, and the latter are shared only with customers, not peers or other providers.) Finally, it is assumed that no "customer/provider" cycles exist (i.e., no AS is an indirect customer of itself). This last restriction is also natural in that it is very unlikely that a local ISP would sell network connectivity to a top-level network.

If ASes have such a hierarchical structure and obey these rules, BGP converges. Thus, we see that with constraints on policies, we can guarantee convergence. Although this initial result was proved directly, Gao, Griffin, and Rexford [9] generalized it by adding backup routes to HBGP and proved the result using machinery from [5]. Note that backup routes today are often established using *community values*, which are extra parameters in BGP updates. These values are not often propagated through the network. Consider the network shown in Fig. 5. Suppose that nodes 3 and 0 have a special agreement to use the link between them as a backup link; the resulting configuration is shown as Griffin's *route pinning* SPP in Fig. 5a. At node 3, route 30 is the less preferred of the two available because it is a backup; the preferences at nodes 1 and 2, assuming that the backup agreement is not specifically propagated, are set using standard HBGP rules (preferring customers). This instance has two solutions, like disagree, and can thus recover from a link failure nondeterministically. An analysis yields the dispute wheels shown in Fig. 5b.

In defining these restrictions, HBGP moves away from any suggested policy language, such as policies determined by edge costs, to the more general approach of giving local constraints on the effects of policies. Later work generalizes these constraints by investigating sets of protocols and constraints that together guarantee robustness; we examine this later. Although HBGP's rules are enforced today for the most part, complex configuration (e.g., route pinning) and misconfiguration can still result in anomalies that are hard to debug. Furthermore, when multiple ASes are administered by the same provider, the natural rules for inter-AS economics do not apply. Therefore, some further understanding of complex policy interactions is necessary.

Path-Vector Policy Systems

The path-vector policy system (PVPS) was introduced by Griffin, Jaggard, and Ramachandran in [6] and was designed to explicitly model many components of the routing process. Thus, a PVPS includes: a protocol definition, which deter-

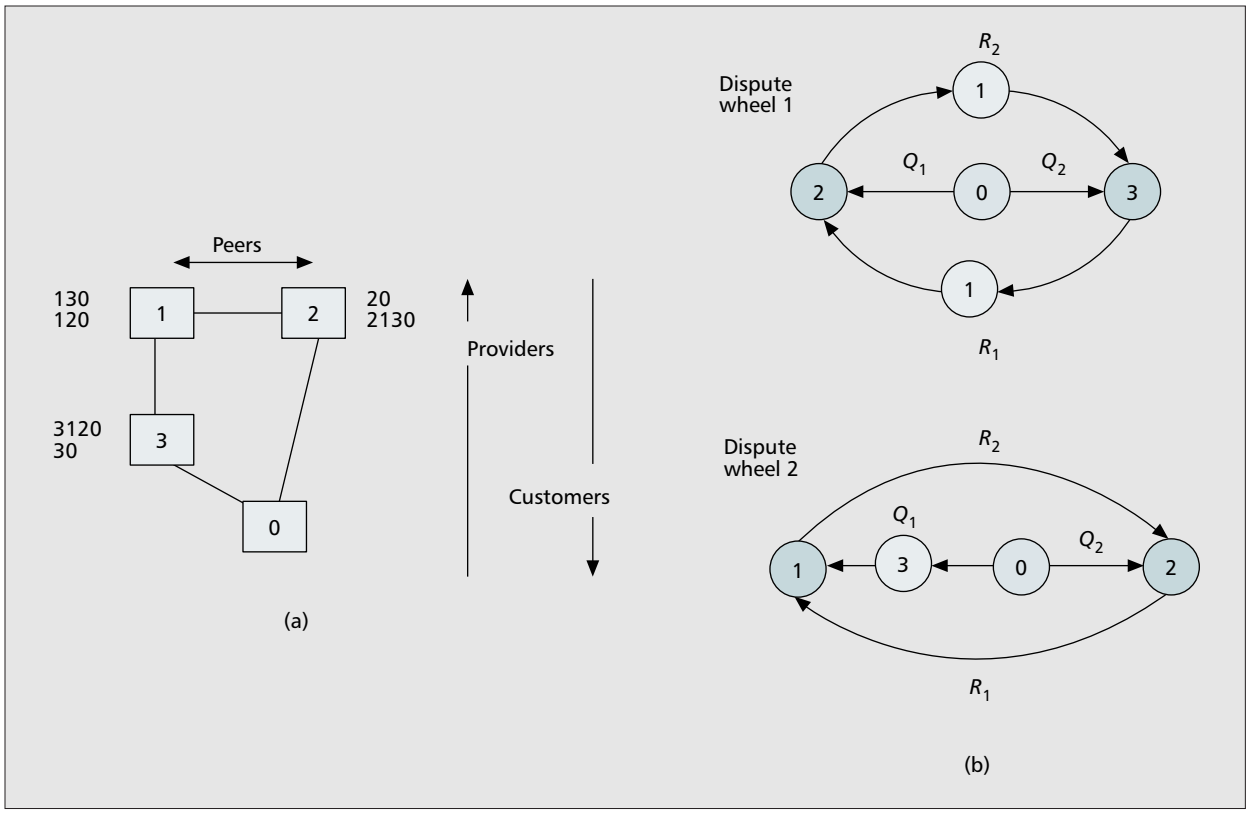

Figure 5. *a) The route pinning SPP; b) its two dispute wheels.*

mines how the protocol mediates the interaction between nodes running it, including the route data structure and route-selection procedure; a policy language, which nodes may use to write policies capturing how they process path data; and a set of assumptions about the network. These assumptions are *global* in the sense that no single node is expected to have enough information to verify that they do indeed hold. (In contrast, the policy language and specification details of policy constraints provide a *local* restriction on how nodes may treat path data. It is in this way policies can be constrained to avoid inputs that lead to routing anomalies.)

One of the main results obtained using this formal model was the description of a local policy condition equivalent to dispute wheel freedom. This condition, satisfied by an *increasing* PVPS, requires that each path can be mapped to some absolute rank value, and that these values increase as paths are extended from one node to another. (As with path costs in earlier models, lower-ranked paths are preferred in PVPSes). Intuitively, an increasing PVPS requires a positive cost on each network edge, although *this cost may depend on the path using the edge as well as the edge itself*, in contrast with the path-independent costs considered previously. Because these costs must be positive, the increasing PVPS condition may seem more restrictive than coherence. However, it was shown in [6] that any network configuration satisfying the conditions of [5] (or any other instance without a dispute wheel) is *equivalent* to one permitted by an increasing PVPS (i.e., some increasing PVPS allows a network configuration with the same permitted paths and relative preferences at each node). The increasing condition precludes bad gadget and disagree. (If we try to write disagree in an increasing system, the rank of 10 must be less than that of 210, a path extending it, which in turn must be less than the rank of 20, which is less preferred than 210. 120 extends 20 and must have greater rank; as its rank is smaller than the less preferred route 10, we obtain a cycle of strict rank inequalities, a contradiction.)

When we view increasing PVPSes as having positive path-dependent edge costs — the original definition contained this idea, but used different language — it is clear that there is some consistent mathematical order involved. In particular, preferences at each node must correspond to cost, which in turn increases as paths are extended. After showing that dispute-wheel-free networks correspond to those running increasing PVPSes, Griffin *et al.* [6] showed that these network configurations are exactly those in which the preference order at each node is consistent with "preferring" a path to any extension of that path. (Such paths are never compared in any route selection procedure because they start at different nodes, but the preference ordering should remain consistent if extended to include these comparisons. Mathematically, they showed that these relations can be extended to a partial order without antisymmetry.) Anomaly-free routing is guaranteed by the absence of dispute wheels, which is in turn equivalent to some mathematical consistency in the network; the question is then how to ensure that consistency.

In an effort to answer this question, the authors of [6] identified and rigorously defined various protocol design goals, including the three mentioned above: expressiveness, robustness, and autonomy. Using these definitions, they showed that enforcing the increasing condition on PVPSes generally infringes on operator autonomy (because policies would have to be continually adjusted based on neighbors' assignments of rank in order to maintain the increasing condition) or requires private information to be shared among nodes, or the protocol to filter routes on the operator's behalf, thus altering the intent of routing policy. In fact, it was proved that achieving a reasonable combination of design goals requires some global constraint on the network: while local conditions alone can enforce robustness, one must take into account global network assumptions when designing a practical routing system. By using a rigorous formal model, [6] was able to prove these inherent design trade-offs independent of specific networks or

protocols. Feamster, Johari, and Balakrishnan [10] extended this idea to other types of autonomy and produced a necessary condition for convergence using a variation of the dispute wheel called a *dispute ring*.

Figure 6, adapted from [6], shows the design space of robust PVPSes. (This figure is meant to aid in developing intuitions, and should not be taken too literally.) A point in the design space indicates a PVPS specification; its position indicates the approximate trade-off between expressiveness and global constraint tractability. (Other design dimensions are not pictured in this figure.) The x-axis shows expressive power, ranging from shortest path routing to allowing all possible robust configurations. We measure expressive power using the SPP framework: Given a PVPS (with a protocol specification and policy language), we consider the set of all SPP instances (network configurations) that correspond to legal sets of input policies. Note that the robust design space of Fig. 6 does not include PVPSes expressive enough to write all possible preference orderings on paths; any PVPS expressive enough to write disagree is not robust. The y-axis shows strength of global constraint, from minimum (no constraint) to maximum (checking all policies individually for robustness). Ideally, we would like systems near the lower right of Fig. 6; however, as discussed, these infringe on other design goals.

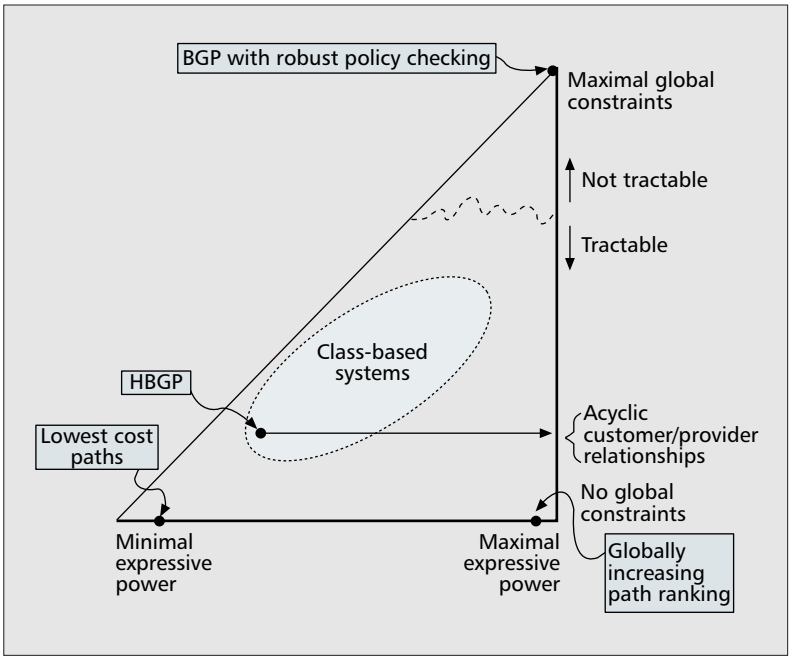

Figure 6. *The design space of robust path vector policy systems.*

Path-Vector Algebras

At the same time PVPSes were introduced, Sobrinho [11] introduced the *path-vector algebra*, a model of path-vector protocols at a higher level of abstraction. In an algebra all of the operations involved in sending route updates, including the application of policies, are represented as a single object per link, its *label*. Route information is represented by a *signature*, and each signature maps to a weight (analogous to rank in a PVPS) that indicates how good a route is. Route updates can be modeled in the following way: When a route is shared across a link, the label of the link is *applied* (as defined by the algebra) to the sender's signature for the route, producing a new signature at the receiver. Thus, conditions on policies and their effects on convergence are expressed in terms of how labels affect weights.

Sobrinho [11] used the algebra framework to prove that a certain condition can be used to guarantee that every node in a network will receive its first choice path, not just its most preferred extension of a neighbor's path, in a routing solution. This condition (*isotonicity*) relates to how the extension of two paths affects their relative ranking; in particular, if one path is at least as preferred as another at a node, a neighboring node must prefer the extension of the first path at least as much as the extension of the second path. Sobrinho also showed that an increasing condition, called *strict monotonicity* in [11] and mirroring the increasing PVPSes discussed above, guarantees robustness, as does other conditions when coupled with a weaker version of monotonicity. He also presented conditions that, along with a weaker version of monotonicity, guarantee robustness.

Algebras are a natural framework to discuss the overall effects of policy on sharing data across an edge, while PVPSes may be more natural for analysis of specific implementations of policy constraints. Jaggard and Ramachandran [12] have recently shown that the algebra and PVPS frameworks are essentially equivalent, and provided a template for translating between them. Thus, in studying path-vector protocols, the more natural framework for describing the relevant questions should be used; the results may then be translated to the other framework as needed.

Class-Based Systems

As an application of the PVPS framework, Jaggard and Ramachandran [13] investigated class-based systems, generalizing the conditions given by Gao *et al.* [9] for robust HBGP with backup routing. Gao and Rexford's earlier work [8] on HBGP suggested a specific combination of local and global constraints, generally enforced by current Internet economics, that guarantee robust routing. The work in [13] extended this by allowing for general relationships between network nodes — a general list of classes, not just customer, provider, and peer — and constructing a global constraint whose satisfaction guarantees robustness, based on the local restrictions defined in terms of these relationships.

A class-based system comes with a (finite) list of class labels, which nodes apply to their neighbors, and restrictions on how these may be used (i.e., which pairs of labels can be assigned together by pairs of neighbors on the same edge). As with HBGP, the system also specifies when a node must prefer routes learned from neighbors in one class over routes learned from neighbors in another class, as well as the classes of neighbors with which a route learned from a neighbor in a given class may be shared.

In HBGP (with and without backup routing), networks were required to avoid customer/provider cycles. The authors of [13] generalized this global constraint for use in class-based systems: given the list of classes, and the restrictions on preferring and sharing route data, they constructed a condition on pairs of classes and then required that networks avoid cycles in which every adjacent pair of edges have class labels satisfying this condition. In the case of HBGP, this constraint exactly reduces to avoiding customer/provider cycles. Furthermore, this constraint is only as restrictive as needed; as shown in [13], networks satisfying it are robust, while networks violating it can set policies that cause divergence. Finally, centralized and distributed algorithms to enforce this constraint were given. The algorithms can be used more broadly; for example, any network configuration in which route preferences are pri-

marily determined by the *next hop of a path* (or the first edge) can be checked for potential routing anomalies by constructing a corresponding class-based system for the routing policies and running the centralized algorithm.

Conclusions

We have reviewed recent work aimed at understanding policy conflicts in networks running path-vector protocols, and how these can be prevented through the design of protocols and routing policies. Initial work in this area focused on network-level examples of route oscillation and conditions on individual networks that would ensure globally good routing behavior. More recent work has emphasized protocol-level analysis and has incorporated consideration of both (local) routing policies in networks and (global) assumptions about the network; the latter are provably necessary in order to achieve reasonable protocol design goals.

This work should foster the design of routing protocols and policy languages that ensure robustness; this is one topic for further work. Given a desired level of expressiveness, the frameworks discussed here can be used to understand potential policy-induced anomalies and generate constraints that can be enforced in the protocol specification, built into a language used to write policies, or through some supplementary mechanism. While the tools developed so far allow us to reason about basic BGP, modifications to the original protocol can lead to other routing problems but are not amenable to modeling in these frameworks. We expect future work to investigate this.

Acknowledgments

This work was partially supported by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR). A. D. Jaggard was partially supported by National Science Foundation (NSF) Grant DMS-0239996 and by ONR Grants N00014-01-1-0795 and N00014-99-1-0150. V. Ramachandran was partially supported by a 2001-2004 DoD National Defense Science and Engineering Graduate (NDSEG) Fellowship, by ONR Grant N00014-01-1-0795, and by NSF Grant ITR-0219018. The authors would like to thank the anonymous reviewers for their suggestions to improve the organization and readability of this article.

References

[1] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 1771, Mar. 1995; http: //www.ietf.org/rfc/rfc1771.txt
[2] Cisco Field Note, "Endless BGP Convergence Problem in Cisco IOS Software Releases," Oct. 2001. http://www.cisco.com/warp/public/770/fn12942.html
[3] K. Varadhan, R. Govindan, and D. Estrin, "Persistent Route Oscillations in Inter-domain Routing," *Comp. Networks*, vol. 32, no. 1, Jan. 2000, pp. 1–16.
[4] R. Musunuri and J. A. Cobb, "An Overview of Solutions for Persistent BGP Divergence," *IEEE Network*, this issue.
[5] T. G. Griffin, F. B. Shepherd, and G. Wilfong, "The Stable Paths Problem and Interdomain Routing," *ACM/IEEE Trans. Net.*, vol. 10, no. 2, Apr. 2002, pp. 232–43.
[6] T. G. Griffin, A. D. Jaggard, and V. Ramachandran, "Design Principles of Policy Languages for Path- Vector Protocols," *Proc. ACM SIGCOMM '03*, Aug. 2003, pp. 61–72; extended version: Yale Univ. Tech. Rep. YALEU/DCS/TR-1250, Apr. 2004; ftp://ftp.cs.yale.edu/pub/TR/tr1250.pdf
[7] T. G. Griffin and G. T. Wilfong, "Analysis of the MED Oscillation Problem in BGP," *Proc. ICNP '02*, Nov. 2002, pp. 90–99.
[8] L. Gao and J. Rexford, "Stable Internet Routing Without Global Coordination," *ACM/IEEE Trans. Net.*, vol. 9, no. 6, Dec. 2001, pp. 681–92.
[9] L. Gao, T. G. Griffin, and J. Rexford, "Inherently Safe Backup Routing with BGP," *Proc. IEEE INFOCOM 2001*, Apr. 2001, pp. 547–56.
[10] N. Feamster, R. Johari, and H. Balakrishnan, "Implications of Autonomy for the Expressiveness of Policy Routing," *Proc. ACM SIGCOMM 2005*, Aug. 2005, pp. 25–36.
[11] J. L. Sobrinho. "Network Routing with Path Vector Protocols: Theory and Applications," *Proc. ACM SIGCOMM '03*, Aug. 2003, pp. 49–60.
[12] A. D. Jaggard and V. Ramachandran, "Relating Two Formal Models of Path-Vector Routing," *Proc. IEEE INFOCOM '05*, Mar. 2005; extended version: Yale Univ. Tech. Rep. YALEU/DCS/TR-1301, Mar. 2005; ftp://ftp.cs.yale.edu/pub/TR/tr1301.pdf
[13] A. D. Jaggard and V. Ramachandran, "Robustness of Class-Based Path-Vector Systems," *Proc. ICNP 2'04*, Oct. 2004, pp. 84–93. Extended version: Yale Univ. Tech. Rep. YALEU/DCS/TR-1296, Mar. 2005; ftp://ftp.cs.yale.edu/pub/TR/tr1296.pdf

Biographies

AARON D. JAGGARD (adj@math.tulane.edu) is an NSF VIGRE postdoctoral fellow in the Department of Mathematics at Tulane University. He took his B.S. (1998) and Ph.D. (2003) degrees in mathematics from Wheaton College, Illinois, and the University of Pennsylvania, respectively. In addition to network routing, his current research interests include information security and combinatorics.

VIJAY RAMACHANDRAN (vijayr@icsi.berkeley.edu) is a postdoctoral researcher at the International Computer Science Institute (ICSI), Berkeley, California. He received his A.B. (2000) from Princeton University and his Ph.D. (2005) from Yale University. His research interests include Internet algorithmics, including formal study of routing and other Internet services.