

# Towards the Design of Robust Inter-domain Routing Protocols\*

Aaron D. Jaggard<sup>†</sup>

Dept. of Mathematics, Tulane University  
adj@math.tulane.edu

Vijay Ramachandran<sup>‡</sup>

Dept. of Computer Science, Yale University  
vijayr@cs.yale.edu

## Abstract

*The Border Gateway Protocol (BGP), the inter-domain 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 that they have been used to prove.*

## 1. Background

The Border Gateway Protocol (BGP) [8] is the protocol used to establish best-effort inter-domain connectivity for the Internet. BGP is unique among IP-routing protocols in that routes are computed using policies configured locally at each router. Because the domains or networks that form the Internet—called autonomous systems (ASes)—are independently administered, a stable set of consistent Internet routes depends on inputs provided with little coordination. The interaction of these local routing policies can produce global anomalies, *e.g.* nondeterministic routing and protocol divergence [1, 10]; these anomalies are often hard to 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 reach-

able destinations is disseminated hop-by-hop through the network. Figure 1 depicts the dynamics of the BGP route-selection procedure: the route choices of neighbors are imported; from these possibilities, best paths are chosen based on local policy; the choices are exported to neighbors to establish routes using the same process. (Information about a route first enters the network when it is announced by a router directly linked to the destination.) Re-calculation of best routes is triggered by learning new routes or receiving withdrawals of existing routes, thus allowing BGP to respond to network changes. Policies are only constrained by router programming languages and are not addressed in the BGP specification (RFC 1771 [8]); policies can depend on varied factors, *e.g.*, insecurity of distant ASes or business relationships with neighboring ASes. It is the composition of expressive, autonomous policies that produce routing anomalies.

An improved understanding of how local policies affect protocol convergence will lead to increased network stability. Ideally, we want to describe a protocol and policies that are *robust*, *i.e.*, those that converge to a predictable set of consistent routes, even with link and node failures present. Initial work inspired by specific problems with BGP has led to recent works that better characterize 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, these works use formal models to prove that certain constraints guarantee good behavior, even in worst-case scenarios; this rigorous treatment of protocol behavior is not a part of existing protocol-adoption standards. In this paper, we trace the development of this work and review best-known principles towards the design of robust path-vector protocols.

## 2. Motivation

### 2.1. Potential problems with BGP

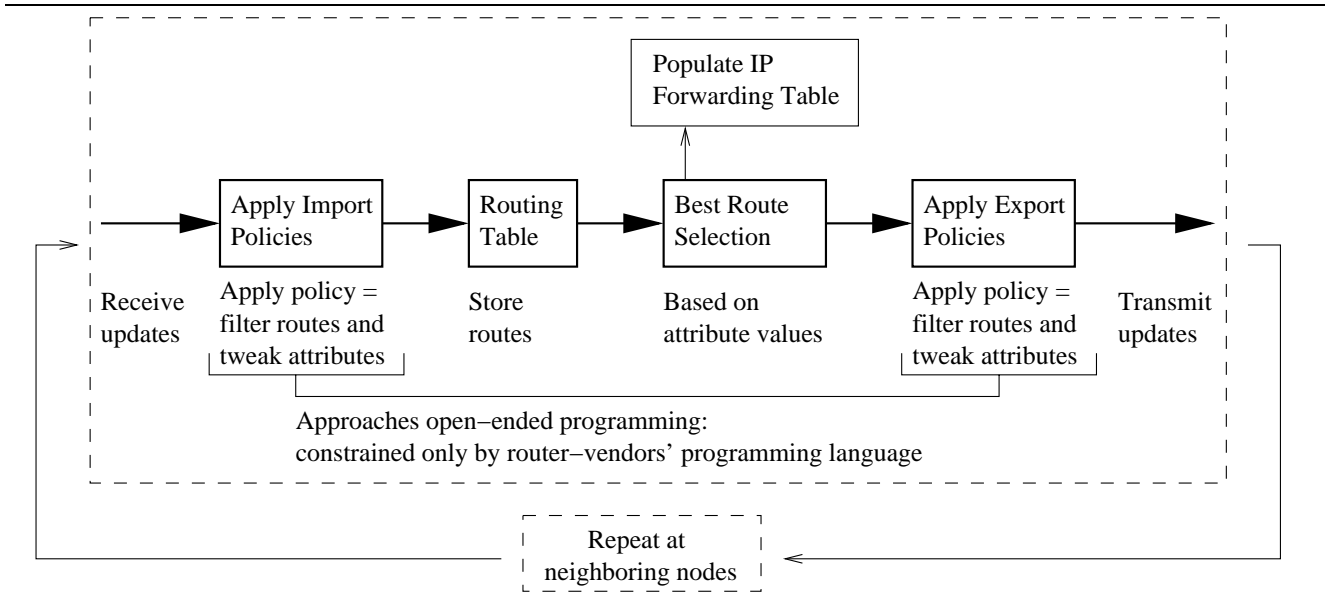
As network nodes write their routing policies and share data using BGP, the interaction between these policies can have undesirable effects: nodes in a network may not settle on a best route. An example of this was first given in [10]

---

\* 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).

† Partially supported by National Science Foundation (NSF) Grant DMS-0239996 and by ONR Grants N00014-01-1-0795 and N00014-99-1-0150.

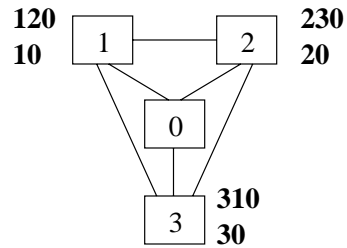
‡ Partially supported by a 2001–2004 DoD National Defense Science and Engineering Graduate (NDSEG) Fellowship, by ONR Grant N00014-01-1-0795, and by NSF Grant ITR-0219018.



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

and is shown in Figure 2. We call this BAD GADGET, adopting the name of a similar example given in [5]. It consists of a small network in which nodes 1, 2, and 3 try to select routes to node 0; every pair of nodes in this network is connected by a link. Next to each node in Figure 2 are the *permitted paths* that each node will consider, listed in order of preference: 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 (Figure 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 [10], this oscillation of path selections does not depend on timing 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 have a stable set of consistent routes.

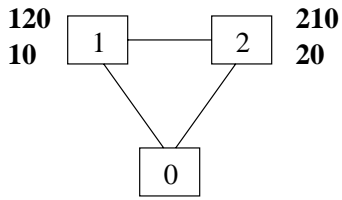
We call a stable set of route choices a *solution*. In a solution, every node is assigned a path such that (1) all paths are valid extensions of neighbors' paths and (2) no node is able to choose a more preferred path given its neighbors'



**Figure 2. The routing configuration (SPP instance) BAD GADGET.**

choices. (This resembles a Nash-equilibrium condition.) A solution may or may not be unique in a given network. The routing configuration shown in Figure 3, originally given in [5] and called DISAGREE, has two solutions: (1) 10 and 210; and (2) 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. In an oscillation similar to that of BAD GADGET, if both nodes begin by choosing direct routes and advertising those to the other, the protocol can oscillate indefinitely, though this depends upon the timing of various router operations.

Routing configurations with multiple solutions are not *predictable*: delays in sending BGP update messages 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.



**Figure 3. The routing configuration (SPP instance) DISAGREE.**

We thus aim for routing configurations with unique solutions.

## 2.2. Design Goals for Path-Vector Protocols

In the previous section, we saw that oscillations or non-determinism due to no solution (*e.g.*, BAD GADGET) or multiple solutions (*e.g.*, DISAGREE) are difficult to debug. Avoiding these situations, however, is only one goal of designing a routing system. [4] identified and rigorously defined additional design goals; as we will discuss later, there are inherent trade-offs among achieving multiple goals in any one protocol. Major design goals include:

**Robustness** Each network should have a unique, stable set of paths to a given destination. A routing configuration is *robust* if it and every sub-network, obtained by deleting some subset of edges and nodes from the original network, all have unique solutions. (This ensures good 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, *i.e.*, those that can model as many different (robust) networks as possible.

**Autonomy** As we model routing policies, and 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 independently 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 [4] for examples of these.

## 3. Studying Policies on Individual Networks

Much initial work investigating the impact of policies on protocol convergence studied how BGP-like protocols ran on example networks. Varadhan, Govindan, and Estrin [10] 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 BAD GADGET of Figure 2, which was the first such oscillation shown for BGP. They also suggested that 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] gave a more detailed formal model for networks running path-vector protocols and proposed the *Stable Paths Problem* (SPP) as the underlying problem that BGP solves. An *instance* of SPP contains essentially the information given in Figures 2–3 above: a graph corresponding to the network and a set of permitted paths at each node, each of which is assigned a *positive rank* by the node at which it is permitted corresponding to that node's level of preference for the path. (The rank of a path must be determined independently; when this is not the case, as with BGP's MED attribute, the modeling and analysis of protocols becomes much more difficult.) What paths are permitted and path-rank values result from interactions between routing policies across the network being modeled; 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 we 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, then a solution is reached. Using their formalism, they could also prove much broader sufficient conditions on network policies that guarantee robustness.

In the first logical generalization of shortest-paths routing, let each network link be assigned an arbitrary positive cost; the cost of a path is then the sum of the costs of its component edges. Lowest-cost routing is thus analogous to shortest-paths routing and, indeed, always converges to a

routing solution. Griffin, *et al.* [5] directly proved an even more general condition, showing the sufficiency of *coherent cost assignments* for robustness. In a coherent assignment, edges may have negative costs, so long as every cycle in the graph has a positive cost (*i.e.*, the sum of edge costs around any cycle is positive). Intuitively, the potential problem with negative edge costs is that a path can traverse a cycle of negative-cost edges multiple times, artificially reducing total path cost; however, preventing non-positive cycles makes this impossible. Coherence also precludes the divergent examples from 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.

Proving the coherence result involved characterizing a necessary condition for divergence. Using SPP’s abstract model of a network configuration, Griffin *et al.* showed that a generalization of BAD GADGET, called a *dispute wheel*, captures this condition. In particular, they described a procedure that attempts to construct a routing solution given an SPP; an unsuccessful attempt implied the existence of a dispute wheel in the SPP. They also showed that multiple routing solutions implied the existence of a dispute wheel. Thus, an SPP without a dispute wheel is robust; this observation was central to the coherence result and subsequent work in this area.

A generic dispute wheel is shown in Figure 4. It comprises a *rim* and *spokes*, which are paths in the network graph (allowing nodes and edges to appear multiple times in a single wheel) 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 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.)

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.

While such an oscillation is not guaranteed to occur if a network contains a dispute wheel—there may be other paths preferred over all of the paths in the wheel—a dispute-wheel-free network cannot produce this type of oscillation.

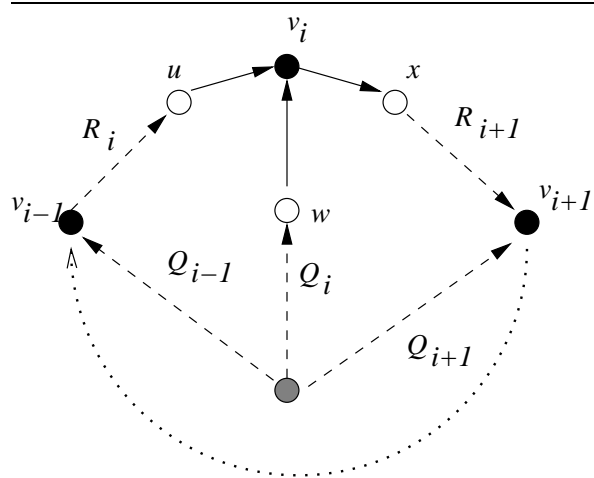


Figure 4. A generic dispute wheel.

The condition of dispute-wheel-freeness 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 which help prevent dispute wheels from appearing in a network. In the next section, we will 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.

## 4. Studying Protocol-Level Constraints

### 4.1. HBGP

Gao and Rexford [3] were the first to discuss the role of local constraints defined in terms other than cost increments assigned to links. They show 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 are naturally enforced by, common Internet economics.

Two connected ASes usually view their relationship either as between a customer and a provider of network connectivity or as between two equals; in the second case, these “peers” may use their connection to provide backup connectivity, to connect their customers, or to short-cut 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).

HBGP then requires that nodes' routing policies satisfy certain restrictions, 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. While the initial results for HBGP convergence were proved directly, Gao, Griffin, and Rexford [2] generalized this work by adding back-up routes to HBGP; using machinery from [5], they showed that networks satisfying generalizations of the conditions from [3] avoid dispute wheels and are thus robust.

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 now turn to two independently developed formal models that allow the study of protocol-level constraints without referring to the specifics of any protocol or network.

## 4.2. Path-Vector Policy Systems

The *Path-Vector Policy System* (PVPS) was introduced by Griffin, Jaggard, and Ramachandran in [4] and was designed to explicitly model many components of the routing process. In order for nodes in a network to run a routing protocol, each node must determine how it wishes to treat path data. Thus a PVPS includes: a protocol definition, which determines how the protocol mediates the interaction between nodes running it; 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 that 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 that is equivalent to dispute-wheel freeness. 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 [4] 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, Jaggard, and Ramachandran [4] 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.) Good routing behavior 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 [4] 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 (1) private information to be shared among nodes, or (2) the protocol to filter routes on the operator's behalf, thus altering the intent of routing policy. In fact, it was proven

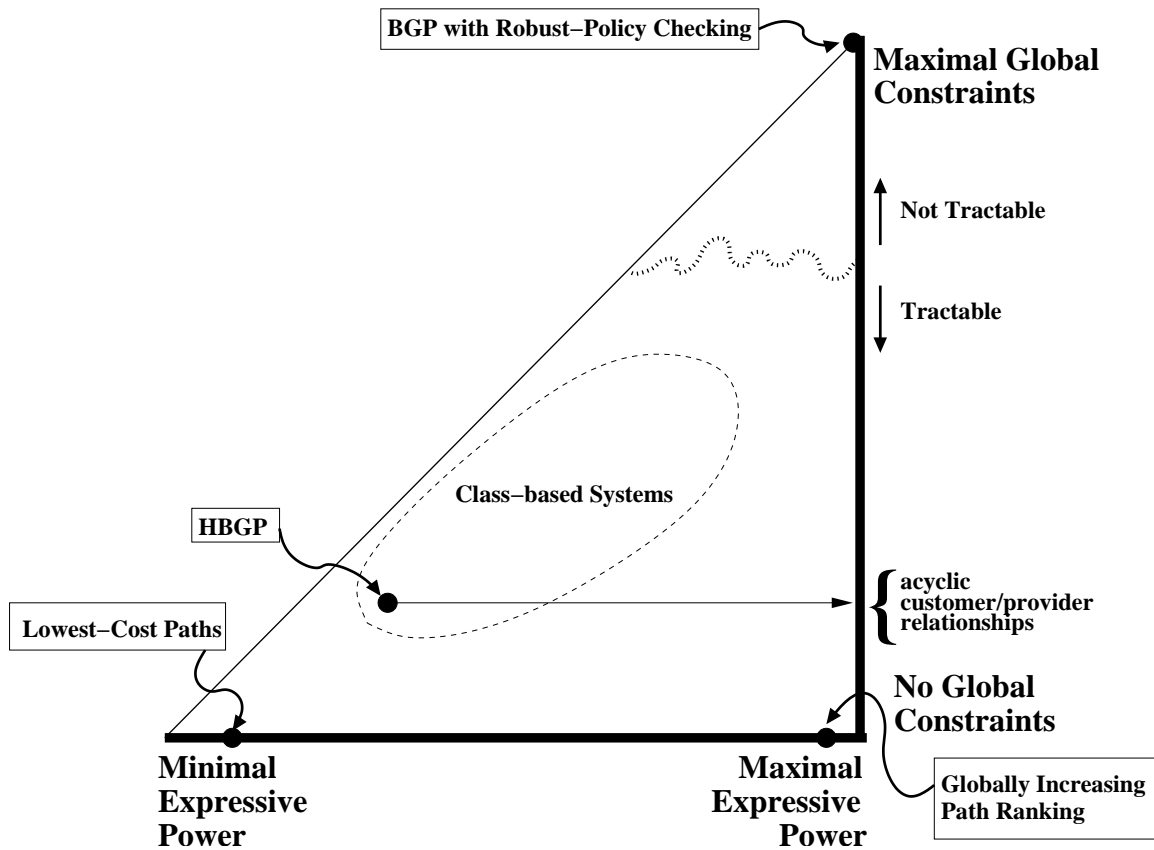


Figure 5. The design space of robust path-vector policy systems.

that achieving a reasonable combination of design goals requires some global constraint on the network; *i.e.*, 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, [4] was able to prove these inherent design trade-offs independent of specific networks or protocols.

Figure 5, adapted from [4], shows the design space of robust path-vector policy systems. (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-paths routing to allowing all possible robust configurations. We measure expressive power using the SPP framework: Given a PVPS (with a protocol specification and a 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 Figure 5 does not include PVPSes expressive enough to write all possi-

ble preference orderings on paths—any PVPS expressive enough to write DISAGREE is not robust. The  $y$ -axis shows strength of global constraint, from the minimum (no constraint) to the maximum (checking all policies individually for robustness). Ideally, we would like systems near the lower-right of Figure 5; but, as we have discussed, these infringe on other design goals.

### 4.3. Path-Vector Algebras

At the same time PVPSes were introduced, Sobrinho introduced the *path-vector algebra* model [9], an elegant way of modeling path-vector protocols at a higher level of abstraction. An algebra represents the all of the transformations of path data being shared across network link as a single object, the *label* of that link; this label combines the effects of the routing policies of the nodes exchanging data with any effects of the protocol itself on this data. The dynamics of sharing path information is modeled by *applying* these labels to the path data—called *signatures*—in order to obtain new path data; each signature maps to a weight (analogous to rank in a PVPS). The conditions on algebras given in [9] were expressed in terms of the effects of labels on

weights.

Sobrinho used the algebra model to prove a that a certain condition can be used to guarantee that every node in a network will receive its first-choice path, and not just its most preferred extension of a neighbor's path, in a routing solution. This condition (*isotonicity*) relates 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, then 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 *monotonicity* in [9] and mirroring the increasing PVPSes discussed above, guarantees robustness. He also presented conditions which, along with weak monotonicity ( $\geq$  rather than  $>$ ), guarantees robustness.

Because of their abstraction, algebras are a very natural framework for discussing the overall effects of sharing data across an edge, and for stating restrictions on these effects in order to guarantee robustness, while PVPSes may be more natural for analysis of specific implementations of policy constraints. Jaggard and Ramachandran have recently [7] 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.

#### 4.4. Class-Based Systems

As an application of the PVPS framework, Jaggard and Ramachandran [6] investigated “class-based systems,” generalizing the conditions given by Gao, Griffin, and Rexford [2] for robust HBGP with back-up routing. Gao and Rexford's earlier work [3] on HBGP had suggested a specific combination of local and global constraints, generally enforced by current Internet economics, that guarantee robust routing. The work in [6] 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 labels may be applied to a node by one of its neighbors to which it has given a particular label). 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 back-up routing), net-

works were required to avoid customer/provider cycles. The authors of [6] 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 reduces to avoiding exactly customer/provider cycles. Furthermore, this constraint is only as restrictive as needed; as shown in [6], 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; *e.g.*, any network configuration in which route preferences are primarily 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.

## 5. Conclusions

Here 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. While the tools developed so far allow us to reason about basic BGP, modifications to the original protocol (such as the MED attribute) can lead to other routing problems but are not amenable to modeling in these frameworks. We expect future work to investigate this.

## References

- [1] Cisco Field Note. Endless BGP Convergence Problem in Cisco IOS Software Releases. Oct. 2001. <http://www.cisco.com/warp/public/770/fn12942.html>
- [2] L. Gao, T. G. Griffin, and J. Rexford. Inherently Safe Backup Routing with BGP. In *Proc. INFOCOM 2001*, pp. 547–556, Apr. 2001.
- [3] L. Gao and J. Rexford. Stable Internet Routing Without Global Coordination. *ACM/IEEE Trans. on Networking*, 9(6): 681–692, 2001.

- [4] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design Principles of Policy Languages for Path-Vector Protocols. In *Proc. ACM SIGCOMM'03*, pp. 61–72, Aug. 2003. Extended version: Yale Univ. Tech. Report YALEU/DCS/TR-1250, Apr. 2004. <ftp://ftp.cs.yale.edu/pub/TR/tr1250.pdf>
- [5] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The Stable Paths Problem and Interdomain Routing. *ACM/IEEE Trans. on Networking*, 10(2):232–243, 2002.
- [6] A. D. Jaggard, V. Ramachandran. Robustness of Class-Based Path-Vector Systems. In *Proc. ICNP 2004*, Oct. 2004. Extended version: Yale Univ. Tech. Report YALEU/DCS/TR-1296, Mar. 2005. <ftp://ftp.cs.yale.edu/pub/TR/tr1296.pdf>
- [7] A. D. Jaggard, V. Ramachandran. Relating Two Formal Models of Path-Vector Routing. In *Proc. IEEE INFOCOM 2005*, Mar. 2005. Extended version: Yale Univ. Tech. Report YALEU/DCS/TR-1301, Mar. 2005. <ftp://ftp.cs.yale.edu/pub/TR/tr1301.pdf>
- [8] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771. <http://www.ietf.org/rfc/rfc1771.txt>
- [9] J. L. Sobrinho. Network Routing with Path Vector Protocols: Theory and Applications. In *Proc. ACM SIGCOMM'03*, pp. 49–60, Aug. 2003.
- [10] K. Varadhan, R. Govindan, and D. Estrin. Persistent Route Oscillations in Inter-domain Routing. *Computer Networks*, 32:1–16, 2000.