Distributed Computing manuscript No.
(will be inserted by the editor)

Incentive-Compatible Interdomain Routing

Joan Feigenbaum · **Vijay Ramachandran** · **Michael Schapira**

Received: 6 Nov 2009 / Accepted: 14 Jan 2011

Abstract The routing of traffic between Internet domains, or *Autonomous Systems* (ASes), a task known as *interdomain routing*, is currently handled by the Border Gateway Protocol (BGP) [21]. Using BGP, ASes can apply semantically rich routing policies to choose interdomain routes in a distributed fashion. This expressiveness in routing-policy choice supports domains' autonomy in network operations and in business decisions, but it comes at a price: The interaction of locally defined routing policies can lead to unexpected global anomalies, including route oscillations or overall protocol divergence (see, *e.g.*, [24]). Networking researchers have addressed this problem by devising constraints on policies that guarantee BGP convergence without unduly limiting expressiveness and autonomy (see, *e.g.*, [9, 11]).

In addition to taking this engineering or "protocol-design" approach, researchers have approached interdomain routing from an economic or "mechanism-design" point of view. It is known that lowest-cost-path (LCP) routing can be implemented in an incentive-compatible, BGP-compatible manner [4, 22] but that several other natural classes of policies cannot [3, 6]. In this paper, we present the first example of a class of interdomain-routing policies that is more general than LCP routing and for which BGP itself is both incentive-compatible and guaranteed to converge. We also present several steps toward a general theory of incentive-compatible, BGP-compatible interdomain routing.

Keywords interdomain-routing protocols · BGP · algorithmic mechanism design · routing policies

CR Subject Classification C.2.2 · C.2.6 · G.2.2

This work was supported in part by the U.S. Department of Defense (DoD) University Research Initiative (URI) program administered by the Office of Naval Research (ONR) under grants N00014–01–1–0795 and N00014–04–1–0725. An extended abstract appeared in [5].

J. Feigenbaum was supported in part by ONR grants N00014–01–1–0795, N00014–04–1–0725, and N00014–09–1–0757, by NSF grants 0208972, 0219018, and 0428422, and by HSARPA grant ARO–1756303.

V. Ramachandran was supported in part by ONR grant N00014–01–1–0795 and by NSF grants 0524139 and 0751674; work done in part while at ICSI and at the Stevens Institute of Technology.

M. Schapira was supported by NSF grant 0331548 and by grants from the Israel Science Foundation and the USA-Israel Bi-national Science Foundation; work done in part while visiting Yale University as a graduate student at the Hebrew University of Jerusalem.

J. Feigenbaum *(corresponding author)*
Yale University
New Haven, CT, USA
Tel: +1 (203) 432-6432
Fax: +1 (203) 432-6373
E-mail: joan.feigenbaum@yale.edu

V. Ramachandran
Colgate University
Hamilton, NY, USA
E-mail: vramachandran@colgate.edu

M. Schapira
Princeton University
Princeton, NJ, USA
E-mail: ms7@cs.princeton.edu

1 Introduction

1.1 Interdomain routing

The Internet comprises many separate administrative domains known as *Autonomous Systems* (ASes). Routing occurs on two levels, intradomain and interdomain, implemented by two different sets of protocols. Intradomain-routing protocols, such as OSPF [19], compute routes within a single AS. Interdomain routing, currently handled by the Border Gateway Protocol (BGP) [21], computes routes between different ASes. For many years, interdomain routing has been studied by computer scientists from an engineering or "protocol-design" perspective, and, more recently, it has been studied from an economic or "mechanism-design" perspective as well. Combining algorithmic and economic considerations in the study of interdomain routing is very natural, because the many separate domains that make up the Internet really are independent economic agents that execute a distributed algorithm in order to choose routes.

In their seminal paper [20], Nisan and Ronen gave the following formulation of interdomain routing as a mechanism-design problem: Each AS incurs a per-packet *cost* for carrying traffic, where the cost represents the additional load imposed on the internal AS network by this traffic. To compensate for these incurred costs, each AS is given a *payment* for carrying *transit* traffic, which is traffic neither originating from nor destined for that AS. It is through these costs and payments that consideration of "incentive compatibility" was introduced to the interdomain-routing framework, which, as currently realized by BGP, does not explicitly consider incentives. The goal in [20] was to optimize the use of network bandwidth by routing packets along *lowest-cost paths* (LCPs) and to do so with a *truthful mechanism* that can be computed in *polynomial time*. Nisan and Ronen observed that the Vickrey-Clarke-Groves (VCG) mechanism, well known to be truthful, solves the LCP mechanism-design problem and can be computed in polynomial time.

Many researchers have followed up on Nisan and Ronen's original work, including Feigenbaum, Papadimitriou, Sami, and Shenker [4], who showed that lowest-cost paths and VCG payments could be computed in a "BGP-compatible" fashion, *i.e.*, computed by a distributed algorithm that requires fairly small modifications to the (already universally deployed) Border Gateway Protocol.

Although it was viewed as a step forward in our understanding of the interplay of engineering, algorithmics, and economics in interdomain routing, the work in [4] was by no means a fully satisfactory solution. In particular, one of the valuable features of BGP is that it allows ASes to choose interdomain routes according to semantically rich policies that meet their operational and business requirements; LCP routing is just one example of a valid policy, and, in practice, many ASes do not use it [1]. Thus, it is natural to ask whether more expressive interdomain-routing policies admit truthful, BGP-compatible computation of routes and payments. Previous work on this question was discouraging, producing only negative results, *i.e.*, proofs that various natural classes of policies did not admit such computation; we give pointers to some examples in Subsec. 1.4 below.

In this paper, we continue the work begun in [4] and give the first positive result along these lines by exhibiting a natural class of routing policies strictly more general than LCP for which routes and payments can be computed in a truthful, BGP-compatible manner. *A fortiori*, we exhibit such a class for which BGP itself is guaranteed to converge and is incentive-compatible.

1.2 Routes and Policies

We now give a short and informal explanation of the interdomain-routing problem so that we can state our main results in Subsec. 1.3 below. The problem is presented more formally and in considerably more detail in Sec. 2.

An interdomain-routing instance consists of an *AS graph* G and a set of *routing policies*. In the vertex set of G, there are n *source nodes* $\{1,\ldots,n\}$ and a *destination node* $d \notin \{1,\ldots,n\}$, each representing an AS. Each source AS i has a routing policy, in part given by a real-valued function v_i defined on the set of routes (*i.e.*, simple paths) from i to d in G. The value that source AS i assigns to route R captures the desirability, from i's point of view, of packets' traveling from i to d along R. For example, in an instance of LCP routing, $v_i(R) = -cost(R)$, for all i and R.

In a *path-vector routing protocol*, of which BGP is an example, a confluent tree of routes to d is built up, round by round, as nodes pass *route announcements* to their neighbors. In round 1, the process is begun by the destination d, which announces its existence to its neighbors; each neighbor i of d now has a route to the destination that consists of the one link (i,d). In subsequent rounds, a node i that has a route R to d may announce or *export* R to a neighbor j; if node j does not appear in R, then j at this point has a route through i to d, which we denote by $(j,i)R$. It is important to note that i need not inform j of every route to d that it knows about; if it does not export R to j, then i is said to have *filtered* R with respect to j. As the protocol proceeds, each AS *independently* chooses from the routes that have been announced to it, and the hope is that these choices will converge on a stable tree of routes to d.

From a mechanism-design point of view, a natural goal is a path-vector routing protocol that builds *welfare-maximizing* routing trees, *i.e.*, those for which the sum, over all source nodes i, of the valuations $v_i(R_i)$ is as large as possible, where R_i is the unique route from i to d in the final tree.

1.3 Our Results

In this paper, we show that welfare-maximizing route computation is feasible for routing policies that are more expressive than LCP. We identify three properties that together form a sufficient constraint on policies to permit the computation of welfare-maximizing routes by any path-vector protocol (including BGP):

1. *Policy consistency*: The *next hop* of a route is the source AS's immediate neighbor along that route. An AS has a *next-hop policy* if it chooses among available routes to a destination based solely on the routes' next hops. Next-hop policies capture an essential feature of interdomain routing as it is currently done: An AS cannot control packet forwarding beyond the neighboring AS to which it initially sends the packets. For this reason, many researchers have studied next-hop policies in order to gain insight into the behavior of interdomain protocols [6, 13, 24]. *Policy consistency*, one of the three properties that together comprise our sufficient condition for welfare maximization, is a generalization of next-hop policies. Note that, although next-hop policies have been a natural and fruitful topic of research, they are not sufficient for practical use; it has been shown that uncoordinated and unconstrained local configuration of next-hop policies can produce route instability [13, 24].
2. *Consistent filtering*: As explained in Subsec. 1.2, ASes need not announce to their neighbors all of their known routes to a given destination; instead, an AS i may engage in export filtering with respect to its neighbor j by not announcing a route R to j, *i.e.*, not offering to j the option of sending traffic to d along the route $(j,i)R$. In order to guarantee welfare maximization, we do not allow ASes to engage in arbitrary export filtering. Specifically, the second part of our sufficient condition is that, for all pairs i and j of neighboring ASes, *i filters consistently* with respect to j, meaning that it only filters out routes that it values less than those it announces: If i does not announce route R to j, then $v_i(R) < v_i(Q)$, for all routes Q that i does announce to j.
3. *No dispute wheel*: Gao and Rexford [9] proposed constraints on policies that guarantee route stability without global coordination. They assume that two types of business relationships exist between neighboring pairs of ASes: *customer-provider*, in which one AS purchases connectivity from another, and *peering*, in which two ASes agree to carry transit traffic to and from each other's customers, *e.g.*, to shortcut routes through providers. These relationships accurately represent today's commercial Internet (see [16]), and they naturally induce route preferences. Gao and Rexford formalized these preferences (we review the formalization in Subsec. 4.1) and proved that they induce stable routing if there are *no customer-provider cycles*, *i.e.*, if no AS is an indirect customer of itself. *No dispute wheel*, the third constituent property of our sufficient condition, is a well studied generalization of the Gao-Rexford constraints [13].

Conversely, we show that, if any of these three properties does not hold, the *price of anarchy* [17]—a measure of how far the computed routing tree is from welfare-maximizing—for path-vector routing is unbounded.

One important implication of our sufficient condition is that ASes cannot do any better than executing BGP, provided that valuation functions are non-negative (or, equivalently, that ASes always prefer participating in the resulting routing tree to not participating in it). In such cases, payments are not needed to incentivize AS participation in route computation (see [7]). Indeed, no changes to BGP are needed at all.

In some cases, it may be necessary to incentivize ASes to participate by paying them; for example, "backbone carriers" that are in the business of carrying transit traffic between local networks may need to be paid to do so. Equivalently, valuation functions may assign negative values to some routes. We give a positive result for this case by presenting the first example of a class of policies that is more general than LCP and that admits incentive-compatible and BGP-compatible computation of both routes and payments: next-hop policies that obey the Gao-Rexford conditions. We use the term "BGP-compatible" to mean that the protocol has the same basic structure as BGP and that it is space-efficient, in that it requires only a modest increase to the storage requirement of the standard BGP; the protocol that we present does, however, require the enhancement of BGP with signature and payment capabilities. This is consistent with the use of "BGP-compatible" in [4].

The policy-consistency, consistent-filtering, and no-dispute-wheel properties are presented in detail in Sec. 3. Our algorithm to compute routes and payments is presented in Sec. 4.

1.4 Related Work

The networking-research community's study of BGP was begun by Varadhan *et al.* [24], who showed that completely unconstrained routing policies can result in *protocol divergence*, *i.e.*, protocol executions that do not produce a stable routing tree that all source ASes would continue to use, given the alternative routes available to them. This fundamental observation led to the formulation of stable routing as an NP search problem [13], the formulation of path-vector protocols as a distributed-computational model [12], and the search for constraints on policies, *e.g.*, the Gao-Rexford constraints [9] and generalizations thereof [8], that guarantee

BGP convergence. Griffin *et al.* [11] provide the most general formulation to date of path-vector protocol properties and inherent tradeoffs among them. These and other works by the networking-research community formulate the ASes' policies as ordinal preferences on available routes: If R and Q are routes available to source AS i, then i's policy determines whether it prefers R to Q or *vice versa* but does not assign numerical values to R and Q.

As explained in Subsec. 1.1, Nisan and Ronen [20] formulated the interdomain-routing problem as a mechanism-design problem, and Feigenbaum *et al.* [4] added to this formulation considerations of distributed computation and BGP compatibility. The mechanism-design formulation entailed the generalization from policies that capture ordinal preferences to those that capture cardinal preferences. For the LCP case, welfare-maximizing, incentive-compatible algorithms were obtained in both centralized [20] and distributed [4] computational models, leading naturally to the question addressed in this paper, *i.e.*, whether such algorithms could be found for more general classes of routing policies. This question was answered in the negative for general policy routing [6], "subjective-cost" policy routing [3], "forbidden-set" policy routing [3], and unconstrained next-hop routing [6]. More precisely, it is shown in [6] that welfare-maximizing routing trees for unconstrained next-hop policies can be found by a polynomial-time centralized algorithm but not by an efficient distributed algorithm. As explained in Subsec. 1.3, we present herein the first positive answer to this basic open question from [4, 20].

After our results were presented in preliminary form [5], Feigenbaum, Schapira, and Shenker [7] used our main result to prove that BGP is incentive-compatible even in the presence of coalitions of manipulating nodes. Levin, Schapira, and Zohar [18] showed that following BGP may not be incentive-compatible under the simpler assumption that the Gao-Rexford conditions hold but that following a "secure" version of BGP (in which nodes cannot lie about the presence of nonexistent routes) is incentive-compatible under Gao-Rexford. Goldberg *et al.* [10] showed that the security property in [18] may not suffice to prevent lying during route computation when nodes' utilities are based on the amount of traffic they transit on behalf of others in addition to the route they are assigned in the final routing tree.

2 Technical Preliminaries

We begin this section by formally defining the interdomain-routing problem and providing some useful notation. We then review the Border Gateway Protocol (BGP), the standard protocol used for interdomain routing today.

2.1 Welfare-Maximizing Route Allocation

In the interdomain-routing problem, we are given an AS graph $G = (N, L)$ that describes the network topology. The set of nodes N corresponds to the ASes in the graph. Because routes are computed independently for each destination, without loss of generality, we assume that N consists of n source nodes $\{1, \ldots, n\}$ and a destination node d. The set of links L corresponds to connections between ASes. Let $L^i \subset 2^L$ be the set of all *simple* routes (*i.e.*, routes with no loops) from i to d in G.

An instance $I = (G, \mathscr{P}, \mathscr{V})$ of the *interdomain-routing problem* is defined by an AS graph G, a set of *permitted routes* $\mathscr{P}(i) = P^i \subset L^i$ for each node $i \in [n]$, and the *valuation function* $\mathscr{V}(i) = v_i : P^i \to \mathbb{R}$ of each node. Every set P^i contains the paths in L^i that are not removed from consideration by either i itself or i's neighbors. Every valuation function v_i specifies the "monetary value" of each route $R \in P^i$ from node i. We assume that $v_i(\emptyset) = 0$, *i.e.*, no route is worth nothing, and that, for all pairs of routes R_1 and R_2 through different neighboring nodes, $v_i(R_1) \neq v_i(R_2)$, *i.e.*, there are no ties in valuations.[1] The *routing policy* of each node i is thus captured by v_i and P^i: The only routes considered for i are those in P^i, and preference among these routes is given by the valuation function v_i.

The goal is to allocate to each source node $i \in [n]$ a route $R_i \in P^i$. The resulting *route allocation* $T_d = \{R_1, \ldots, R_n\}$ should form a confluent tree to the destination d. Furthermore, we are interested in route allocations that maximize the "total social welfare" of the nodes, *i.e.*, we want to find an allocation satisfying

$$T_d = \text{argmax}_{T=\{R_1,\ldots,R_n\}} \sum_{i=1}^{n} v_i(R_i).$$

Incentive compatibility is introduced into this problem by attempting to incentivize truthful behavior. In particular, a node i may have to be given some payment $s_i(T_d)$ for its contribution to the routing tree T_d.

We define the *utility function* of each node i, $u_i : \prod_i P^i \to \mathbb{R}$, to be $u_i(T_d) = v_i(R_i) + s_i(T_d)$. Although the global goal is to maximize the total social welfare, every rational node i would only be interested in maximizing its own utility, even if this comes at the expense of not achieving the global goal. An algorithm (protocol) is *truthful* if it is in the best interest of each node to reveal its true valuation function to the algorithm. An algorithm is *incentive-compatible* (with respect to some notion of equilibrium) if it is in the best interest

[1] This assumption is consistent with BGP and the model of interdomain routing in [13]: Because at most one route can be installed in a router's forwarding table to each destination, nodes have some deterministic way to break ties, *e.g.*, based on the next hop's IP address; so, valuations can be adjusted accordingly to match this. However, because only one route per neighbor is considered at a time, ties in valuation are permitted for routes through the same neighboring node.

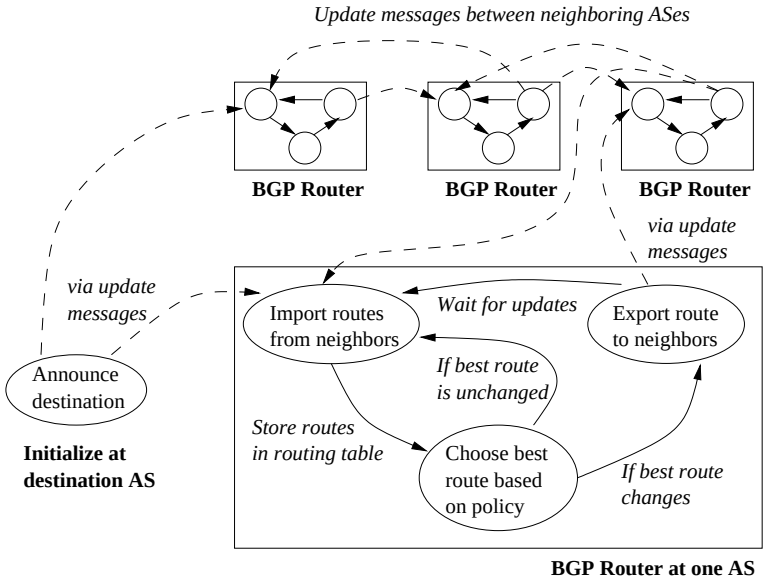

Fig. 1 Route computation using BGP.

of each node to comply with all the algorithm's instructions (with respect to the same notion of equilibrium); compliance includes, but is not limited to, providing truthful input of valuation functions.

A distributed model such as ours poses an inherently different challenge for the design of incentive-compatible mechanisms (see [4, 22]) than a centralized one. This is because the computation is performed by the strategic agents themselves and not by a reliable third party. In this paper, we focus on achieving incentive compatibility in *ex-post Nash equilibrium*, which has been argued to be most appropriate for distributed-mechanism computation [22]; using this concept enables the consideration of several forms of rational manipulation other than lying about inputs (see Subsec. 4.4.2 for a detailed discussion).

We are interested in efficient, distributed, and incentive-compatible welfare-maximizing algorithms for the interdomain-routing problem. We require our algorithms to assume no prior knowledge of the nodes of the topology of the network.

2.2 Routing Notation

First, we present some notation for the representation of routes. A *simple* route is a finite sequence of consecutive links from a source node to the destination node that contains no loops (cycles). *All routes in this paper are simple unless stated otherwise.* We say that node i is in route R (or write $i \in R$) if i participates in one of the links in R.

If R is a route from j (its *source*) to the destination d, and i is a node that is not in R and is adjacent to j in G, we denote by $(i, j)R$ the route that has (i, j) as a first link and then follows R to the destination. We call this an *extension* of R (to i). If j and k are intermediate nodes on a route R, we denote by $R_{[j,k]}$ the subpath of R from j to k.

Throughout this paper, we will consider sub-instances of the interdomain-routing problem obtained by removing one node from the AS graph G. For every node i, we denote by G^{-i} the subgraph of G that contains all nodes in N except i and all links in L except those i participates in. We can now define $I^{-i} = (G^{-i}, \mathscr{V}', \mathscr{P}')$ to be a sub-instance of the original interdomain-routing instance I, in which the *AS* graph is G^{-i} and, for each node $j \neq i$, $\mathscr{P}'(j) = \{R \in \mathscr{P}(j) \mid i \notin R\}$, *i.e.*, any route containing i is removed from the permitted-route set of j, and $\mathscr{V}'(j)$ is $\mathscr{V}(j)$ restricted to the sub-domain $\mathscr{P}'(j)$, *i.e.*, the valuation of a permitted route in I^{-i} is identical to the valuation of that route in I. We denote by T_d^{-i} a welfare-maximizing route allocation for I^{-i}.

2.3 Overview of BGP

The Border Gateway Protocol (BGP) [21] belongs to the family of *path-vector protocols*, the abstract properties of which were studied in [12]. A sketch of how BGP computes routes is shown in Fig. 1. The basic idea is that a routing tree to a given destination is built, hop-by-hop, as knowledge of how to reach that destination propagates through the network. Communication between nodes takes place through *update messages* that announce chosen routes.

The process is initialized when some destination AS d announces itself to its neighbors by sending update messages. Then, each node i iteratively establishes routes to d by:

1. importing, via update messages, routes to d chosen by neighbors[2] and storing the routes in a *routing table*;
2. choosing the best route from i to d (through a neighbor of i) among those available in the routing table based on local routing policy; and
3. if there is a change to i's best route, exporting the newly selected route to all of i's neighbors using update messages.[3]

At any given time, each node's (internally stored) routing table contains the route updates received from its neighbors, and each node is assigned at most one best route based on its policy. (A node may not have a best route if it has not yet received any updates or if its neighbors have *withdrawn* their routes, *e.g.*, because of network failures). We assume that the network is asynchronous; so, it is possible that the network delays the arrival of update messages along selective links.

Path-vector routing has several advantages. First, as the only routes considered are those announced by neighbors, the protocol enforces that the route choices form a confluent tree. Second, each node is able to maintain its autonomy by making its route choice based on local, expressive routing policies. Third, changes in the network due to the addition or subtraction of nodes or links can be announced through update messages, and routers can use alternate routes stored in the routing table to adapt quickly. Fourth, because entire paths are announced, nodes can check for loops and exclude them from routing tables.

Because BGP is currently the standard protocol for Internet interdomain routing, we desire algorithms that are *BGP-compatible*, *i.e.*, that can be implemented using only small modifications to BGP; in particular, we are interested in algorithms that can be implemented using a message structure similar to BGP and that operate without a significant increase in the size or number of messages.

3 A Sufficient Condition for Incentive Compatibility

Path-vector protocols like BGP function much like an iterative game, because, at each step of the protocol, ASes examine the routes chosen by their neighbors and make local decisions as to which routes are best. Convergence to some equilibrium is thus an implicit goal of the protocol. We say that a route allocation is *stable* if no node prefers changing its allocated route to a different route that follows one of its neighbors' allocated routes. A stable route allocation can be regarded as a Nash equilibrium.

Definition 1 A route allocation $T_d = \{R_1, ..., R_n\}$ is stable iff, for every node i,

$v_i(R_i) = \text{argmax}_{\{(i,j)R_j \in P^i \mid (i,j) \in L \wedge i \notin R_j\}} v_i((i,j)R_j)$;

i.e., the route R_i allocated to each node i is the most highly-valued route consistent with the routes allocated to node i's neighbors.

However, a stable route allocation that is reached by local, selfish decision making may not be welfare maximizing. The *price of anarchy* [17], formally defined as follows, measures how bad selfish computation can be.

Definition 2 In an instance I, let

$$W_{\text{selfish}} = \min_{\text{stable } T_d = \{R_1,\ldots,R_n\}} \sum_{i=1}^{n} v_i(R_i)$$

be the minimum total social welfare obtained by a stable routing tree, and let

$$W_{\text{opt}} = \max_{T_d = \{R_1,\ldots,R_n\}} \sum_{i=1}^{n} v_i(R_i)$$

be the maximum total social welfare (over all routing trees). The *price of anarchy* of path-vector routing on I is $\frac{W_{\text{opt}}}{W_{\text{selfish}}}$.

To design a welfare-maximizing path-vector protocol—a distributed protocol in which decisions are made locally and selfishly—we must find conditions under which the price of anarchy is 1. We develop such a condition in the remainder of this section.

3.1 Policy Consistency

Our interdomain-routing problem is an optimization problem in which each node assigns *cardinal* preferences to the different routes, *i.e.*, the magnitude of valuation difference between routes is meaningful. However, BGP's local decision-making finds a stable route allocation based on *ordinal* preferences at each node—although operators can assign integer preferences to each route, only the rank ordering induced by those preferences at each node is relevant to BGP's decision process. This does not suffice, because the value of BGP's allocation, *i.e.*, the sum of the each node's valuation of the route assigned by BGP, can be much lower than that of the optimal route allocation (that maximizes that sum).

Fig. 2 shows an instance for which this is true. Assume $\alpha > 0$. Observe that the unique stable route allocation is $\{1d, 2d, 31d, 431d\}$. However, the optimal route allocation is $\{1d, 2d, 32d, 432d\}$. This allocation will never be chosen by local decisions, because node 3 would prefer routing through node 1, a route that is always available for it to choose. The price of anarchy in this example, $1 + \frac{1}{399}\alpha$, is thus arbitrarily large.

To overcome this problem, we introduce the *policy-consistency* property, which helps to ensure that the optimal route allocation is stable. Informally, a node i is policy-consistent with an adjacent node j if there are no two routes to d starting with (i, j) such that i and j disagree about which route is more preferred.

[2] Some neighbors may refuse to announce particular routes.

[3] Again, nodes may not announce certain routes to certain neighbors.

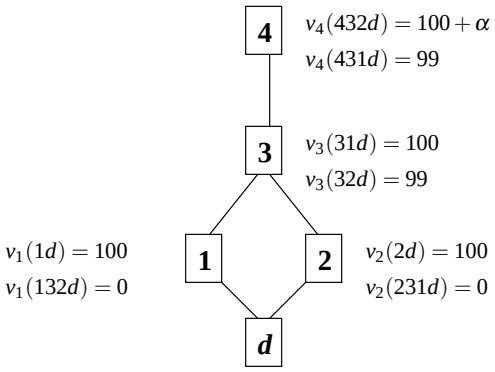

Fig. 2 A routing instance without policy consistency.

Definition 3 Let i and j be adjacent nodes in G. We say that i is *policy-consistent* with node j iff, for every two routes Q and R permitted at j with extensions permitted at i (*i.e.*, $\{Q,R\} \subset P^j$ and $\{(i,j)Q,(i,j)R\} \subset P^i$):

$$\text{if } v_j(Q) > v_j(R), \text{ then } v_i((i,j)Q) > v_i((i,j)R).$$

Definition 4 An instance is policy-consistent ("policy consistency holds") iff, for every two adjacent nodes i and j, i is policy-consistent with j.

One common example of policy consistency is *next-hop valuations*, in which nodes only consider the immediate neighbor along a route:

Definition 5 For node $i \in [n]$, define $\text{neighbors}(i) = \{j \in N \mid (i,j) \in L\}$, *i.e.*, the set of nodes adjacent to i. If $R' \in L^j$ is a simple route from source node j, and $R = (i,j)R'$ is its extension to node i, then define the *next hop* on R to be $\text{next}(R) = j$; *i.e.*, the next hop of a route is the source node's neighbor on that route.

Definition 6 Node $i \in [n]$ has a *next-hop valuation function* v_i iff there exists a function $f_i : \text{neighbors}(i) \to \mathbb{R}_{\geq 0}$ such that, for every route $R \in P^i$, $v_i(R) = f(\text{next}(R))$; *i.e.*, the valuation of a route depends only on its next hop.

If all nodes have next-hop valuation functions, we say that "the instance uses next-hop policies." Note that, while appearing simple, next-hop policies are semantically rich enough to permit global routing instability (see Subsec. 3.3).

Another example of policy-consistent valuations are *metric-based valuations* (defined in [12]):

Definition 7 Let $\delta : L \to \mathbb{R}_{>0}$ be a positive real-valued function that specifies the "length" of each link (a "metric" function). A valuation function v that is based on δ is one in which $v(Q) > v(R)$ iff $\sum_{l \in Q} \delta(l) < \sum_{l \in R} \delta(l)$.

It is easy to see that, if all nodes' valuations are based on the same underlying metric function δ, then the network is policy-consistent. In particular, if $\delta(l) = 1$ for every link l, then this is precisely the well known shortest-path-routing problem.

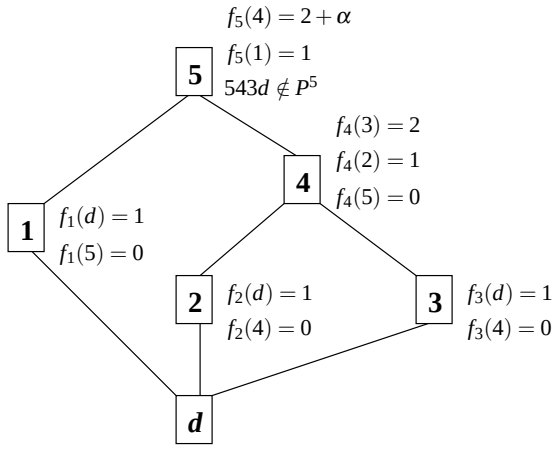

Fig. 3 Next-hop policies without consistent filtering. (Because all nodes have next-hop valuation functions, the valuations here are simply written as mappings from neighbors to values, consistent with Def. 6.)

3.2 Consistent Filtering

In traditional formulations of interdomain routing, nodes are allowed to *filter* routes arbitrarily when exporting updates to or importing updates from neighbors, *i.e.*, nodes can arbitrarily remove paths from consideration (restricting P^i).

Arbitrary filtering is rarely considered in the welfare-maximizing formulation of interdomain routing. Like the lack of policy consistency, arbitrary filtering can make the price of anarchy unbounded, because a node may value a route that is filtered by a neighbor much more than any other route available. This is the case in Fig. 3, an instance with next-hop policies (which are policy-consistent) and only one stable route allocation. (Again, assume $\alpha > 0$.) Although node 5 generally prefers routing through node 4, the path $543d$ is filtered. If node 4 chooses to route through node 2, node 5 can route through node 4, and this leads to the optimal route allocation, $\{1d, 2d, 3d, 42d, 542d\}$. However, this allocation is not stable, because node 4 prefers routing through node 3. This prevents node 5 from routing through node 4, causing node 5 to choose the only available route remaining, which goes through node 1. Thus the unique stable route allocation is $\{1d, 2d, 3d, 43d, 51d\}$. The price of anarchy in this example is $1 + \frac{1}{6}\alpha$, which can grow arbitrarily large as $\alpha \to \infty$.

In order to achieve our objective of welfare maximization, we require that nodes not filter routes arbitrarily. If a node filters a route, it must value that route less than any route that is not filtered—this is called *consistent filtering*.

Definition 8 Node i *filters consistently* with respect to (adjacent) node j iff any route R that is filtered from i to j (R is permitted at i but its extension to j is simple but not permitted at j, *i.e.*, $R \in P^i$, $(j,i)R \in L^j$, and $(j,i)R \notin P^j$) is valued less highly at i than any route not filtered from i to j, *i.e.*, $v_i(R) < v_i(Q)$ for all routes $Q \in P^i$ such that $(j,i)Q \in P^j$.

We say that an instance "filters consistently" if every node filters consistently with respect to every other adjacent node.

Remark 1 The *isotonicity* property studied by Sobrinho in [23] for its relationship to optimal routing essentially combines policy consistency and consistent filtering.

3.3 Robustness and Dispute Wheels

Although BGP attempts to find a stable route allocation, it may not always do so; the hope is that the distributed, independent route choices over time approach a confluent routing tree that does not keep changing. Unfortunately, it has been shown that anomalous interaction of local policies can induce protocol oscillation, causing routes to change indefinitely [24]. Therefore, an important desideratum for path-vector protocols like BGP is convergence:

Definition 9 We say that a path-vector protocol *converges* on an instance of the interdomain-routing problem if, for every initial route allocation and for every sequence of nodes taking turns updating, there exists some time after which a stable route allocation (see Def. 1 is reached, *i.e.*, the route chosen by each node never changes.

Even though BGP may converge when all nodes and links are functioning, it may diverge after failures introduce topology changes. We call the desirable property of guaranteed convergence even in the presence of failures *robustness*, which is formally defined as follows:

Definition 10 An instance of the interdomain-routing problem is *robust* iff, for every sub-instance obtained by removing any set of nodes and links from the original graph, there exists a unique stable route allocation to which a path-vector protocol converges from any initial route allocation.

Previous work has studied the effects of routing policies on robustness, and there is an inherent trade-off in achieving the desired autonomy and policy expressiveness at a local level and robustness at a global level [11]. Early work conjectured that only shortest-paths routing might be provably robust [24]. However, Griffin, Shepherd, and Wilfong [13] presented a sufficient condition on policies that guarantees robust convergence while allowing policies broader than shortest-path routing.

This condition is called *no dispute wheel*. A dispute wheel is essentially a representation of a set of nodes and their routing policies (*i.e.*, ordinal preferences on paths) that induce a routing anomaly. Any instance on which BGP diverges or nondeterministically converges contains a dispute wheel; without a dispute wheel, BGP converges to a unique, stable route allocation on the instance and every sub-instance.

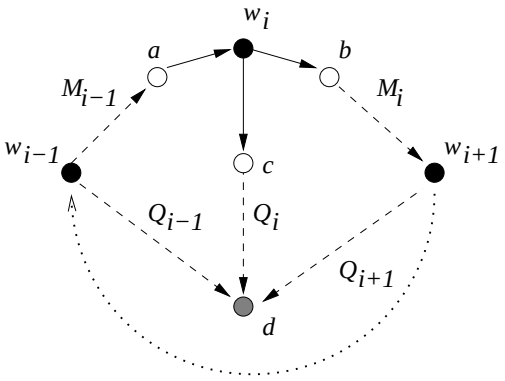

Fig. 4 A dispute wheel. Dashed lines represent routes while solid lines represent edges; the black nodes are pivots.

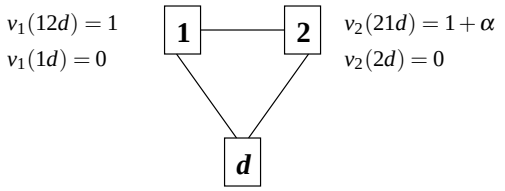

Fig. 5 A routing instance with a dispute wheel.

Definition 11 An instance contains a *dispute wheel* (see Fig. 4) iff there exists a sequence of *pivot nodes* $w_0, \ldots, w_{m-1}$ such that for all $0 \leq i \leq m$ (interpret subscripts modulo m):

1. there exists a *spoke route* $Q_i \in P^{w_i}$;
2. there exists a *rim route* M_i from w_i to w_{i+1} such that $M_i Q_{i+1} \in P^{w_i}$; and
3. $v_{w_i}(Q_i) < v_{w_i}(M_i Q_{i+1})$.

Fig. 5 shows a routing instance (DISAGREE, from [13]) with policies that induce a dispute wheel. This instance has two stable route allocations: $\{1d, 21d\}$ and $\{12d, 2d\}$. Because the network is asynchronous, the timing of update messages may cause BGP to converge to either of these solutions or oscillate between them [13]. This anomaly is characterized by the dispute wheel with pivot nodes 1 and 2, each having the direct route to d as a spoke route and the edge to the other pivot as a rim route. The price of anarchy in this example is $(1+\alpha)$, which can be arbitrarily bad.

The absence of a dispute wheel is, in fact, the broadest-known sufficient condition for stability and robustness. In the design of an incentive-compatible routing mechanism, we want to ensure that our BGP-based routing algorithm does reach a stable tree in some equilibrium. We now show that, in the presence of policy consistency and consistent filtering, having no dispute wheel in the valuations is equivalent to robustness. We note that this is the first known necessary and sufficient condition for robustness.

Theorem 1 *A policy-consistent instance that filters consistently is robust iff it contains no dispute wheel.*

Proof The sufficient condition (the "if" direction) is a specific case of the main theorem in [13], which states that an instance containing no dispute wheel is robust.

To prove the necessary condition (the "only if" direction), we show that any policy-consistent instance that filters consistently containing a dispute wheel must also contain a *dispute ring*, which is a dispute wheel containing no repeated nodes on its rim. Feamster, Johari, and Balakrishnan [2] showed that an instance containing a dispute ring is not robust.

Assume we have an instance containing a dispute wheel. If the dispute wheel is a dispute ring, we are done; therefore, assume that node x appears at least twice on the rim, and let M_i and M_j be two of the rim routes containing x. If x appears as a pivot node, then let $x = i+1$ or $x = j+1$ as appropriate, so the rim route leads to x; we note that $x = i+1$ implies that $M_{i[x,i+1]}$ is empty (likewise for $x = j+1$ and $M_{j[x,j+1]}$). If x is not a pivot, assume the next hops of x on M_i and M_j are not the same; if they are the same, take the next hop as the repeated node x being considered. Without loss of generality, let $v_x(M_{i[x,i+1]}Q_{i+1}) < v_x(M_{j[x,j+1]}Q_{j+1})$. Then, for each node $y \in R_{i[i,x]}$ (starting with the node closest to x), consistent filtering implies that $M_{i[y,x]}M_{j[x,j+1]}Q_{j+1} \in \mathscr{P}^y$, and policy consistency then implies that $v_y(M_{i[y,x]}M_{j[x,j+1]}Q_{j+1}) > v_y(M_{i[y,i+1]}Q_{i+1})$. Therefore, we can contract the dispute wheel at x by replacing the rim routes $M_i, M_{i+1}, \ldots, M_j$ with the single rim route $M_{i[i,x]}M_{j[x,j+1]}$; this removes one appearance of any nodes appearing in both M_i and M_j, in particular, the second appearance of x. Repeatedly applying this procedure generates a dispute ring. □

3.4 Local and Global Optimality

The above subsections presented negative results for welfare-maximizing routing when any one of three properties—policy consistency, consistent filtering, or robustness—is absent. We now turn to a positive result derived from the interesting relationship among these three properties. Recall that, if an instance is robust, then it has a unique stable route allocation. The following theorem states that, if all three properties hold, then this unique route allocation is globally optimal (*i.e.*, it maximizes the total social welfare).

Theorem 2 *In any robust, policy-consistent instance that filters consistently, there exists a unique stable route allocation T_d that is optimal (welfare maximizing)*, i.e.,

$$T_d = \text{argmax}_{T=\{R_1,\ldots,R_n\}} \sum_{i=1}^{n} v_i(R_i).$$

Proof We will use the following lemma in the proof of the theorem.

Lemma 1 *If $T = \{R_1, \ldots, R_n\}$ is a globally optimal allocation for an instance with policy consistency and consistent filtering, then T is stable.*

Proof (of Lem. 1) Assume by contradiction that T is not stable; then, by Def. 1, there are two adjacent nodes i and j such that

$$v_i(R_i) < v_i((i,j)R_j). \tag{1}$$

Let k be a node such that $\text{next}(R_k) = i$. Because k is policy consistent with i, and because i and k filter consistently, (1) implies that

$$v_k(R_k) < v_k((k,i)(i,j)R_j);$$

by induction, this is also true for every node k' with next hop k in T, *etc.*, so that every node u routing through i in T prefers the route $R_{u[u,i]}(i,j)R_j$ to R_u. Note that we have identified a route allocation in which i and all nodes routing through i are strictly better off, and all nodes not routing through i are unaffected. This new allocation has higher total social welfare than T; however, this contradicts the optimality of T. Thus, our assumption must be incorrect, and T must therefore be stable. □

We are now ready to prove Thm. 2. Let T be some optimal route allocation. By Lem. 1, because of policy consistency and consistent filtering, T is stable. However, because of robustness, there is only one stable allocation T_d [13]. Therefore, $T = T_d$, and the unique stable allocation is also optimal. □

A locally optimal route allocation is one in which nodes are assigned their most valued routes. Such a route allocation would best satisfy selfish nodes interested in maximizing their own—as opposed to the total—welfare.

Definition 12 A route allocation $T_d = \{R_1, \ldots, R_n\}$ is *locally optimal* iff, for every node i, $R_i = \text{argmax}_{R \in P^i} v_i(R)$, *i.e.*, every node i is allocated its highest-valued route.

The following theorem shows that the combination of robustness, policy consistency, and consistent filtering ensures not only global optimality but local optimality as well.

Theorem 3 *In a robust, policy-consistent instance that filters consistently, any globally optimal, stable route allocation is also locally optimal.*

Proof We follow a proof technique of [23]. Consider a node $m \in N$. Let $R = u_k u_{k-1} \ldots u_i \ldots u_0$ be some simple route in P^{u_k}, such that $u_k = m$ and $u_0 = d$. By induction, we show for each $u_i \in R$ that S_i, the route for node u_i in the globally optimal allocation T_d, is at least as good as $R_i = R_{[u_i,d]}$. When $i = m$ we get that S_m is at least as good as R; because R and m were chosen arbitrarily, we prove local optimality of T_d.

Base case. $i = 0$. The induction hypothesis is trivially true, because the only route is the empty one.

Induction step. Assume that the induction hypothesis is true for u_{i-1}, *i.e.*,

$$v_{u_{i-1}}(S_{i-1}) > v_{u_{i-1}}(R_{i-1} = u_{i-1}u_{i-2}\dots d). \quad (2)$$

Note that u_i does not lie on R_{i-1}, or R would not be simple.

Case I. Assume $u_i \notin S_{i-1}$. Then extend S_{i-1} and R_{i-1} along the edge (u_i, u_{i-1}). Consistent filtering ensures that $(u_i, u_{i-1})S_{i-1} \in P^{u_i}$; thus, from (2) and policy consistency, we have

$$v_{u_i}((u_i, u_{i-1})S_{i-1}) > v_{u_i}(R_i = u_i u_{i-1} u_{i-2}\dots d). \quad (3)$$

T_d is stable; so, S_i is at least as good as any other route at u_i; in particular,

$$v_{u_i}(S_i) > v_{u_i}((u_i, u_{i-1})S_{i-1}). \quad (4)$$

Combining (3) and (4) gives

$$v_{u_i}(S_i) > v_{u_i}(R_i),$$

which is the induction statement for u_i.

Case II. Assume $u_i \in S_{i-1}$. In this case we cannot use the policy consistency argument as in case I, because extending S_{i-1} to u_i creates a loop.

Suppose the induction statement for u_i is not true: $v_{u_i}(R_i) > v_{u_i}(S_i)$. We can then create a dispute wheel of size $m = 2$, following Def. 11, in which the pivot nodes are $w_0 = u_{i-1}$ and $w_1 = u_i$. Let the spoke route from w_0 be $Q_0 = R_{i-1}$ and let the spoke route from w_1 be $Q_1 = S_{i-1[u_i,d]}$ (recall the case-II assumption that $u_i \in S_{i-1}$). Let the rim route M_0 from w_0 to w_1 be $S_{i-1[u_{i-1},u_i]}$, and let the rim route M_1 from w_1 to w_0 be the edge (u_i, u_{i-1}).

The first condition in Def. 11 is satisfied because Q_0 and Q_1 are permitted routes by assumption. The second condition is satisfied because $M_0Q_1 = S_{i-1}$, which is permitted because it is the globally allocated route for u_{i-1}, and $M_1Q_0 = R_i$, which is permitted by assumption. The third condition is satisfied for w_0 because $v_{w_0}(M_0Q_1) = v_{u_{i-1}}(S_{i-1}) > v_{u_{i-1}}(R_{i-1}) = v_{w_0}(Q_0)$ by the induction hypothesis for u_{i-1}. Finally, note that $S_i = S_{i-1[u_i,d]} = Q_1$, because the globally optimal route allocation is consistent (and $u_i \in S_{i-1}$); therefore, our assumption that the induction statement is not true exactly translates to the third dispute-wheel condition for w_1: $v_{w_1}(M_1Q_0) = v_{u_i}(R_i) > v_{u_i}(S_i) = v_{w_1}(Q_1)$.

The presence of a dispute wheel contradicts our assumption of robustness because of Thm. 1; this must mean the induction statement is indeed true for u_i. (Recall there are no ties in valuations.) □

Remark 2 Global and local optimality also hold for sub-instances. If any of the three properties (robustness, policy consistency, consistent filtering) hold in an instance, they also hold in all sub-instances. Thus, all sub-instances of an instance satisfying the requirements of Thm. 2 and Thm. 3 also satisfy the requirements of these theorems.

The above result shows that BGP, *with no modifications or payments*, converges to the unique welfare-maximizing routing tree when nodes consistently filter and valuations are policy-consistent and do not induce a dispute wheel. In other words, nodes cannot do any better than executing BGP, except in the case when nodes would prefer not to participate in the routing tree (*i.e.*, some node has a negative valuation for the route it is allocated in the tree). Feigenbaum, Schapira, and Shenker [7] used this result (from a a preliminary version of this paper [5]) to prove that, for a subclass of dispute-wheel-free, policy-consistent, nonnegative valuations, BGP is incentive-compatible in collusion-proof ex-post Nash equilibrium. We deal with the case of negative valuations in the next section by presenting a modification to BGP that computes payments required to incentivize nodes to behave truthfully.

4 A BGP-Compatible, Incentive-Compatible Algorithm for Negative Valuations

Payments may be required to incentive participation if nodes have negative valuations on routes. We now present an incentive-compatible, BGP-compatible algorithm to compute routes and payments in the case of valuation functions that may assume negative values. As explained in Sec. 1, we use the term *BGP-compatible* to mean that the algorithm has the same basic structure as BGP and that it is "space-efficient," in that it requires only a modest increase to the storage requirement of the original BGP. (This is consistent with use of the term in [4].)

Previously, a positive result in the presence of negative valuations was known only for LCP policies [4]. Here we expand the class of policies that admits a positive result: next-hop routing that obeys the Gao-Rexford conditions for global stability.

4.1 Policies for the Commercial Internet

Next-hop policies (Def. 6) are those in which AS route preferences are based only on the neighbor to which packets are forwarded. It has been studied in the interdomain-routing literature, because it captures the property that, in today's standard IP forwarding, an AS does not control a packet once it has been delivered to a neighboring AS. Although it is a conceptually simple class of policies, it is sematically rich enough to permit global routing instability [24], and it does not permit incentive-compatible, BGP-compatible, welfare-maximizing routing [6]. Thus, in this paper, we start with next-hop routing and add additional restrictions to obtain a positive result. The additional restrictions that we add have been studied previously; none is introduced here for the first time.

One well studied set of constraints assumes that a business hierarchy underlies the AS graph and that policies are based on the economic nature of this hierarchy. Huston's study of the commercial Internet [16] suggests two types of business relationships that characterize AS inter-connections: Pairs of neighboring nodes have either a *customer-provider* or a *peering* relationship. Customers pay their provider nodes for connectivity—access to Internet destinations through the provider's links and announcement of customer destinations to the rest of the Internet. Peers are nodes that find it mutually advantageous to exchange traffic for free, *e.g.*, to shortcut routes through providers. A node can be in many different relationships simultaneously: It can be a customer of one or more nodes, a provider to others, and a peer to yet other nodes. These agreements are assumed to be longer-term contracts that are formed because of various external factors, *e.g.*, the traffic pattern between two nodes.

Intuitively, these business relationships naturally induce routing policies. Gao and Rexford [9] formally modeled these relationships and policies with the following three conditions.

No customer-provider cycles: Let G_{CP} be the digraph with the same set of nodes as the AS graph G and with a directed edge from every customer to its provider. We demand that there be no directed cycles in this graph. If this requirement is met, we say that "the AS graph contains no customer-provider cycles." This demand is a natural economic assumption, because, if there is a cycle in G_{CP}, then a node is indirectly its own provider.

Prefer customers to peers and providers: A *customer route* is a route in which the next-hop AS is a customer. *Provider* and *peer routes* are defined similarly. We require that nodes always prefer (*i.e.*, assign a higher value to) customer routes over peer and provider routes. This has an economic justification given the financial agreements underlying the business relationships: Providers want to maintain traffic flow along links for which they are paid, and customers want traffic along routes they announce (otherwise, they would not announce them).

Provide transit services only to customers: Nodes do not always carry *transit traffic*—traffic that originates and terminates at hosts outside the node. An AS is obligated (by financial agreements) to carry transit traffic to and from its customers, but it does not carry transit traffic among only providers and peers, because it receives no payment for doing so. Therefore, we require that nodes announce only customer routes to their providers and peers but announce *all* of their routes to their customers.

Using the terminology and notation of Sec. 2, we formally define the Gao-Rexford conditions as follows:

Definition 13 The *Gao-Rexford conditions* hold iff the AS graph contains no customer-provider cycles, and, for all nodes $i \in [n]$, the following hold for all pairs of nodes $\{j,k\} \subset \mathrm{neighbors}(i)$ and for all pairs of routes $\{R_j, R_k\} \subset P^i$ such that $\mathrm{next}(R_j) = j$ and $\mathrm{next}(R_k) = k$:

1. If j is a customer and k is not, then $v_i(R_j) > v_i(R_k)$.
2. If neither j nor k is a customer, then $(j,i)R_k \notin P^j$ and $(k,i)R_j \notin P^k$, because i does not export R_k to j or R_j to k. If j is a customer, then, whatever i's relationship to k, R_j is exported to k, and R_k is exported to j. Thus, if j is a customer, $(k,i)R_j \in P^k$ if permitted by k, and $(j,i)R_k \in P^j$ if permitted by j.

The Gao-Rexford conditions limit the types of routes available to ASes. Specifically, if a node i receives a route announcement from one of its customers, then every AS on that route is a provider of its next hop on that route.[4] This is because ASes export only customer routes to their providers. If R is a customer route at i with next hop j, then j must have announced the route to i, its provider; thus, R must be a customer route at j. This argument can then be applied inductively along R, implying that R consists entirely of provider-customer links.

It was proven in [9] that, if all nodes obey the Gao-Rexford conditions, enforced naturally by Internet economics, BGP predictably converges to a stable routing tree, even after node and link failures. Later work [8] showed that the Gao-Rexford conditions imply the no-dispute-wheel property introduced by [13] and reviewed earlier in Subsec. 3.3. In addition, the Gao-Rexford conditions enforce consistent filtering: All routes are announced to customers, and only non-customer routes (which are valued less than customer routes) are filtered to peers and providers. These remarks—along with the property that next-hop policies are policy-consistent (see Subsec. 3.1)—prove the following proposition, which states that the policies outlined in this subsection satisfy the requirements of Thms. 1–3.

Proposition 1 *An instance with next-hop policies that obey the Gao-Rexford conditions is policy-consistent, filters consistently, and has no dispute wheel.*

4.2 The Algorithm

The following algorithm is an extension to BGP that computes routes and payments for incentive-compatible, welfare-maximizing routing when policies are next-hop based and obey the Gao-Rexford conditions.

4.2.1 High-Level Overview

The mechanism implemented by the algorithm belongs to the Vickrey-Clarke-Groves (VCG) family of mechanisms,

[4] This property is similar to the *valley-free* property described in [8].

just as previous routing mechanisms have (*e.g.*, the algorithms in [20] and [4]). The payments issued by this mechanism essentially compensate each node for its contribution to the routing tree; intuitively, this can be determined by considering the best routing tree available when that node is not present in the AS graph (*i.e.*, when it refuses to carry any transit traffic). Although this can trivially be done by running BGP n additional times—on the n AS graphs obtained by removing each of n nodes from the original—our algorithm accomplishes this by modifying a single run of BGP. The payments and their analysis are discussed more fully in Subsec. 4.3 below.

The algorithm computes best routes in essentially the same manner as BGP, but it adds extra information to update messages so that nodes can compute the mechanism's payments once the routes have been determined. This information is also stored in nodes' routing tables, requiring one extra bit of storage for every transit AS on an imported route. These bits are used to determine the next hop of the best *k-avoiding route*—the best route in I^{-k}, *i.e.*, for the instance in which node k does not participate—for every transit node k on the best route for each node in I. These next hops are used directly in computing payments and can be stored using one extra row in the routing table, denoted L_i below. The extra bit per transit node in each row of the routing table and the extra row used to store the next hops require a constant-factor increase in the space complexity of the original BGP; a similar amount of extra storage was used by the algorithm described in [4] for lowest-cost-path routing and satisfies the condition of "BGP compatibility" put forth in that paper.

The dynamics of the algorithm can be summarized as follows. Computation of best routes and k-avoiding next hops is triggered when nodes receive update messages, just as in BGP (see Subsec. 2.3). Update-message processing is divided into two cases: (I) the message is from the most valued neighbor that has yet sent a message, in which case the route contained in the message is chosen as the best route; and (II) the message is not from the most valued neighbor that has yet sent a message, in which case the extra bits in the message are used to update the choices of the best k-avoiding next hops. Unlike BGP, if node x chooses node y as its next hop, an update message is still sent from x back to y; this extra message is used to convey availability to y of k-avoiding routes through x and is processed using case (II).

4.2.2 Input and Output

Input: An instance of the interdomain-routing problem with next-hop policies obeying the Gao-Rexford conditions. As in Def. 6, we assume that each node $i \in [n]$ has a function $f_i : \text{neighbors}(i) \to \mathbb{R}_{\geq 0}$, such that $v_i(R) = f_i(\text{next}(R))$.

Output: A route allocation $T_d = \{R_1, \ldots, R_n\}$ that forms a confluent tree to d, such that the tree maximizes the total social welfare, *i.e.*,

$$T_d = \text{argmax}_{T=\{R'_1,\ldots,R'_n\}} \sum_{i=1}^{n} v_i(R'_i),$$

and a payment s_i to each node i.

4.2.3 Communication and Storage

Structure of Update Messages: An update message m sent by node i contains a route $R_m \in P^i$ and, for every transit node $k \in R_m$ ($k \notin \{i,d\}$), a bit $B_m(k)$. $B_m(k) = 1$ if i has, in its routing table, a k-avoiding route to d, *i.e.*, some permitted route $R \in P^i$ such that $k \notin R$. These bits are used to populate the list L_i, defined below, that is used to compute the mechanism's payments.

Storage at Each Node: Each node i has a routing table Y_i indexed by neighbors of i. If $j \in \text{neighbors}(i)$, then let $Y_i(j)$ be the most recent update message sent by node j, so that at most one announced route is stored per neighbor. Initially, $Y_i(j) = \emptyset$ for all j. Each node i also has a list L_i, defined as follows: Assume the current best route at i is R_i; if $k \in R_i$ is a transit node ($k \notin \{i,d\}$), then $L_i(k) = \text{next}(R')$, the next hop on the best k-avoiding route R' in i's routing table. $L_i(k)$ will be used, at the end of the algorithm, to compute the component of the payment to node k that is attributable to node i, denoted s^i_k. Fig. 6 shows an example of the storage at each node.

4.2.4 Execution of the Algorithm

Start: AS d sends update message $m = (d, \emptyset)$ to all neighbors.

Update-Message Processing: Let $m = (R_m, B_m)$ be the update message received at node i from $j \in \text{neighbors}(i)$. If $(i,j)R_m \notin P^i$ and $\text{next}(R_m) \neq i$ (the route is not permitted), then discard the message. Otherwise, $(i,j)R_m \in P^i$ or $\text{next}(R_m) = i$, and the update message should be stored in the routing table so that $Y_i(j) = (R_m, B_m)$.

(Case I) Suppose that the update message is received from the most valued neighbor so far, *i.e.*, $\text{next}(R_m) \neq i$ and

$$f_i(j) = \max_{\{j' \in \text{neighbors}(i) | Y_i(j) \neq \emptyset\}} f_i(j').$$

Then, either R_m is a new best route to d (*i.e.*, R_m is the new R_i) or the neighbor exporting R_m has an updated bit vector B_m. Reset L_i to empty and, for each $k \in R_m$ such that $k \neq d$, do the following to repopulate L_i: If $B_m(k) = 1$, then set $L_i(k) = j$ (if node j has a k-avoiding route, then it is recorded as the best next hop when k cannot be used for transit); if $B_m(k) = 0$ or $k = j$, then:

Dest.	Valuation	$L_5(4)=1$	$L_5(3)=4$	$\rightarrow L_z$: best k-avoiding next-hop ASes for transit k on the best route
d	$v_5(43d)=2+\alpha$	AS 4	AS 3 $B_4(3)=1$	$\rightarrow R_4$, the route chosen by neighbor AS 4; *the current best route* $\rightarrow B_4$, the bit vector sent with update from neighbor 4
d	$v_5(1d)=1$	AS 1		$\rightarrow R_1$, the route chosen by neighbor AS 1 $\rightarrow B_1$, the bit vector sent with update from neighbor 1, is empty

Fig. 6 An example routing table for node 5 in Fig. 2 using the algorithm from Subsec. 4.2. (Assume in Fig. 2 that no routes are filtered and that all links are customer-provider links, where the AS with the greater number is the provider.)

1. Let $A = \text{neighbors}(i) - \{j\}$ and let

   $a = \text{argmax}_{\{a' \in A | Y_i(a') \neq \emptyset\}} f_i(a')$

   be the most valued node in A. Let $(R_a, B_a) = Y_i(a)$ be the routing-table entry for a.
2. If $k \notin R_a$, then set $L_i(k) = a$.
3. If not, $k \in R_a$. If $B_a(k) = 1$, then set $L_i(k) = a$.
4. If $L_i(k)$ has still not been set, then repeat at (1) with $A = A - \{a\}$. Discontinue repeat if $A = \{a\}$, *i.e.*, if there would be no nodes left in A.

Finally, set $R_i = (i, j)R_m$. (Because j is the most valued neighbor to send an update so far, its route is the best route so far.)

(Case II) Suppose that the update message's source j is not the most valued neighbor that has communicated so far, *i.e.*, $\text{next}(R_m) = i$ or

$$f_i(j) \neq \max_{\{j' \in \text{neighbors}(i) | Y_i(j) \neq \emptyset\}} f_i(j').$$

For each current transit node $k \in R_i$ ($k \notin \{i, d\}$), set $L_i(k) = j$ if j has a k-avoiding route and j is more valued than $L_i(k)$, the current best k-avoiding next hop; *i.e.*:

1. $f_i(j) > f_i(L_i(k))$; and either
2a. $k \in R_m$ and $B_m(k) = 1$; or
2b. $k \notin R_m$.

If any changes were made to L_i in either of the cases above (including any time case I was triggered), then send update messages $m' = (R_i, B'_m)$ to all neighbors of i, where $B'_m(k) = 1$ if $L_i(k) \neq \emptyset$ (there is a k-avoiding route known) and $B'_m(k) = 0$ if $L_i(k) = \emptyset$ (there is no k-avoiding route known). (If R_i is a non-customer route and neighbor n is also a non-customer, then the update message $(\emptyset, \emptyset)$ should be sent to comply with the Gao-Rexford conditions, implying a withdrawal of the previous route. Note that, in Lem. 2 below, we prove that a withdrawal will never happen.)

Payment Computation: Once the algorithm converges, each node i can compute the *payment component* $s^i_k = f_i(\text{next}(R_i)) - f_i(L_i(k))$ for every transit node $k \in R_i$ ($k \notin \{i, d\}$), which is the component of the total payment to k that is attributable to i. The total payment to each node k is then the sum of all the payment components to k: $s_k = \sum_{i \neq k} s^i_k$.

In the following subsection we analyze the algorithm presented above and discuss its properties.

4.3 Convergence, Optimality, and BGP Compatibility

We now show that, on instances that obey the Gao-Rexford conditions, and where all source nodes have next-hop policies, the following properties hold for the algorithm in Subsec. 4.2:

1. It converges, *i.e.*, there exists a time after which route choices have settled on a unique, stable route allocation (in the sense of Def. 1).
2. It outputs a route allocation that optimizes the social welfare.
3. It is BGP-compatible, in the sense that it entails only a constant-factor increase in space complexity over BGP. To show this we establish that our algorithm requires only slight modification to BGP messages (with a limited increase in message size).

4.3.1 Convergence

Theorem 4 *The algorithm in Subsec. 4.2 is robust on instances with next-hop policies that obey the Gao-Rexford conditions,* i.e., *it converges in finite time to a unique, stable route allocation.*

Proof This theorem follows from more general results discussed in Sec. 3. In the algorithm, routes are chosen exactly as they are in BGP; by Prop. 1 and Thm. 1, the theorem statement is true for BGP on the instances being considered. However, the update-message dynamics of the algorithm differ from BGP in two ways, and we must reconcile these differences for the result to apply to the algorithm.

The first difference is the lack of withdrawal messages in the algorithm. In both the algorithm and in BGP, an update message is sent from i to j when a new best route is chosen at i. In BGP, this message either (1) contains the new route (if i can export its choice to j), or (2) contains a withdrawal (if i cannot export its choice to j). In the algorithm, (1) still occurs, but (2) does not. However, the following lemma shows that this is irrelevant: For valuations obeying the Gao-Rexford conditions, withdrawal messages are never sent.

Lemma 2 *If, at some time, node a sends node i an update message* (R_m, B_m) *such that* $R_m \neq \emptyset$, i.e., *node a exports a route to node i, and we assume there are no failures, then*

at any future time, there will exist a route R_a in i's routing table, such that $\text{next}(R_a) = a$.

Informally, this lemma means that once a node exports a usable route to a neighbor (where "usable" means allowed by the Gao-Rexford conditions), any route chosen by the node will be a usable route for that neighbor. Therefore, route withdrawals are unnecessary; routes are only replaced with new (usable) routes.

Proof (of Lem. 2) Changes to the routing table are update-driven. A change, due to a new update or withdrawal, will only be sent if a switches from R_m to some other route R_a. We must show that, in this case, an update message with R_a is sent to i, and a withdrawal is not sent.

If a is a provider of i, then a will export R_a to i. Therefore, we can assume, without loss of generality, that a is a peer or customer of i; then R_m must be a customer route of a, or it would not have been sent to i. If a switches to R_a because $v_a(R_a) > v_a(R_m)$, then R_a must also be a customer route, and it will be exported to i. If not, then R_m must have been withdrawn. (If it was replaced, next-hop policies dictate that $v_a(R_a) = v_a(R_m)$, and that route will be exported to i.) In this case, its customer $c = \text{next}(R_m)$ switched to a route that was filtered; but, this new route must be a non-customer route at c. Because it is less valued than the customer route $R_{m[c,d]}$, that switch must have also happened because of a withdrawal, and these same arguments apply. This could continue downstream to d, but the last link must be a customer route that is always available; this leads to a contradiction. □

Given Lem. 2, the convergence to a stable route allocation implied by Thm. 1 for BGP also applies to our algorithm instances that obey the policy restrictions in Subsec. 4.1, because the dynamics of route choices made by our algorithm (for the original instance I) are the same as BGP.

The second difference is that the algorithm sends additional messages to find next hops in sub-instances I^{-k}, where $k \in [n]$. In particular, update messages are sent whenever the availability of k-avoiding routes changes (*i.e.*, some change in the list L_i). These messages are not used in BGP; so, to prove that the algorithm converges, we must show that they eventually stop as well.

First, note that the Gao-Rexford conditions hold for sub-instances if they hold for the original instance; therefore, a unique, stable routing tree exists for each sub-instance, and route withdrawals are unnecessary. Second, because valuations are next-hop based, only the availability of a k-avoiding route through a given neighbor needs to be known, not the route itself. (This is why the algorithm only sends a bit vector of availability.) But, because routes are never withdrawn, once a neighbor indicates that a k-avoiding route is available, a k-avoiding route through that neighbor will always be available. Because there are a finite number of neighbors, k-avoiding-route availability can improve only a finite number of times. Thus, at some point along every edge, update messages will no longer be sent.

This means the algorithm will converge on the instances considered, and, by Prop. 1 and Thm. 1, its output is the unique, stable route allocation. □

4.3.2 Welfare Maximization

Theorem 5 *The routing tree T_d output by the algorithm in Subsec. 4.2 maximizes the total social welfare on instances with next-hop policies that obey the Gao-Rexford conditions.*

Proof This result also follows from more general results presented in Sec. 3. By Prop. 1 and Thm. 2, the unique stable route allocation is welfare-maximizing. By Thm. 4, the algorithm is robust on instances with next-hop policies that obey the Gao-Rexford conditions; thus, it converges and outputs that unique stable route allocation. □

4.3.3 BGP Compatibility

We are left with showing that the algorithm is BGP-compatible. In addition to the routing-table storage required by the original BGP, this algorithm requires, at node i, storage of:

1. the bit $B_m(j)$ for every $j \in R_m$ sent in an update message m stored at i; and
2. the next hops on the currently best known k-avoiding routes for every $k \in R_i$, where R_i is the current best route to d.

This requires one additional bit per transit AS, per row (*i.e.*, per update message) in the routing table and one additional row to store the next hops. This amounts to a constant-factor increase in space complexity and fulfills our requirements for BGP compatibility.

4.4 Incentive Compatibility

We now prove that our algorithm is incentive-compatible. We first prove this result in a restricted, centralized model and then use it to prove incentive compatibility in a more general, distributed model.

4.4.1 Centralized Model

To prove that our algorithm is incentive-compatible in ex-post Nash equilibrium, we first consider the following centralized (and unrealistic) model. The nodes are communicating directly with some trusted central entity ("the mechanism"). Each node reports its valuation function to the mechanism, which then runs the algorithm in Subsec. 4.2, simulating the nodes' actions, to compute the route allocation and the nodes' payments.

Classical results in microeconomic theory (see [14]) establish that *Vickrey-Clarke-Groves* (VCG) payments guarantee the strongest possible result for this centralized model: truthful reporting by all nodes leads to a *dominant-strategy equilibrium*. That is, a rational node's best strategy is to report its true preferences *regardless* of the valuation functions reported by the other nodes. Hence, a node need not make any assumptions about the other nodes' behavior or have any *a priori* knowledge about their preferences. Intuitively, the VCG payment to each node i is the increase in the social welfare of the other nodes caused by i's participation in the algorithm.

In the language of microeconomic theory, a centralized algorithm in which truth telling is a dominant-strategy equilibrium is called *strategyproof*. We prove that our algorithm is strategyproof by showing that it is a member of the VCG class.

Theorem 6 *The algorithm in Subsec. 4.2 is strategyproof.*

VCG payments are expressible as

$$p_k = \sum_{i \neq k} v_i(R_i) - h_k(T_d^{-k}), \tag{5}$$

in which $h_k(\cdot)$ is an arbitrary function of T_d^{-k}. Note that this implies that every strategic agent's payment must depend solely on the other agents.

We define the payment to each node to be

$$s_k = \sum_{i \neq k} v_i(R_i) - \sum_{i \neq k} v_i(R_i^{-k}), \tag{6}$$

where R_i is the route allocated to i in T_d, and R_i^{-k} is the route allocated to i in T_d^{-k}. Note that, if

$$h_k(T_d^{-k}) = \sum_{i \neq k} v_i(R_i^{-k})$$

in (5), then $p_k = s_k$.

The key observation is that these payments can be "broken down" into components computed by the different nodes (in a distributed fashion). Loosely speaking, node i's component in the payment to node j corresponds to j's contribution to i's welfare—the difference in the values i assigns to the paths he gets with and without j. These components are computed during the algorithm, and the final payment is the sum of payment components computed once the algorithm converges.

Definition 14 The *payment component* for j attributable to i is

$$s_j^i = v_i(R_i) - v_i(R_i^{-j}),$$

and the *payment* to each node k is

$$s_k = \sum_{i \neq j} s_k^i.$$

It is easy to verify that the payment s_k in Def. 14 is the same as that in (6).

Payment components must be computed for transit nodes only; if j is not a transit node on i's best route, *i.e.*, $j \notin R_i$, then $R_i = R_i^{-j}$, and $s_j^i = 0$. We now show that, at the end of the algorithm, each node i has enough information to compute s_j^i for all transit nodes j. Because preferences are next-hop based, $s_j^i = v_i(R_i) - f_i(L_i(j))$, where f_i is the next-hop valuation as in Def. 6. Thus, Thm. 6 will follow from the fact that $L_i(j)$ is the next hop of the best j-avoiding route computed by the algorithm, which we prove in Thm. 7.

Theorem 7 *For every source node i, the node $L_i(k)$ in the algorithm in Subsec. 4.2 is the next hop of the optimal route for i in G^{-k}.*

Proof We shall require the following four lemmas.

Lemma 3 *If j is the optimal next hop for i, and, for some $k \in [n]$, j has a k-avoiding route, then the next hop of the optimal k-avoiding route at i is also j.*

This lemma justifies the step in the algorithm that immediately sets k-avoiding next hops whenever an update message containing a new best route is received.

Proof (of Lem. 3) By Thm. 3, if j is the optimal next hop, then

$$j = \text{argmax}_{a \in \text{neighbors(a)}} f_i(a).$$

Therefore, if j has a k-avoiding route R for some $k \in [n]$, then $v_i(R) = f_i(j) \geq f_i(\text{next}(R'))$ for all other k-avoiding routes R'. Thus j is also the next hop of the optimal k-avoiding route at i. □

Lemma 4 *If node i has not received an update message from neighbor a, then either node a's route in I^{-k} (for any $k \in [n]$) cannot be exported to i, or node a has no route in I^{-k}.*

This lemma means that neighbors with k-avoiding routes permitted at i will send update messages to i; information from neighbors that do not send update messages to i is irrelevant in computing payment components.

Proof (of Lem. 4) If a is routing through i, then a will send an update message if it has any k-avoiding routes available. Thus, without loss of generality, we can assume that a is not routing through i.

If a has not sent an update message to i because it has not learned any paths to d, then a also has no k-avoiding routes to d.

The remaining case is that a has not sent an update message to i because it cannot announce its route R_a to i. In this case, i must not be a customer of a, and $\text{next}(R_a)$ is also not a customer of a. If $k \notin R_a$, then R_a is a k-avoiding route,

but a cannot export it to i because next(R_a) and i are both non-customers.

If $k \in R_a$, then a may choose a different route R_a^{-k} in I^{-k}. If R_a^{-k} is a non-customer route, then it is still unusable by i, which explains who no update was sent. If R_a^{-k} is a customer route, then it must not be available to a when k is present; otherwise, a would choose it over the non-customer route R_a. But this is not possible, because every link $(u,w) \in R_a^{-k}$ is a customer link, including the last link to d. This means that the route must be exported up the chain of providers to node a at all times, which leads to a contradiction; therefore, R_a^{-k} cannot be a customer route at node a, which makes it unusable by node i. □

Lemma 5 *If $k \notin R_a$ (the route allocated to a by the algorithm for the original instance I) and $(i,a)R_a \in P^i$, then there exists a route $R_a^{-k} \in P^a$ such that $(i,a)R_a^{-k} \in P^i$ for the sub-instance I^{-k}.*

This lemma addresses availability of k-avoiding routes. Although a node may choose a k-avoiding route as its best route for I, it may be that downstream changes prevent it from choosing that route in the sub-instance I^{-k}; in fact, it is possible that no k-avoiding route is available. This lemma excludes this possibility. The algorithm uses this fact to populate the lists L_i.

Proof (of Lem. 5) If no node $j \in R_a$ chooses a different path (other than R_j) when k is not present, then R_a itself is a k-avoiding path usable by i. If some downstream node j switches to a different path R'_j when k is removed, then the path $R_{a[a,j]}R'_j$ should be usable at i, unless it is filtered somewhere between j and i.

Assume this happens. The relationships among nodes between j and i have not changed: Because these nodes originally propagated R_j, they would also propagate R'_j; therefore, j itself must filter R'_j. This means that R'_j must be a non-customer route, and the node upstream of j towards a must also be a non-customer. But because R_j was not filtered, it must be a customer route. Because $v_j(R_j) > v_j(R'_j)$ in this case, j would never have switched to R'_j upon removal of k unless R_j was filtered downstream of j. However, this same argument applies to all downstream nodes (which must all be customers); because the last link adjacent to d must be a customer link and the direct route is always exported, this leads to a contradiction. □

Lemma 6 *Given some fixed k, it is not possible to have $L_i(k) = j$ and $L_j(k) = i$ at the same time.*

In the algorithm, nodes send their k-avoiding-route availability to their neighbors. This lemma precludes the possibility that two nodes choose each other as their k-avoiding next hop.

Proof (of Lem. 6) If i is a customer of j, then the only routes exported to j are customer routes. Therefore, if i exports a k-avoiding route R to j such that j considers $(j,i)R$ its best k-avoiding route, R is a customer route at i. This implies $f_i(\text{next}(R)) > f_i(j)$; so, $L_i(k) \neq j$. The same argument works, by symmetry, if j is a customer of i.

If i and j are peers, then the only routes they export to each other are customer routes. Assume that each node chooses the other as a best k-avoiding next hop; then each must have a customer route exported to the other. But those customers would be better choices for k-avoiding next hops, contradicting the assumption. □

We are now ready to prove Thm. 7. We have already shown that the algorithm converges and that, when it does, the route choice is optimal; thus, every node i receives a route through its most highly valued neighbor j. From Lem. 2, we know that, once i learns a route through j, it always has a current update message from j; update messages are sent whenever a change to the best route or the best k-avoiding next hop (for any k) occurs.

For each k, consider the entry $L_i(k)$ that is in the list when the algorithm converges. These entries have been populated in the following way. $L_i(k) = j$ if $B_j(k) = 1$ or $k \notin (i,j)R_j$; *i.e.*, $L_i(k) = j$ if j has a k-avoiding route. By Lem. 3, if j has a k-avoiding route for some k, then this entry $L_i(k)$ is optimal.

If $B_j(k) = 0$ and $k \in (i,j)R_j$, then j does not have a k-avoiding route. In this case, the algorithm sets $L_i(k)$ to be the most valued neighbor m that has sent an update message (R_m, B_m) in which either $k \notin R_m$ or $B_m(k) = 1$. First, we show that the algorithm chooses the most valued neighbor; then we show that the neighbor has a k-avoiding route.

By Lem. 4, we must only consider neighbors that send update messages as candidates for the optimal k-avoiding next hop; thus, the algorithm is not excluding viable choices by examining update messages alone. The entry for $L_i(k)$ is set in either case I or case II of the algorithm. If set in case I, the entry is the most valued neighbor because the latest update messages are scanned in decreasing order of valuation; the scan is accurate because case I resets L_i and then examines the most recent update messages. If set in case II, the entry is the most valued because $L_i(k)$ is only set when an update message is received from a neighbor more valued than the previous $L_i(k)$, which was either set by a case-I or case-II message; thus, at convergence, the entry will represent the most valued neighbor with a k-avoiding route.

By Lem. 5, if $k \notin R_m$, then m must have a k-avoiding route usable by i, and the algorithm does not need to scan B_m. If $B_m(k) = 1$, the update message from m itself states that m has a k-avoiding route. Therefore, the neighbor chosen for $L_i(k)$ certainly has a k-avoiding route.

Finally, Lem. 6 and the Gao-Rexford conditions assure us that the next hops chosen at different nodes do not create routing loops; thus they are consistent with a tree. □

4.4.2 Distributed Model

We have thus far considered a centralized model in which nodes report valuations to a trusted mechanism that then computes the route allocation and payments. We now turn our attention to the distributed model, in which the computation is executed by the strategic agents themselves (in our case, the ASes). The distributed model is strictly less restrictive than the centralized model above. As in the centralized case, a node can pretend to have another valuation function, simply by following the specification of the algorithm as if its valuation function were different. However, in the distributed model, a node also has other forms of "manipulation" available to it: making bogus route announcements to other nodes, announcing inconsistent information to different neighbors, inconsistently filtering, and more. Thus, incentive compatibility in the distributed model requires stronger assumptions than in the centralized model.

The techniques and assumptions we use to prove incentive compatibility in the distributed model follow closely the work of Shneidman and Parkes [22]. We show that a node cannot benefit by deviating from the information-revelation, communication, and computational actions it is instructed to perform by the protocol.[5] We assume an environment in which there exists a *unique* trusted node called "*the bank*" that functions as a non-strategic accounting and charging infrastructure, communicates with the strategic source nodes across the network, and can enforce penalties when it detects a problem. The only modification needed to the algorithm is requiring that all communication between the bank and the nodes be signed and receive signed acknowledgments. With this minor modification, we are able to prove that our distributed algorithm is incentive-compatible in *ex-post Nash equilibrium*.

An ex-post Nash equilibrium is a robust solution concept: In such an equilibrium, no single node would deviate from the algorithm even if it knew the other nodes' private valuations.[6] In the context of interdomain routing, this means that no AS would deviate from the algorithm even if it knew the other ASes' routing policies.

We now define ex-post Nash equilibrium in the context of interdomain routing; see [22] for a general game-theoretic definition. Let A be an algorithm and let $v = (v_1, \ldots, v_n)$ be an n-tuple of nodes' valuation functions. Let v_{-i} denote the tuple of all valuations except that of node i. Let $o_i^A(v)$ denote i's outcome (route and payment) when all nodes (including i) execute A and their valuations are as in v. Finally, let $O_i^A(v_{-i})$ denote the set of outcomes (routes and payments) that i can achieve if all *other* nodes execute A and their valuations are as in v_{-i} (*i.e.*, all outcomes that node i can obtain via "manipulations").

Definition 15 An algorithm A is *incentive-compatible in ex-post Nash equilibrium* if, for all v,

$$\forall o \in O_i^A(v_{-i}),\ u_i(o_i^A(v)) \geq u_i(o).$$

Thus, if A is incentive-compatible in ex-post Nash equilibrium, then each AS is best off (*i.e.*, its utility is weakly highest) by following A whenever all other ASes follow A, *regardless* of the routing policies of the other ASes. Shneidman and Parkes [22] view the need to settle for an ex-post Nash equilibrium in the distributed model (instead of a dominant-strategy equilibrium, as in the centralized model) as "the cost of distributing mechanism computation across a network."

Theorem 8 *The modified algorithm (with signed communication) is incentive-compatible in ex-post Nash equilibrium.*

Proof Consider the following three components of nodes' actions as prescribed by a distributed algorithm:

1. **Information-revelation actions:** These are the subset of the algorithm's prescribed actions from which the deviation of a node is equivalent to that node's executing the algorithm with a different valuation function.
2. **Message-passing actions:** These are the subset of the algorithm's prescribed actions that instruct the node to pass a message from one neighbor to another (*e.g.*, passing messages from other nodes to the bank).
3. **Computational actions:** These are the subset of the algorithm's prescribed actions that instruct the node to participate in the algorithm's calculations (*e.g.*, computing the payment components).

We will use the following proposition in our proof.

Proposition 2 *(Shneidman and Parkes [22]) An algorithm A is incentive-compatible in ex-post Nash (in the distributed model) if the following three conditions hold:*

1. *A is strategyproof in the centralized model;*
2. *Each node is always best off when it executes the message-passing actions prescribed by A (regardless of that node's information-revelation and computational actions);*
3. *Each node is always best off when it executes the computational actions prescribed by A (regardless of that node's information-revelation and message-passing actions).*

[5] These three properties are called IC-, CC-, and AC-compatibility in [22].

[6] The ex-post Nash equilibrium concept is strictly stronger than the well known Nash-equilibrium concept. A Nash-equilibrium-oriented implementation of our algorithm would have to assume that every node is familiar with the preferences of all other nodes. This assumption is unrealistic in interdomain routing.

We now show that our algorithm in Subsec. 4.2, modified with signed communication, meets the requirements of Prop. 2. Thm. 6 establishes that our algorithm is strategyproof in the centralized model. Thus, we are left with showing that a node is never incentivized to deviate from the prescribed message-passing and computational actions (regardless of its other actions).

We first consider message-passing actions. Observe that the only message-passing actions in our algorithm are those in which a node forwards messages from another node to the bank. Because these messages are signed, the node cannot change the messages' contents; thus, the only form of deviation available to it is dropping the messages. However, recall that all communication between the bank and the nodes must be paired with signed acknowledgments, and that the bank is capable of penalizing nodes when detecting deviations. Thus, a node is never incentivized not to follow the suggested message-passing actions.

We next consider the computational actions, which, in our algorithm, are the computation of the payment components. Because each node a only computes payment components for other nodes, and because these payment components have no bearing on a's route allocation and payment, node a is never incentivized not to follow the suggested computational actions.

Hence, the three conditions of Prop. 2 hold for our modified algorithm, and Thm. 8 follows. □

5 Conclusions and Open Questions

In this paper, we addressed the problem of incentive-compatible, welfare-maximizing interdomain routing. We presented welfare-maximizing, incentive-compatible and BGP-compatible mechanisms for a class of routing policies that is more general than LCP routing, thus answering an open question from [4, 20]. Additionally, we derived general conditions that are sufficient for designing incentive-compatible, welfare-maximizing protocols for more general classes of routing policies. It would be interesting to find other natural classes of valuations for which BGP-compatible mechanisms exist, especially in the case of negative valuations.

There are many other issues that remain unresolved and call for further research. One such issue is that of designing distributed BGP-compatible mechanisms that obtain *good approximations* of the total social welfare. A first step towards the design of BGP-compatible approximation mechanisms would be a nontrivial characterization of routing policies for which the price of anarchy is low.

Introducing incentive compatibility into the interdomain-routing problem involves paying ASes for their participation in the algorithm when valuations may be negative. The way these payments are computed leads to many interesting questions: How can we make sure that the ASes are not overpaid for the transit services they provide? (VCG mechanisms are often criticized in the literature for overpaying the strategic agents.) In our formulation, the ASes do not pay each other but are paid by *the bank* (as in [22]). Is it possible to get rid of the bank and have ASes pay other ASes directly for transit services rendered?

A distributed model such as ours poses an inherently different challenge for the design of incentive-compatible mechanisms that involve payments than a centralized one (see [4, 22]). This is because the computation is performed by the strategic agents themselves and not by a reliable third party. We reconcile the strategic model and the distributed computational model by using techniques similar to those in [22]. In particular, we use cryptographic signing. Is it possible to reconcile the two models without having to resort to this technique?

Finally, the question of optimal communication complexity for the computation of routes and payments remains open. We have stressed space complexity in this paper, but there may be an increase over BGP in the number of update messages sent by our algorithms. This is because our algorithms have an additional condition that triggers sending an update message, namely, any change to the best known k-avoiding route (or next hop), for any transit node k on the current best path. Update messages are not sent for this reason in the original BGP. Although the message complexity of our algorithms is not unreasonable with respect to BGP's worst-case performance, the optimal number of messages needed to compute payments in addition to routes is currently unknown.

Acknowledgements The authors thank Tim Griffin, Aaron Jaggard, Jennifer Rexford, Rahul Sami, and Scott Shenker for many helpful discussions about interdomain routing.

References

1. Caesar, M., Rexford, J.: BGP Policies in ISP Networks. *IEEE Network Magazine* **19**(6):5–11 (2005)
2. Feamster, N., Johari, R., Balakrishnan, H.: The Implications of Autonomy for the Expressiveness of Path-Vector Routing. *IEEE/ACM Trans. Networking* **15**(6):1266–1279, (2007)
3. Feigenbaum, J., Karger, D., Mirrokni, V., Sami, R.: Subjective-Cost Policy Routing. *Theor. Comput. Sci.* **378**(2):175-189. 2007
4. Feigenbaum, J., Papadimitriou, C. H., Sami, R., Shenker, S.: A BGP-based Mechanism for Lowest-Cost Routing. *Distributed Computing* **18**(1):61–72 (2005)
5. Feigenbaum, J., Ramachandran, V., Schapira, M.: Incentive-Compatible Interdomain Routing (Extended Abstract). In *Proc. 7th ACM Conference on Electronic Commerce (EC'06)*, pp. 130–139. ACM Press (2006)
6. Feigenbaum, J., Sami, R., Shenker, S.: Mechanism Design for Policy Routing. *Distributed Computing* **18**(4):293-305 (2006)
7. Feigenbaum, J., Schapira, M., Shenker, S.: Distributed Algorithmic Mechanism Design. In: Nisan, N., Roughgarden, T., Tar-

dos, É., Vazirani, V. (eds.) *Algorithmic Game Theory*, pp. 363–384. Cambridge University Press (2007)
8. Gao, L., Griffin, T. G., Rexford, J.: Inherently Safe Backup Routing with BGP. In *Proc. 20th IEEE International Conference on Computer Communications (INFOCOM'01)*, pp. 547–556. IEEE Computer Society (2001)
9. Gao, L., Rexford, J.: Stable Internet Routing without Global Coordination. *IEEE/ACM Trans. Networking* **9**(6):681–692 (2001)
10. Goldberg, S., Halevi, S., Jaggard, A. D., Ramachandran, V., Wright, R. N.: Rationality and Traffic Attraction: Incentives for Honest Path Announcements in BGP. In *Proc. 14th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'08)*, pp. 267–278. ACM Press (2008)
11. Griffin, T. G., Jaggard, A. D., Ramachandran, V.: Design Principles of Policy Languages for Path Vector Protocols. In *Proc. 9th ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM'03)*, pp. 61–72. ACM Press (2003)
12. Griffin, T. G., Shepherd, F. B., Wilfong, G.: Policy Disputes in Path Vector Protocols. In *Proc. 7th International Conference on Network Protocols (ICNP'99)*, pp. 21–30. IEEE Computer Society (1999)
13. Griffin, T. G., Shepherd, F. B., Wilfong, G.: The Stable Paths Problem and Interdomain Routing. *IEEE/ACM Trans. Networking* **10**(2):232–243 (2002)
14. Green, J., Laffont, J.: Incentives in Public Decision Making. In *Studies in Public Economics*, vol. 1, pp. 65–78. North Holland, Amsterdam (1979)
15. Hershberger, J., Suri, S.: Vickrey Prices and Shortest Paths: What is an edge worth? In *Proc. 42nd IEEE Symposium on Foundations of Computer Science (FOCS'01)*, pp. 129–140. IEEE Computer Society (2001)
16. Huston, G.: Interconnection, Peering, and Settlements. In *Proc. 9th Internet Global Summit (INET'99)*. The Internet Society (1999)
17. Koutsoupias, E., Papadimitriou, C. H.: Worst-Case Equilibria. *Comp. Sci. Review* **3**(2):65–69 (2009)
18. Levin, H., Schapira, M., Zohar, A.: Interdomain Routing and Games. In *Proc. 40th ACM Symposium on Theory of Computing (STOC'08)*, pp. 57–66. ACM Press (2008)
19. Moy, J.: Open Shortest Pouting First (OSPF) version 2. RFC 2328. Internet Engineering Task Force (1998)
20. Nisan, N., Ronen, A.: Algorithmic Mechanism Design. *Games and Economic Behavior* **35**(1,2):166–196 (2001)
21. Rekhter, Y., Li, T.: A Border Gateway Protocol (BGP-4). RFC 4271. Internet Engineering Task Force (2006)
22. Shneidman, J., Parkes, D. C.: Specification Faithfulness in Networks with Rational Nodes. In *Proc. 23rd ACM Symposium on Principles of Distributed Computing (PODC'04)*, pp. 88–97. ACM Press (2004)
23. Sobrinho, J. L.: An Algebraic Theory of Dynamic Network Routing. *IEEE/ACM Trans. Networking* **13**(5):1160–1173 (2005)
24. Varadhan, K., Govindan, R., Estrin, D.: Persistent Route Oscillations in Interdomain Routing. *Computer Networks* **32**(1):1–16 (2000)