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

A BGP-based Mechanism for Lowest-Cost Routing*

Joan Feigenbaum[**,1], **Christos Papadimitriou**[***,2], **Rahul Sami**[†,3], **Scott Shenker** [‡,4]

[1] Yale University, Computer Science Department, New Haven, CT 06520 USA. e-mail: `feigenbaum@cs.yale.edu`
[2] University of California at Berkeley, Computer Science Division, Berkeley, CA 94720 USA. e-mail: `christos@cs.berkeley.edu`
[3] Yale University, Computer Science Department, New Haven, CT 06520 USA. e-mail: `sami@cs.yale.edu`
[4] ICSI, 1947 Center Street, Berkeley, CA 94704 USA. e-mail: `shenker@icsi.berkeley.edu`

January 7, 2005

Summary. *The routing of traffic between Internet domains, or* Autonomous Systems *(ASs), a task known as* interdomain routing, *is currently handled by the Border Gateway Protocol (BGP). In this paper, we address the problem of interdomain routing from a mechanism-design point of view. The application of mechanism-design principles to the study of routing is the subject of earlier work by Nisan and Ronen [16] and Hershberger and Suri [12]. In this paper, we formulate and solve a version of the routing-mechanism design problem that is different from the previously studied version in three ways that make it more accurately reflective of real-world interdomain routing: (1) we treat the nodes as strategic agents, rather than the links; (2) our mechanism computes lowest-cost routes for all source-destination pairs and payments for transit nodes on all of the routes (rather than computing routes and payments for only one source-destination pair at a time, as is done in [16,12]); (3) we show how to compute our mechanism with a distributed algorithm that is a straightforward extension to BGP and causes only modest increases in routing-table size and convergence time (in contrast with the centralized algorithms used in [16,12]). This approach of using an existing protocol as a substrate for distributed computation may prove useful in future development of Internet algorithms generally, not only for routing or pricing problems. Our design and analysis of a strategyproof, BGP-based routing mechanism provides a new, promising direction in* distributed algorithmic mechanism design, *which has heretofore been focused mainly on multicast cost sharing.*

* This work was supported by the DoD University Research Initiative (URI) program administered by the Office of Naval Research under Grant N00014-01-1-0795. It was presented in preliminary form at the 2002 ACM Symposium on Principles of Distributed Computing [5].

** Supported in part by ONR grants N00014-01-1-0795 and N00014-01-1-0447 and NSF grant CCR-0105337.

*** Supported in part by NSF grants ITR-0081698 and ITR-0121555 and by an IBM Faculty Development Award.

† Supported by ONR grant N00014-01-1-0795.

‡ Supported in part by NSF grants ITR-0205519, ANI-0207399, ITR-0121555, ITR-0081698, ITR-0225660, and ANI-0196514.

1 Introduction

The Internet is comprised of many separate administrative domains known as *Autonomous Systems* (ASs). Routing occurs on two levels, intradomain and interdomain, implemented by two different sets of protocols. Intradomain-routing protocols, such as OSPF, route packets within a single AS. Interdomain routing, currently handled by the Border Gateway Protocol (BGP), routes packets between ASs. Although routing is a very well-studied problem, it has been approached by computer scientists primarily from an engineering or "protocol-design" perspective.

In their seminal paper on *algorithmic mechanism design*, Nisan and Ronen [16] advocate combining an economic, "incentive-compatibility" approach with the more traditional protocol-design approach to the problem. Internet routing is an extremely natural problem in which to consider incentives, because ownership, operation, and use by numerous independent, self-interested parties give the Internet the characteristics of an economy as well as those of a computer. In this paper, we continue the study of routing from a mechanism-design perspective, concentrating specifically on interdomain routing, for reasons explained below.

In our formulation of the routing-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 paid a *price* for carrying *transit* traffic, which is traffic neither originating from nor destined for that AS. It is through these costs and prices that consideration of "incentive compatibility" is introduced to the interdomain-routing framework, which, as currently implemented, does not explicitly consider incentives. Our goal is to maximize network efficiency by routing packets along the lowest-cost paths (LCPs). We are following previous work on mechanism design for routing [16,12] by introducing incentives in this way, and focusing on lowest-cost paths.

The main difference between this paper and previous algorithmic mechanism design for routing is our focus on computing the routes and prices in a *distributed* fashion. Furthermore, in order to steer the mechanism-design approach towards practical implementation, we consider only distributed algorithms that retain the data structures and communication protocol of BGP. We develop a "BGP-based computational model" to capture these requirements and seek to compute the routes and prices in this model.

Given a set of costs, the LCPs can be computed using standard routing protocols (such as BGP). However, under many pricing schemes, an AS could be better off lying about its costs;[1] such lying would cause traffic to take non-optimal routes and thereby interfere with overall network efficiency.

To prevent this, we first ask how one can set the prices so that ASs have no incentive to lie about their costs; as we discuss in Section 2, such pricing schemes are called "strategyproof." We also require that ASs that carry no transit traffic receive no payment. We prove that there is only one strategyproof pricing scheme with this property; it is a member of the Vickrey-Clarke-Groves (VCG) class of mechanisms [25, 3, 11]. This mechanism requires a per-packet price to be paid to each transit node k; this price is determined by the cost of the LCP and the cost of the lowest-cost path that does not pass through k. We next ask how the VCG prices should be computed, and we provide a "BGP-based" distributed algorithm that accomplishes this.

Our results contribute in several ways to the understanding of how incentives and computation affect each other in routing-protocol design. Nisan and Ronen [16] and Hershberger and Suri [12] considered the LCP mechanism-design problem, motivated in part by the desire to include incentive issues in Internet-route selection. The LCP mechanism studied in [16, 12] takes as input a biconnected graph, a single source, a single destination, and a (claimed) transmission cost for each link; the strategic agents are the links, and the mechanism computes, in a strategyproof manner, both an LCP for this single routing instance and a set of payments to the links on the LCP. This mechanism is a member of the VCG family and forms the point of departure for our work. However, our formulation of the problem differs in three respects, each of which makes the problem more representative of real-world routing:

- First, in our formulation, it is the nodes that are the strategic agents, not the links as in [16, 12]. We make this choice because we are trying to model *interdomain* routing. ASs actually *are* independent economic actors who could strategize for financial advantage in interdomain-routing decisions; in the BGP computational model into which we seek to incorporate incentive issues, it is the nodes that represent ASs and that are called upon to "advertise" their inputs to the protocol. Formulations in which the links are the strategic agents might be more appropriate for intradomain routing, but it is not clear that incentive issues are relevant in that context; because all links and routers within a domain are owned and managed by a single entity, they are unlikely to display strategic behavior.
- Second, instead of taking as input a single source-destination pair and giving as output a single LCP, our mechanism takes in n AS numbers and constructs LCPs for all source-destination pairs. Once again, we make this choice in order to model more accurately the interdomain-routing scenario. This complicates the problem, because there are now n^2 LCP instances to solve.
- Third, we compute the routes and the payments not with a centralized algorithm, as is done in [16, 12], but with a distributed protocol based on BGP. This is necessary if the motivation for the mechanism-design problem is Internet routing, because interdomain-route computation is in fact done in a distributed fashion, with the input data (AS-graph topology) and the outputs (interdomain routes) stored in a distributed fashion as well. The various domains are administratively separate and in some cases competitors, and there is no obvious candidate for a centralized, trusted party that could maintain an authoritative AS graph and tell each of the ASs which routes to use. Real-world BGP implementations could be extended easily to include our pricing mechanism, and we prove that such an extension would cause only modest increases in routing-table size and convergence time.

Our approach of using an existing network protocol as a substrate for realistic distributed computations may prove useful generally in Internet-algorithm design, not only in routing or pricing problems. Algorithm design for the Internet has the extra subtlety that adoption is not a decision by a systems manager, concerned only with performance and efficiency, but rather a careful compromise by a web of autonomous entities, each with its own interests and legacies. Backward compatibility with an established protocol is a constraint and criterion that is likely to become increasingly important and prevalent.

Despite these efforts to formulate the problem realistically, there are several aspects of reality that we deliberately ignore. First, per-packet costs are undoubtedly not the best cost model, *e.g.*, in some cases transit costs are more administrative than traffic-induced. Second, BGP allows an AS to choose routes according to any one of a wide variety of local policies; LCP routing is just one example of a valid policy, and, in practice, many ASs do not use it [24]. Furthermore, most ASs do not allow non-customer transit traffic on their network.[2] In this paper, we ignore general policy routing and transit restrictions; we only use LCPs. Lastly, BGP does not currently consider general path costs; in the cases in which AS policy seeks LCPs, the current BGP simply computes *shortest* AS paths in terms of number of AS hops. This last aspect is minor, because it would be trivial to modify BGP so that it computes LCPs; in what follows, we assume that this modification has been made.

Because of these limitations, our results clearly do not constitute a definitive solution to the incentive problem in interdomain routing. Nonetheless, they represent measurable progress on two fronts. First, although it does not capture all of the important features of interdomain routing, our problem formulation is an improvement over the previous ones in the algorithmic mechanism-design literature [16, 12], as explained above. Second, we have expanded the scope of *dis-*

[1] There are two ways in which lying might increase the AS's total welfare: Announcing a lower-than-truthful cost might attract more than enough additional traffic to offset the lower price, or announcing a higher-than-truthful cost might produce an increase in the price that is more than sufficient to offset any resulting decrease in traffic.

[2] We say that two ASs are "interconnected" if there is a traffic-carrying link between them. Interconnected ASs can be *peers*, or one can be a customer of the other. Most ASs do not accept transit traffic from peers, only from customers.

tributed algorithmic mechanism design, which has heretofore been focused mainly on multicast cost sharing [6,1,4].

In the next section, we give a brief review of algorithmic mechanism design. In Section 3, we provide a formal statement of the problem and in Section 4 derive the pricing scheme. In Section 5, we describe the BGP-based computational model that we use for the distributed price-calculation algorithm given in Section 6. We conclude in Section 7 with a brief discussion of open problems and future work.

2 Algorithmic Mechanism Design

The purpose of this section is to review the basics of algorithmic mechanism design. Readers already familiar with this area, *e.g.,* through the early papers of Nisan and Ronen [16] and Feigenbaum, Papadimitriou, and Shenker [6], should skip to the next section.

In designing efficient, distributed algorithms and network protocols, computer scientists typically assume either that computational agents are *obedient* (*i.e.,* that they follow the protocol) or that they are *Byzantine adversaries* (*i.e.,* that they may deviate from the protocol in arbitrary ways that harm other users, even if the deviant behavior does not bring them any obvious tangible benefits). In contrast, economists design market mechanisms in which it is assumed that agents are neither obedient nor adversarial but rather *strategic*: They respond to well-defined incentives and will deviate from the protocol only for tangible gain. Until recently, computer scientists ignored incentive compatibility, and economists ignored computational efficiency.

The emergence of the Internet as a standard, widely used distributed-computing environment and of Internet-enabled commerce (both in traditional, "real-world" goods and in electronic goods and computing services themselves) has drawn computer scientists' attention to incentive-compatibility questions in distributed computation. In particular, there is growing interest in incentive compatibility in both distributed and centralized computation in the theoretical computer science community (see, *e.g.,* [2,6,8,12,16,19]) and in the "distributed agents" part of the AI community (see, *e.g.,* [15,17,18,21,26, 27]).

A standard economic model for the design and analysis of scenarios in which the participants act according to their own self-interest is as follows: There are n agents. Each agent i, for $i \in \{1, \ldots, n\}$, has some private information t^i, called its *type*. For each mechanism-design problem, there is an *output specification* that maps each type vector $t = (t^1, \ldots, t^n)$ to a set of allowed outputs. Intuitively, the specification tells us what a well-meaning planner would do, if she could access all the agents' private information. Each agent has his own preferences over the possible outputs, which may depend on his private information. Agent i's preferences are given by a *valuation function* v^i that assigns a real number $v^i(t^i, o)$ to each possible output o. For example, in an instance of the task-allocation problem studied in the original paper of Nisan and Ronen [16], there are k tasks $z_1, \ldots, z_k$, agent i's type $t^i = (t^i_1, \ldots, t^i_k)$ is the set of minimum times in which it is capable of completing each of the tasks, the space of feasible outputs consists of all partitions $Z = Z^1 \sqcup \ldots \sqcup Z^n$, in which Z^i is the set of tasks assigned to agent i, and the valuation functions are $v^i(Z, t^i) = -\sum_{z_j \in Z^i} t^i_j$. Except for the private-type information, everything else in the scenario is public knowledge.

A *mechanism* defines for each agent i a set of strategies A^i. Each agent i "plays" a strategy $a^i \in A^i$; the input to the mechanism is the vector of strategies $(a^1, \ldots, a^n)$. For each input vector $(a^1, \ldots, a^n)$ the mechanism computes an *output* $o = o(a^1, \ldots, a^n)$ and a *payment vector* $p = (p^1, \ldots, p^n)$, where $p^i = p^i(a^1, \ldots, a^n)$. The payment p^i is used to incentivize agent i to behave in accordance with the mechanism's overall goals. Agent i's *utility* is $v^i(t^i, o) + p^i$, and it is this quantity that the agent seeks to maximize. A *strategyproof* mechanism is one in which types are part of the strategy space A^i, and each agent maximizes his utility by giving his type t^i as input regardless of what other agents do. In other words, the relation

$$\begin{aligned} &v^i(t^i, o(a^{-i}, t^i)) + p^i(a^{-i}, t^i) \\ \geq\ &v^i(t^i, o(a^{-i}, a^i)) + p^i(a^{-i}, a^i) \end{aligned}$$

(where a^{-i} denotes the vector of strategies of all players except player i) must hold for all i and all possible values of t^i, a^{-i} and a^i.

Thus, the mechanism wants each agent to report his private type truthfully, and it is allowed to pay agents in order to provide incentives for them to do so. In the task-allocation problem described above, an agent may be tempted to lie about the times he requires to complete each task, in the hope that his resulting allocation will have a higher valuation. If tasks were allocated by a strategyproof mechanism, he would have no incentive to do this, because his resulting payment would be lower; indeed it would be sufficiently lower that his overall utility would be no greater than it would have been if he had told the truth.

For a thorough introduction to economic mechanism design, see Chapter 23 of the book by Mas-Colell, Whinston, and Green [13].

In their seminal paper on *algorithmic mechanism design*, Nisan and Ronen [16] add computational efficiency to the set of concerns that must be addressed in the study of how privately known preferences of a large group of selfish entities can be aggregated into a "social choice" that results in optimal allocation of resources. Succinctly stated, Nisan and Ronen's contribution to the mechanism-design framework is the notion of a (centralized) *polynomial-time mechanism*, *i.e.,* one in which $o()$ and the $p^i()$'s are polynomial-time computable. They also provide strategyproof, polynomial-time mechanisms for some concrete problems of interest, including LCPs and task allocation.

To achieve feasible algorithmic mechanisms within an Internet infrastructure, the mechanism-design framework must be enhanced with more than computational efficiency; it also requires a distributed computational model. After all, if one assumes that massive numbers of far-flung, independent agents are involved in an optimization problem, one cannot reasonably assume that a *single, centralized* "mechanism" receives all of the inputs and doles out all of the outputs and payments. The first work to address this issue is the multicast cost-sharing paper of Feigenbaum, Papadimitriou, and Shenker [6]. This work does not attempt to provide a general decentralized-mechanism computational model. Rather, it achieves the more modest goal of using the same network-algorithmic infrastructure that is needed for multicast to compute two natural mechanisms for assigning cost shares to the recipients of the multicast. It puts forth a general concept of "network complexity" that requires the distributed algorithm executed over an interconnection network T to be modest in four different respects: the total number of messages that agents send over T, the maximum number of messages sent over any one link in T, the maximum size of a message, and the local computational burden on agents.

Routing has been part of the algorithmic mechanism-design agenda from the beginning. Nisan and Ronen [16] provide a polynomial-time, strategyproof mechanism for optimal route selection in a centralized computational model. In their formulation, the network is modeled as an abstract graph $G = (V, E)$. Each edge e of the graph is an agent and has a private type t^e, which represents the cost of sending a message along this edge. The mechanism-design goal is to find an LCP o between two designated nodes x and y. The valuation of an agent e is $-t^e$ if e is part of o and 0 otherwise. Nisan and Ronen show that the following simple mechanism is strategyproof: The payment to agent e is 0 if e is not on the LCP o, and the payment is $d_{G|e=\infty} - d_{G|e=0}$ if e is on o, where $d_{G|e=c}$ is the cost of the LCP through G when the cost of e is set to be c. The graph needs to be biconnected to prevent the charging of monopoly prices. Note that LCP computation and biconnectivity testing can both be accomplished by standard, polynomial-time algorithms in a centralized computation model.

As explained in the previous section, our goal in this paper is to reformulate the LCP mechanism-design problem so that it more accurately reflects the real-world problem that is the motivation for studying it (*i.e.,* interdomain routing) and to develop a distributed algorithmic mechanism that can be computed by a BGP-based protocol.

3 Statement of Problem

The network has a set of nodes N, $n = \|N\|$, where each node is an AS. There is a set L of (bidirectional) links between nodes in N. We assume that this network, called the *AS graph*, is biconnected; this is not a severe restriction, because the route-selection problem only arises when a node has multiple potential routes to a destination. For any two nodes $i, j \in N$, T_{ij} is the intensity of traffic (number of packets) originating from i destined for j.

We assume that a node k incurs a transit cost c_k for each transit packet it carries. In the terminology of Section 2, c_k is the type of agent k. For simplicity, we assume that this cost is independent of which neighbor k received the packet from and which neighbor k sends the packet to, but our approach could be extended to handle a more general case: We could have a different cost depending on which neighbor k sends the packet to, in which case we would have a cost associated with each edge, as in the cost model of [16, 12]. (The strategic agents would still be the nodes, and hence the VCG mechanism we describe here would remain strategyproof.) We write c for the vector $(c_1, \ldots, c_n)$ of all transit costs and c^{-k} for the vector $(c_1, \ldots, c_{k-1}, c_{k+1}, \ldots c_n)$ of all costs except c_k.

We also assume that each node k is given a payment p^k to compensate it for carrying transit traffic. In general, this payment can depend on the costs c, the traffic matrix $[T_{ij}]$, and the network topology. Our only assumption, which we invoke in Section 4, is that nodes that carry no transit traffic whatsoever receive no payment.

Our goal is to send each packet along the LCP, according to the true cost vector c. We assume the presence of a routing protocol (such as BGP, suitably configured) that, given a set of node costs c, routes packets along LCPs. Furthermore, we assume that, if there are two LCPs between a particular source and destination, the routing protocol has an appropriate way to break ties. Let $I_k(c; i, j)$ be the indicator function for the LCP from i to j; *i.e.*, $I_k(c; i, j) = 1$, if node k is an intermediate node on the LCP from i to j, and $I_k(c; i, j) = 0$ otherwise. Note that $I_i(c; i, j) = I_j(c; i, j) = 0$; only the *transit* node costs are counted.

Consider the routes specified by the indicator functions $\{I_k(c; i, j)\}$. When the traffic is sent along these paths, each transit node will incur a cost. Node k incurs a cost c_k for a packet sent from i to j if and only if k lies on the selected route from i to j. The total cost u_k incurred by transit node k is thus given by:

$$u_k(c) = c_k \sum_{i,j \in N} T_{ij} I_k(c; i, j)$$

The objective function we want to minimize is the total cost to society $V(c)$ of routing all packets:

$$V(c) = \sum_k u_k(c) = \sum_{i,j \in N} T_{ij} \sum_{k \in N} I_k(c; i, j) c_k$$

Minimizing V is equivalent to minimizing, for every $i, j \in N$, the cost of the path between i and j.

We treat the routing problem as a game in which the ASs are the strategic agents. Each node plays the game by reporting a transit cost. A node's transit cost is private information not known to any other node, and thus no other agent can assess the correctness of an agent's claimed transit cost. Moreover, $V(\cdot)$ is defined in terms of the true costs, whereas the routing algorithm operates on the declared costs; the only way we can be assured of minimizing $V(\cdot)$ is for agents to input their true costs. Therefore, we must rely on the pricing scheme to incentivize agents to do so.

To do so, we design an algorithmic mechanism as described in Section 2. The mechanism takes as input the AS graph and the vector c of declared costs[3] and produces as output the set of LCPs and prices.[4] The pricing mechanism must be strategyproof so that agents have no incentive to lie about their costs. For a given cost vector c, the payment p^k minus the total costs incurred by a node k is $\tau_k(c) = p^k - \sum_{i,j} T_{i,j} I_k(c; i, j) c_k$. In the terminology of Section 2, $\tau_k(\cdot)$ is the utility of agent k. In this context, the mechanism is strategyproof if for all x, $\tau_k(c) \geq \tau_k(c|^k x)$, where the expression $c|^k x$ means that $(c|^k x)_i = c_i$, for all $i \neq k$, and $(c|^k x)_k = x$.

[3] We will often use c to denote the declared costs and the true costs; usually, the context will make clear which we mean.

[4] BGP (or any other protocol configured for lowest-cost routing) can be viewed as taking the AS graph and c as input and producing the set of LCPs. We use this output of BGP in our mechanism and do not alter this aspect of BGP in our algorithm.

4 The Pricing Mechanism

Recall that we assume we have a biconnected graph with a routing algorithm that, when given a vector of declared costs c, will produce a set of LCPs, breaking ties in an appropriate manner; these paths are represented by the indicator functions $\{I_k(c;i,j)\}_{k\in N}$. Furthermore, both the inputs and the outputs are distributed, *i.e.*, neither ever resides at a single node in the network. In this section, we derive the pricing scheme, and, in Sections 5 and 6, we describe the distributed computation.

We require that the pricing mechanism be strategyproof and that nodes that carry no transit traffic receive no payment. We now show that these two conditions uniquely determine the mechanism we must use. Moreover, we show that they require that the payments take the form of a per-packet price that depends on the source and destination; that is, the payments p^k must be expressible as

$$p^k = \sum_{i,j\in N} T_{ij} p^k_{ij},$$

where p^k_{ij} is the per-packet price paid to node k for each transit packet it carries that is sent from node i destined for node j.

Theorem 1 *When routing picks lowest-cost paths, and the network is biconnected, there is a unique strategyproof pricing mechanism that gives no payment to nodes that carry no transit traffic. The payments to transit nodes are of the form* $p^k = \sum_{i,j\in N} T_{ij} p^k_{ij}$, *where*

$$p^k_{ij} = c_k I_k(c;i,j) + \left[\sum_{r\in N} I_r(c|^k\infty;i,j)c_r - \sum_{r\in N} I_r(c;i,j)c_r\right].$$

Proof. Consider a vector of costs c. Recall that our objective function $V(c) = \sum_k u_k(c)$, and note that the routing function $\{I_k(c;i,j)\}_{k\in N}$ minimizes this quantity. A classic result due to Green and Laffont [9] states that any strategyproof pricing mechanism that minimizes a function of the form $V(c) = \sum_{k\in N} u_k(c)$ (*i.e.*, minimizes the total cost) must be a VCG mechanism, with payments expressible as

$$p^k = u_k(c) - V(c) + h_k(c^{-k}),$$

where $h_k(\cdot)$ is an arbitrary function of c^{-k}. When $c_k = \infty$, we have $I_k(c|^k\infty;i,j) = 0$, for all i,j (because the graph is biconnected, and all other costs are finite); so (1) $p^k = 0$, because we require that payments be 0, and (2) $u_k(c) = 0$. Thus,

$$h_k(c^{-k}) = V(c|^k\infty).$$

This, in turn, implies that

$$\begin{aligned} p^k &= V(c|^k\infty) + u_k(c) - V(c) \\ &= \sum_{i,j\in N} T_{ij}\Big\{ c_k I_k(c;i,j) + \\ &\qquad \sum_{r\in N} I_r(c|^k\infty;i,j)c_r - \sum_{r\in N} I_r(c;i,j)c_r \Big\} \\ &= \sum_{i,j\in N} T_{ij} p^k_{ij}, \end{aligned}$$

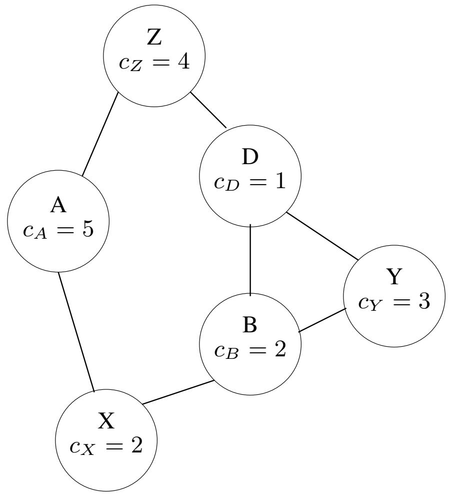

Fig. 1. Example AS graph from Section 4

where

$$p^k_{ij} = c_k I_k(c;i,j) + \left[\sum_{r\in N} I_r(c|^k\infty;i,j)c_r - \sum_{r\in N} I_r(c;i,j)c_r\right].$$

□

This mechanism belongs to the Vickrey-Clarke-Groves (VCG) family [25,3,11]. It is in essence a node-centric, all-pairs extension of the LCP mechanism studied by Nisan and Ronen [16] and Hershberger and Suri [12]. There are several aspects of this result that are worth noting. First, although the payments could have taken any form and could have depended arbitrarily on the traffic matrix, it turns out the payments are a sum of per-packet payments that do not depend on the traffic matrix. Second, the prices $p^k_{i,j}$ are zero if the LCP between i and j does not traverse k. Thus, these payments can be computed, once one knows the prices, merely by counting the packets as they enter the node. Third, although the costs did not depend on the source and destination of the packet, the prices do. Lastly, the payment to a node k for a packet from i to j is determined by the cost of the LCP and the cost of the lowest-cost path that does not pass through k. We use the term *k-avoiding path* to refer to a path that does not pass through node k.

For example, consider the AS graph in *Figure 1*, and suppose the traffic consists of a single packet from X to Z. The LCP is $XBDZ$, which has transit cost 3. How much should AS D be paid? The lowest-cost D-avoiding path from X to Z is XAZ, which has transit cost 5. Hence, Theorem 1 says that D should be paid $c_D + [5-3] = 3$. Similarly, AS B is paid $c_B + [5-3] = 4$. Note that the total payments to nodes on the path is greater than the actual cost of the path. A more extreme example of *overcharging* occurs in sending a packet from Y to Z. The LCP is YDZ, which has transit cost 1. However, the next best path is $YBXAZ$ which has cost 9, and hence D's payment for this packet is $1+[9-1] = 9$, even though D's cost is still 1. We return to this issue of overcharging in Section 7. These examples also show why the network must be biconnected; if it weren't, the payment would be undefined.

5 BGP-based Computational Model

We now seek to compute these prices p_{ij}^k, using a computational model based on BGP, which is the repository of interdomain routing information. We start with the abstract model of the BGP protocol described by Griffin and Wilfong [10], which involves several simplifying assumptions. Specifically, we assume that there is at most one link between any two ASs, that the links are bidirectional, that there is no route aggregation, and that each AS can be treated as an atomic entity without regard to intradomain routing issues. The network can then be modeled as a graph in which every node represents an AS, and every edge represents a bidirectional interconnection between the corresponding ASs.

BGP is a *path-vector* protocol in which every node i stores, for each AS j, the lowest-cost *AS Path* (the sequence of ASs traversed) from i to j; in this vector, ASs are identified by their AS numbers. In addition, in our treatment, the LCP is also described by its total cost (the sum of the declared AS costs). If d is the diameter of the network (the maximum number of ASs in an LCP), a router stores $\mathcal{O}(nd)$ AS numbers and $\mathcal{O}(n)$ path costs. BGP's route computation is similar to all path-vector routing protocols. Each router sends its routing table[5] and, in our treatment, its declared cost, to its neighbors, and each node can then, based on this information, compute its own LCPs. When there is more than one LCP, our model of BGP selects one of them in a loop-free manner (to be defined more precisely below). As mentioned earlier, we are making the oversimplifying assumption that every node is using lowest cost as its routing policy.

These routing-table exchanges only occur when a change is detected; that is, a router only sends its routing table to its neighbors when that table is different from what was sent previously. Routing tables can change either because a link was inserted or deleted (which would be detected by the nodes on either end) or when updated routing-table information is received from some other router that changes the paths and/or costs in the current table.[6]

The computation of a single router can be viewed as consisting of an infinite sequence of *stages*, where each stage consists of receiving routing tables from its neighbors, followed by local computation, followed (perhaps) by sending its own routing table to its neighbors (if its own routing table changed). The communication frequency is limited by the need to keep network traffic low, and hence the local computation is unlikely to be a bottleneck. Thus, we adopt as our measures of complexity the number of stages required for convergence and the total communication (in terms of the number of routing tables exchanged and the size of those tables).

If we assume that all the nodes run synchronously (exchange routing tables at the same time), BGP converges, *i.e.*, computes all LCPs, within d stages of computation (where, again, d is the maximum number of AS hops in an LCP). Each stage involves $\mathcal{O}(nd)$ communication on any link.[7] The computation time required by node i in a single stage is $\mathcal{O}(nd \times \text{degree}(i))$.

Because this level of complexity is already deemed feasible in the current Internet, we seek to compute the prices with a similar (or better) complexity and state requirements. We describe such an algorithm in the next section.

6 Distributed Price Computation

We want to compute the prices p_{ij}^k using the BGP computational model described in Section 5. The input to the calculation is the cost vector c, with each c_i known only to node i. The output we desire is the set of prices, with node i knowing all the p_{ij}^k values.[8] In describing our algorithm we assume a static environment (no route changes). The effect of removing this assumption is that the process of "converging" begins again each time a route is changed.

Our algorithm introduces additional state to the nodes and to the message exchanges between nodes, but it does not introduce any new messages to the protocol. In particular, all messages are between neighbors in the AS graph. The added state at each node consists of the reported cost of each transit node and the set of prices. This is $\mathcal{O}(nd)$ additional state, resulting in a small constant-factor increase in the state requirements of BGP. The costs and prices will be included in the routing message exchanges, and so there will be a corresponding constant-factor increase in the communication requirements of BGP.

Let $P(c; i, j)$ denote the LCP from i to j for the vector of declared costs c, and let $c(i, j)$ denote the cost of this path. For any node k on this LCP, define $P^{-k}(c; i, j)$ to be the lowest-cost k-avoiding path from i to j. Then, the payment p_{ij}^k can be written as:

$$p_{ij}^k = c_k + \text{Cost}(P^{-k}(c; i, j)) - c(i, j) \tag{1}$$

Our price-computation algorithm computes the prices at each node from the costs and current prices at neighboring nodes, and then iterates this procedure until the prices converge to a stable value. We first investigate how the prices p_{ij}^k at node i are related to the prices at i's neighbors.

Recall that, if there are multiple LCPs between two nodes, the routing mechanism selects one of them in a loop-free manner. *Loop-free* means that the routes are chosen so that the overall set of LCPs from every other node to j forms a tree. In other words, for each destination j, we assume that the LCPs selected form a tree rooted at j; call this tree $T(j)$. For example, the tree $T(Z)$ corresponding to the graph in *Figure 1* is shown in *Figure 2*. We say that D is the *parent* of B in $T(Z)$ or, equivalently, that B is a *child* of D in $T(Z)$.

[5] The routing table is the set of all *selected* routes, one for each destination.

[6] In practice, BGP only sends the portion of the routing table that has changed. Nodes keep the routing tables received from each of their neighbors so that they can reconstruct the new routing table from the incremental update. Because the worst-case behavior is to send the entire routing table, and we care about worst-case complexity, we ignore this incremental aspect of BGP in the statements of our bounds.

[7] Because of the incremental nature of updates, where nodes need only process and forward routing entries that have changed, the communication and computational load is likely to be much lower in practice.

[8] More precisely, these are the parts of the input and output that we introduce; BGP, with its standard distributed input (AS graph and costs) and distributed output (LCPs) is used as a substrate.

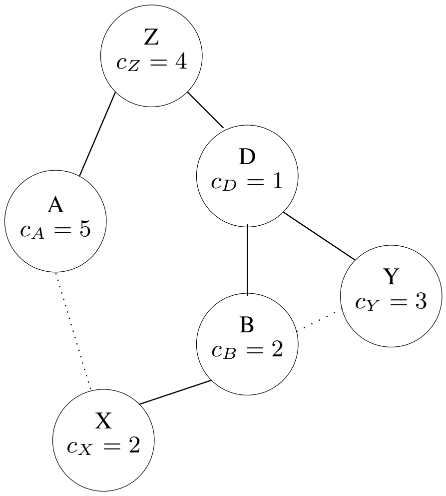

Fig. 2. Tree T(Z) for the example in *Figure 1*

We treat each destination j separately. Consider the computation of p_{ij}^k at some node i for another node k on the path from i to j. Let a be any neighbor of i. The tree $T(j)$ provides the structure we need to characterize the relationship between the prices at node i and the prices at node a. There are four cases:

- **Case (i)**: a is i's parent in $T(j)$
In other words, the LCP from i to j passes through a. In fact, it consists of the link ia followed by the LCP from a to j; thus, its cost is given by $c(i,j) = c(a,j) + c_a$. Now, suppose k is some node on the LCP from a to j. Then, k is also on the LCP from i to j. In this case, we can extend any k-avoiding path from a to j to a k-avoiding path from i to j by appending the link ia to it. In particular, we can extend the path $P^{-k}(c;a,j)$ in this fashion; thus, the lowest-cost k-avoiding path from i to j has cost at most c_a greater than the cost of $P^{-k}(c;a,j)$. Observing that the difference in cost of the corresponding LCPs is exactly c_a, Equation 1 gives us:
$$p_{ij}^k \leq p_{aj}^k \tag{2}$$
- **Case (ii)**: a is i's child in $T(j)$
In other words, i is on the LCP from a to j. Thus, the costs of the two LCPs are related by $c(i,j) = c(a,j) - c_i$. Consider any node k on the LCP from i to j; it must be on the LCP from a to j. Now, consider a k-avoiding path from a to j. If this path passes through i, then we can simply take the suffix of this path from i to j; this gives us a k-avoiding path from i to j. If the k-avoiding path from a to j does not pass through i, then we can append the link ia to it, to get a k-avoiding path from i to j. Thus, here too we see that the cost of $P^{-k}(c;i,j)$ is at most c_a more than the cost of $P^{-k}(c;a,j)$. Combining this with the difference in LCP costs, we get:
$$p_{ij}^k \leq p_{aj}^k + c_i + c_a \tag{3}$$
- **Case (iii)**: a is neither parent nor child of i in $T(j)$, and k is on $P(c;a,j)$.
In other words, k is on the LCPs from both a and i to j. The cost difference between the LCPs is $(c(i,j) - c(a,j))$. As in case (ii), given a k-avoiding path from a to j, we can construct a k-avoiding path from a to j by either shortening the path or appending the edge ia. Thus, the difference in cost of the lowest-cost k-avoiding paths is at most c_a, and so we get:
$$p_{ij}^k \leq p_{aj}^k + c_a + c(a,j) - c(i,j) \tag{4}$$
- **Case (iv)**: a is neither parent nor child of i in $T(j)$, and k is not on $P(c;a,j)$.
In this case, consider $P(c;a,j)$, the LCP from a to j. The node k does not lie on this path, and so we can append the edge ia to construct a k-avoiding path from i to j. This k-avoiding path has cost $(c(a,j) + c_a)$, and hence Equation 1 gives us:
$$p_{ij}^k \leq c_k + c_a + c(a,j) - c(i,j) \tag{5}$$

Note that these four cases are not exhaustive. In particular, the case in which $a = k$ is the parent of i are excluded. In this case, the link ia will not be used in $P^{-k}(c;i,j)$; thus, we can ignore neighbors in this category.

Let b be the neighbor of i on $P^{-k}(c;i,j)$; *i.e.*, the link ib is the first link on the lowest-cost k-avoiding path from i to j. We claim that, for this neighbor, the upper bounds in the previous inequalities are tight:

Lemma 1. *Let ib be the first link on $P^{-k}(c;i,j)$. Then, the corresponding inequality (2)-(5) attains equality for b.*

Proof. We can consider each of the four cases separately.

- **Case (i)**: Given that $P^{-k}(c;i,j)$ goes through its parent, it follows that b is not k. Now, the suffix of $P^{-k}(c;i,j)$ from b to j is a k-avoiding path from b to j; it follows that it must be $P^{-k}(c;b,j)$. Thus, in this case, we have $\text{Cost}(P^{-k}(c;i,j)) - \text{Cost}(P^{-k}(c;b,j)) = c_b = c(b,j) - c(i,j)$, and so $p_{ij}^k = p_{bj}^k$.
- **Case(ii)**: If $P^{-k}(c;i,j)$ passes through a child b, then the same reasoning as in case (i) above shows that it must have cost exactly c_b more than $P^{-k}(c;b,j)$, and hence we get $p_{ij}^k = p_{bj}^k + c_i + c_b$.
- **Case(iii)**: Again, a similar argument to the previous cases shows that if $P^{-k}(c;i,j)$ passes through b, it must contain $P^{-k}(c;b,j)$, and hence have cost exactly c_b greater than that of $P^{-k}(c;b,j)$. Hence, Inequality 4 holds with equality.
- **Case(iv)**: In this case, the lowest-cost k-avoiding path through b must contain $P(c;b,j)$, as that is the lowest-cost path from b to j. Thus, $\text{Cost}(P^{-k}(c;i,j)) = c(b,j) + c_b$, and so Inequality 5 is exact.

□

Inequalities (2)-(5) and Lemma 1 together mean that p_{ij}^k is exactly equal to the minimum, over all neighbors a of i, of the right-hand side of the corresponding inequality.

Thus, we have the following distributed algorithm to compute the payment values:

The Algorithm

Consider each destination j separately. The BGP table at i contains the LCP to j:

$$P(c;i,j) \equiv v_s, v_{s-1}, \cdots, v_0 = j,$$

and the cost of this path, $c(i,j)$, where $v_s, v_{s-1}, \cdots, v_0$ are the nodes on the LCP to j and $c(i,j) = \sum_{r=1}^{s} c_{v_r}$.

Note that each node can infer from the routing tables it receives from its neighbors whether a is its parent, child, or neither in the tree $T(j)$, for each neighbor a.

At the beginning of the computation, all the entries of $p_{ij}^{v_r}$ are set to ∞. Whenever any entry of this price array changes, the array and the path $P(c;i,j)$ are sent to all neighbors of i. As long as the network is static, the entries decrease monotonically as the computation progresses. If the network is dynamic, price computation (and, as explained above, convergence) must start over whenever there is a route change.

When node i receives an updated price from a neighbor a, it performs the following updates to its internal state.

- If a is i's parent in $T(j)$, then i scans the incoming array and updates its own values if necessary:

$$p_{ij}^{v_r} = \min(p_{ij}^{v_r}, p_{aj}^{v_r}) \quad \forall r \leq s-1$$

- If a is a child of i in $T(j)$, i updates its payment values using

$$p_{ij}^{v_r} = \min(p_{ij}^{v_r}, p_{aj}^{v_r} + c_i + c_a) \quad \forall r \leq s$$

- If a is neither a parent nor a child, i first scans a's updated path to find the nearest common ancestor v_t. Then i performs the following updates:

$$\forall r \leq t \quad p_{ij}^{v_r} = \min(p_{ij}^{v_r}, p_{aj}^{v_r} + c_a + c(a,j) - c(i,j))$$
$$\forall r > t \quad p_{ij}^{v_r} = \min(p_{ij}^{v_r}, c_k + c_a + c(a,j) - c(i,j))$$

The algorithm is summarized in Figure 3.

Correctness of the algorithm

Inequalities (2)-(5) can be used to show that the algorithm never computes a value p_{ij}^k that is too low. In order to show that the p_{ij}^k values will ultimately converge to their true values, we observe that, for every node s on $P^{-k}(c;i,j)$, the suffix of $P^{-k}(c;i,j)$ from s to j is either $P(c;s,j)$ or $P^{-k}(c;s,j)$. It follows that, in general, the path $P^{-k}(c;i,j)$ consists of a sequence of nodes $[v_l, v_{l-1}, \cdot\cdot, v_1, u_m, u_{m-1}, \cdot\cdot, u_1]$ such that, for each u_x, $P(c;u_x,j)$ is the suffix $[u_x, u_{x-1}, \cdot\cdot, u_1]$, and, for each v_y, $P^{-k}(c;v_y,j)$ is the suffix $[v_y, v_{y-1}, \cdot\cdot, v_1, u_m, u_{m-1}, \cdot\cdot, u_1]$. Note that, once the LCPs are computed, u_m will know the correct $P(c;u_m,j)$ and cost $c(a,j)$. This information will be sent to v_1 in the next update message from u_m to v_1; thus, v_1 will then be able to compute the correct $P^{-k}(c;v_1,j)$ and $p_{v_1 j}^k$. Proceeding by induction on y, we can show that i will ultimately have all the correct p_{ij}^k values.

In fact, the preceding inductive argument shows that all prices will be stable after d' stages, where d' is the maximum over all i, j, k, of the number of nodes on $P^{-k}(c;i,j)$. In general, d' can be much higher than the lowest-cost diameter d of a graph. However, we don't find that to be the case for the current AS graph, as we explain in Section 7.

```
Initialize ()
{
   /* Compute routes, initialize payments */
   for each destination j
      Compute P(c;i,j) and c(i,j)
      [v_s, v_{s-1}, ..., v_1] = P(c;i,j)
      for each node k on P(c;i,j)
         p^k_ij := ∞
}

Update (a, j, c(a,j), P(c;a,j), [p^{u_1}_aj, p^{u_2}_aj, ..., p^{u_l}_aj])
{
   /* Called when an UPDATE message    */
   /* for destination j is received    */
   /* from neighbor a.                 */
   /* u_1, u_2, ... u_l are the transit nodes  */
   /* on the route P(c;a,j) from a to j  */

   modified := FALSE

   if a is on P(c;i,j)            /* parent */
      /* u_r = v_r, for r = 1,2,...l */
      for each k in {v_1, v_2, ... v_l}
         if p^k_ij >    p^k_aj
            p^k_ij :=   p^k_aj
            modified := TRUE

   else if i on P(c;a,j)           /* child */
      /* u_r = v_r, for r = 1,2,...(l-1) */
      for each k in {v_1, v_2, ... v_{l-1}}
         if p^k_ij >    p^k_aj + c_a + c_i
            p^k_ij :=   p^k_aj + c_a + c_i
            modified := TRUE

   else        /*neither parent nor child*/
      t := largest index such that u_t = v_t
      for each k in {v_1, v_2, ... v_t}
         if p^k_ij >    p^k_aj + c_a + c(a,j) - c(i,j)
            p^k_ij :=   p^k_aj + c_a + c(a,j) - c(i,j)
            modified := TRUE
      for each k in {v_{t+1} ... v_s}
         if p^k_ij >    c_k + c_a + c(a,j) - c(i,j)
            p^k_ij :=   c_k + c_a + c(a,j) - c(i,j)
            modified := TRUE

   if modified = TRUE
      /* Send UPDATE message to neighbors*/
      for each neighbor b of i
         send UPDATE (i, j, c(i,j), P(c;i,j),
                      [p^{v_1}_ij, p^{v_2}_ij, ..., p^{v_s}_ij]) to b
}
```

Fig. 3. Price-computation algorithm run by AS i

Convergence time[9]

Up to this point, we have assumed for simplicity that the prices computation begins only after the LCPs have been found. In reality, however, the algorithm can start to compute prices even before the routes have stabilized. This leads to the following bound:

Lemma 2. *Let $d_i = \max\{|P(c;i,j)|, |P^{-k}(c;i,j)|\}$, where $|P|$ denotes the number of hops in path P. Then, after the first d_i stages, i knows the correct path $P(c;i,j)$, and the correct price p^k_{ij}.*

Proof. The intuition behind this proof is as follows: The critical information that i needs to compute the correct price p^k_{ij} is the cost of $P(c;i,j)$ and the cost of $P^{-k}(c;i,j)$. (The cost c_k can be distributed with LCPs to k, and so it will be known to i before or at the same time as the cost of $P(c;i,j)$.) After these costs have been discovered, the price p^k_{ij} will not change. The key observation is that, for both $P(c;i,j)$ and $P^{-k}(c;i,j)$, all suffixes of the path are also LCPs or minimum-cost k-avoiding paths; moreover, this is true even at intermediate stages of the computation. Using this, we can show that the costs along these paths will be propagated further in each stage and hence will reach i in d_i stages.

We now formalize this argument into an inductive proof. First, observe that both the LCP costs and the prices never increase as the computation proceeds. Now, if there is an r-hop path from i to j with cost α, then, after r stages, we must have $c(i,j) \leq \alpha$. It follows that i discovers the LCP in $|P(c;i,j)| \leq d_i$ stages from the beginning.

Now, suppose $|P^{-k}(c;i,j)| = r$. We can write $P^{-k}(c;i,j)$ as $v_r, v_{r-1}, \cdots, v_1, j$, where $v_r = i$. Let $c'(v_m, j)$ denote the total cost *along this path* from v_m to j. We show by induction on m that, after m stages, node v_m satisfies one of the following two conditions:

1. $c(v_m, j) = c'(v_m, j)$, and the current LCP from v_m to j does not pass through k.
2. The current LCP from v_m to j passes through k, and the current $p^k_{v_m j} = c'(v_m, j) - c(v_m, j)$.

The base case for v_1 is easy: After 1 stage of computation, condition 1 is clearly satisfied. Suppose it is true for all $r \leq (m-1)$. After $(m-1)$ stages, v_{m-1} satisfies one of the two conditions; we consider the two cases separately:

- **Case (i)**: v_{m-1} satisfies condition 1
  In this case, in the mth stage v_m will receive an advertised path from v_{m-1} that has cost $c'(v_m, j)$ and does not pass through k. If this is the lowest-cost path to j that v_m has seen, then v_m will satisfy condition 1 at the end of this stage. If not, then the current LCP from v_m to j must pass through k, or else $P^{-k}(c;i,j)$ would not pass through v_{m-1}. In this scenario, the path advertised by v_{m-1} must be the lowest-cost k-avoiding path from v_m to j; hence after this stage condition 2 will be satisfied.
- **Case (ii)**: v_{m-1} satisfies condition 2
  In this case, v_{m-1} advertises a path of cost less than $c'(v_m, j)$ to v_m. Thus, the LCP from v_m to j after the mth stage must pass through k, or else it would be part of the lowest-cost k-avoiding path from i to j. Now, the advertisement from v_{m-1} (which includes the price $p^k_{v_{m-1}j}$) allows v_m to infer that there is a candidate k-avoiding path of cost $c'(v_m, j)$; this is the lowest-cost k-avoiding path, and so condition 2 will be satisfied.

Extending this inductive argument to $v_r = i$, we see that, after r stages, i will know of the existence of a k-avoiding path of cost $c'(v_r, j)$. Thus, after d_i stages, i will know the correct LCP to j and its cost, as well as the cost of the lowest-cost k-avoiding path to j, and so it will compute the correct price p^k_{ij}. □

Corollary 1. *After $\max(d, d')$ stages, every node has the correct LCPs and prices.*

□

Using the Prices

At the end of the above price computation, each node i has a full set of prices p^k_{ij}. The next question is how we can use these prices actually to compute the revenue due each node.

The simplest approach is to have each node i keep running tallies of owed charges; that is, every time a packet is sent from source i to a destination j, the counter for each node $k \neq i, j$ that lies on the LCP is incremented by p^k_{ij}. This would require $\mathcal{O}(n)$ additional storage at each node. At various intervals, nodes can send these quantities in to whatever accounting and charging mechanisms are used to enforce the pricing scheme. We assume that the submission of these running totals is done infrequently enough that the communication overhead can be easily absorbed.

In summary, we have:

Theorem 2 *Our algorithm computes the VCG prices correctly, uses routing tables of size $O(nd)$ (i.e., imposes only a constant-factor penalty on the BGP routing-table size), and converges in at most $\max(d, d')$ stages.*

7 Conclusions and open problems

In this paper, we considered some incentive issues that arise in interdomain routing. We asked what payments are needed to elicit truthful revelation of AS transit costs and whether they can be efficiently computed. We showed that the payments take the form of a per-packet price and that they can be computed using a simple extension to BGP that requires only a constant factor increase in communication costs. There are several promising directions for additional research.

Our results are based on a simple model in which ASs attempt to minimize per-packet transit costs. In practice, ASs have more complex costs and route preferences that are embodied in their *routing policies*. We are currently extending the algorithmic-mechanism-design approach to handle more general routing policies. The main challenge is to find

[9] The convergence-time analysis given here is a modest improvement over the one in [5]. There, it was shown that the correct routes and prices are known after $(d + d')$ stages.

a class of routing policies that is expressive enough to model real policies, yet simple enough to allow for a practical distributed algorithmic mechanism. Some preliminary results in this line of research have been reported in Feigenbaum, Sami, and Shenker [7]. Another interesting direction is to augment the network model with link or node capacities in order to tackle the problem of routing in congested networks. This is particularly natural because it seems plausible that transit traffic imposes costs only in the presence of congestion.

One important issue that is not yet completely resolved is the need to reconcile the strategic model with the computational model. On the one hand, we acknowledge that ASs may have incentives to lie about costs in order to gain financial advantage, and we provide a strategyproof mechanism that removes these incentives. On the other hand, it is these very ASs that implement the distributed algorithm we have designed to compute this mechanism; even if the ASs input their true costs, what is to stop them from running a different algorithm that computes prices more favorable to them? This issue does not arise in [16,12], where the mechanism is a centralized computational device that is distinct from the strategic agents who supply the inputs, or in previous work on distributed multicast cost-sharing mechanisms [6,1,4], where the mechanism is a distributed computational device (*i.e.*, a multicast tree) that is distinct from the strategic agents (who are users resident at various nodes of the tree but not in control of those nodes). If ASs are required to sign all of the messages that they send and to verify all of the messages that they receive from their neighbors, then the protocol we gave in Section 6 can be modified so that all forms of cheating are detectable [14]. Achieving this goal without having to add public-key infrastructure (or any other substantial new infrastructure or computational capability) to the BGP-based computational model is the subject of ongoing further work. In two recent papers, Shneidman and Parkes [22,23] have suggested using redundant communication to tackle this problem.

There is also the issue of overcharging. VCG mechanisms have been criticized in the literature because there are graphs in which the total price along a path, *i.e.*, the sum of the per-packet payments along the path, is much more than the true cost of the path. Examples of this phenomenon were given in Section 4. In the worst case, this total path price can be arbitrarily higher than the total path cost [2]. Although this is undesirable, it may be unavoidable, because VCG mechanisms are the only strategyproof pricing mechanisms for protocols that always route along LCPs. In addition, our distributed algorithm has a convergence time (measured in number of stages) of d', whereas BGP's convergence time for the LCP computation alone would be d; in the worst case, $\frac{d'}{d}$ could be $\Omega(n)$. These are serious problems that could undermine the viability of the pricing scheme we present here. Thus, we ask whether these problems occur in practice.

To provide a partial answer to this question, we looked at the prices that would be charged on the current AS graph if we assumed that all transit costs were the same. Out of a 9107-node AS graph, reflecting a recent snapshot of the current Internet[10], we selected a 5773-node biconnected subset. We then computed d, d', and the payments that would result from our pricing scheme, assuming a transit cost of 1 for each node. We find that $d = 8$ and $d' = 11$, and so the convergence time of the pricing algorithm is not substantially worse than that of BGP. The highest transit node price was 9, and, with uniform traffic between all pairs, the mean node payment is 1.44. In fact, 64% of the node prices were 1, and 28% of them were 2. Thus, overcharging appears not to be a problem in this case, reflecting the high connectivity of the current Internet. Of course, the values of d and d' and the overcharging margin would be different with non-uniform transit costs; however, we expect them to exhibit similar trends towards low d, d', and overcharging margin.

It would be interesting to ask whether this is because of the incentive issues in AS-graph *formation*. In this paper, we merely looked at the routing aspects of a given AS graph. However, if one considers the incentives present when an AS decides whether or not to connect to another AS, the resulting transit prices would be a serious consideration. In particular, we conjecture that high node prices will not be sustainable in the Internet precisely because, if present, they would give an incentive for another AS to establish a link to capture part of that revenue, thereby driving down the transit prices. We are currently working on models of network formation to verify this conjecture.

[10] These data were taken from Route Views [20], which collates BGP tables from many sites across the Internet.

8 Acknowledgments

We thank Ramesh Govindan for providing us with a recent AS graph and for teaching us about the intricacies of BGP. We also thank Kunal Talwar for helpful discussions of the role of incentives in AS-graph formation, and Gauri Shah for suggesting improvements to the convergence time analysis.

References

1. A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and Collusion in Multicast Cost Sharing. *Games and Economic Behavior* **47** (2004), pages 36–71.
2. A. Archer and E. Tardos. Frugal path mechanisms. To appear in *Journal of Algorithms*. Extended abstract appeared in *Proceedings of 13th Symposium on Discrete Algorithms*, ACM Press/SIAM, New York/Philadelphia, pages 991–999, 2002.
3. E. Clarke. Multipart pricing of public goods. *Public Choice* **11** (1971), pages 17–33.
4. J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Hardness Results for Multicast Cost Sharing. *Theoretical Computer Science.* **304** (2003), pages 215–236.
5. J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *Proceedings of the 21st Symposium on Principles of Distributed Computing*, ACM Press, New York, pages 173–182, 2002.
6. J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences* **63** (2001), pages 21–41.
7. J. Feigenbaum, R. Sami, and S. Shenker. Mechanism Design for Policy Routing In *Proceedings of the 23rd Symposium on Principles of Distributed Computing*, ACM Press, New York, pages 11–20, 2004.

8. A. Fiat, A. Goldberg, J. Hartline, and A. Karlin. Competitive Generalized Auctions. In *Proceedings of the 34th Symposium on Theory of Computing*, ACM Press, New York, pages 72–81, 2002.
9. J. Green and J. Laffont. Incentives in public decision making. In **Studies in Public Economics**. Volume 1, North Holland, Amsterdam, pages 65–78, 1979.
10. T. G. Griffin and G. Wilfong. An analysis of BGP convergence properties. In *Proceedings of SIGCOMM '99*, ACM Press, New York, pages 277–288, 1999.
11. T. Groves. Incentives in teams. *Econometrica* **41** (1973), pages 617–663.
12. J. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth? In *Proceedings of the 42nd Symposium on the Foundations of Computer Science*, IEEE Computer Society Press, Los Alamitos, pages 129–140, 2001.
13. A. Mas-Colell, M. Whinston, and J. Green. **Microeconomic Theory**, Oxford University Press, New York, 1995.
14. J. Mitchell, R. Sami, K. Talwar, and V. Teague. Private communication, December 2001.
15. D. Monderer and M. Tennenholtz. Distributed Games: From Mechanisms to Protocols. In *Proceedings of the 16th National Conference on Artificial Intelligence* , pages 32–37, 1999.
16. N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior* **35** (2001), pages 166-196.
17. D. Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, ACM Press, New York, pages 148–157, 1999.
18. D. Parkes and L. Ungar. Iterative combinatorial auctions: Theory and practice. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 71–81, 2000.
19. T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal of the ACM* **49** (2002), pages 236–259.
20. Route Views. University of Oregon Route Views Project. `http://www.routeviews.org`
21. T. Sandholm. Distributed rational decision making. In G. Weiss, editor, *Multiagent systems: A Modern Introduction to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, pages 201–258, 1999.
22. J. Shneidman and D. Parkes. Using redundancy to improve robustness of distributed mechanism implementations. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, ACM Press, New York, pages 276–277, 2003.
23. J. Shneidman and D. Parkes. Specification Faithfulness in Networks with Rational Nodes. In *Proceedings of the 23rd Symposium on Principles of Distributed Computing*, ACM Press, New York, pages 88–97, 2004.
24. H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proceeding of SPIE ITCom 2001*, SPIE Press, Bellingham, pages 19–24, 2001.
25. W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance* **16** (1961), pages 8–37.
26. M. Wellman. A market-oriented programming environment and its applications to distributed multicommodity flow problems. *Journal of AI Research* **1** (1993), pages 1–23.
27. M. Wellman, W. Walsh, P. Wurman, and J. Mackie-Mason. Auctions for decentralized scheduling. *Games and Economic Behavior* **35** (2001), pages 271–303.