Sharing the Cost of Multicast Transmissions[1]

Joan Feigenbaum[2]

Computer Science Dept., Yale University, P. O. Box 208285, New Haven, CT 06520-8285

E-mail: joan.feigenbaum@yale.edu

and

Christos H. Papadimitriou[3]

Computer Science Division, U. C. Berkeley, Berkeley, CA 94704

E-mail: christos@cs.berkeley.edu

and

Scott Shenker

ACIRI, International Computer Science Institute, 1947 Center Street, Suite 600, Berkeley, CA 94704-1198

E-mail: shenker@icsi.berkeley.edu

We investigate cost-sharing algorithms for multicast transmission. Economic considerations point to two distinct mechanisms, *marginal cost* and *Shapley value*, as the two solutions most appropriate in this context. We prove that the former has a natural algorithm that uses only two messages per link of the multicast tree, while we give evidence that the latter requires a quadratic total number of messages. We also show that the welfare value achieved by an optimal multicast tree is NP-hard to approximate within any constant factor, even for bounded-degree networks. The lower-bound proof for the Shapley value uses a novel algebraic technique for bounding from below the number of messages exchanged in a distributed computation; this technique may prove useful in other contexts as well.

---

[1]These results appeared in preliminary form in the Proceedings of the 2000 ACM Symposium on Theory of Computing.

[2]Most of this work was done while the author was a member of the Information Sciences Research Center of AT&T Labs, Florham Park, NJ.

[3]Supported in part by a grant from the National Science Foundation.

1. INTRODUCTION

The traditional form of routing provided by the Internet is *unicast* routing; each packet sent from a source is delivered to a single receiver. To send the same packet to multiple sites, a source has to send a separate copy of this packet to each receiver. This results in many identical packets traversing the network links close to the source. To avoid this waste of bandwidth, a different form of routing, called *multicast* routing [4], has been developed. Multicast routing creates a directed tree connecting the source to all the receivers; when a packet reaches a branch point in the tree, the router duplicates the packet and then sends a copy over each downstream link. Thus, a source can reach multiple receivers without sending duplicate copies of the packet over any link. Because of this increased transmission efficiency, multicast routing will greatly facilitate the transmission of highly popular content such as live concerts and movies.

Such applications are likely to be significant sources of revenue. However, multicasting high quality audio and video to a large population is likely to incur significant costs. Unlike the unicast case, the bandwidth used by a multicast transmission is not directly attributable to a single receiver.[4] Thus, one must find a way to distribute the cost among the various receivers. A *cost-sharing* mechanism determines which users receive the multicast transmission and how much they are charged. We let $x_i \geq 0$ denote how much user i is charged and σ_i denote whether user i receives the transmission; $\sigma_i = 1$ if the user receives the multicast transmission, and $\sigma_i = 0$ otherwise.

We consider a particular multicast transmission of, say, a movie and assume that user i derives utility u_i from the movie. A user's individual *welfare*, her overall happiness after seeing the movie (or not) and paying the resulting cost, is given by $w_i = \sigma_i u_i - x_i$. The vectors σ and x are functions of the utility vector u; a cost-sharing mechanism is defined by the functions $x(u)$ and $\sigma(u)$. However, the network does not *a priori* know the u_i, because they are properties of the users, not of the network; it must rely on the users to report these values. In the applications we are considering, such as the transmission of a movie, the users are independent and have little reason to cooperate with each other or with the network. Therefore, we cannot assume that users will truthfully report their utilities to the network; instead, we assume that users are selfish and will report the values that maximize their individual welfare w_i. The network can discourage such deception by using *strategyproof* cost-sharing mechanisms in which each user maximizes her welfare by truthfully revealing u_i; that is, $w_i(u) \geq w_i(u|^i v_i)$ for any v_i and all i. (Here, $(u|^i v_i)_j = u_j$, for $j \neq i$, and $(u|^i v_i)_i = v_i$.) Strategyproofness is a very strong condition; with a few additional natural constraints described in Section 2, it leads us to two natural strategyproof cost-sharing mechanisms: Marginal Cost (MC) and Shapley Value (SH) (see [22]).

[4] There are many contexts in which it would be natural for the source to foot the bill (*e.g.*, a sales presentation) or for the bill to be shared in some fashion between source and receiver (*e.g.*, a teleconference). However, we are restricting our attention here to cases in which the receiver is responsible for paying for the transmission. This is the most challenging, if perhaps not the most common, case. It may also be the case most appropriate for mass-market content.

A cost-sharing *mechanism* is merely a pair of formulas for $x(u)$ and $\sigma(u)$. However, computing these quantities is not a trivial task, because the u_i (and the network costs, as we describe in Section 2) are spread throughout the network, and the results of the computation, the $x_i(u)$ and $\sigma_i(u)$, must be delivered to the individual users. Therefore, once given a cost-sharing *mechanism*, we need to define the distributed cost-sharing *algorithm* or *protocol* that implements that mechanism. We do not require that the algorithm end with some network router knowing the results of the entire computation, rather that all users are informed of their individual allocations x_i and σ_i. Because these cost-sharing algorithms might deal with extremely large user populations, it is crucial that they not impose significant computational and communication burdens on the network. In particular, we care about how these computational and communication burdens grow with the size of the user population. If the computational burden on each network router is reasonable, as it is for the algorithms we discuss, then there is little motivation to reduce the computational complexity further, because the computing resources of a router are not fungible. However, there is intense competition for link bandwidth; thus, minimizing the communication burden of these distributed cost-sharing algorithms is of great interest. In this paper, we focus on the communication burden of algorithms for implementing the MC and SH cost-sharing mechanisms. We prove a lower bound suggesting that any cost-sharing algorithm implementing the Shapley Value mechanism must send a linear number of messages over each of a linear number of links, a quadratic number of messages overall.[5] By contrast, we give a natural algorithm that implements the Marginal Cost mechanism using only two messages traversing each link (one in each direction).

This paper brings together two separate concerns. The incentive considerations of strategyproofness restrict the class of cost-sharing mechanisms that can be employed. Similarly, complexity considerations constrain the set of algorithms that can be used to implement these mechanisms. One must find mechanisms that both have the desired incentive properties and also can be feasibly implemented. Our results suggest that, although both the Shapley Value and Marginal Cost mechanisms have the proper incentive properties (and are, in a well defined sense made clear in the next section, the two most appropriate mechanisms for this problem), only the Marginal Cost mechanism can be feasibly implemented.

The remainder of this paper is organized as follows. Section 2 contains some technical preliminaries and a brief discussion of related work. We present our results on the Marginal Cost mechanism in Section 3. Section 5 contains a lower bound (in a restricted computational model) for algorithms implementing the Shapley Value mechanism. In our model, we assume that the multicast tree is fixed and given to us; if, instead, we are given a graph and the task of finding the tree that optimizes welfare, we show in Section 4 that the welfare value achieved by an optimal multicast tree is NP-hard to approximate within any constant. We conclude in Section 6 with a brief summary and discussion of open problems.

[5] While we focus primarily on message (not bit) complexity, we do, in our lower bound proof in Section 5, prevent concatenation of messages and other tricks by requiring that the operations on these messages be linear.

2. TECHNICAL PRELIMINARIES

2.1. Network Model

Consider a user population P, a set of network nodes N, and a set of bidirectional network links L. Each user $i \in P$ resides[6] at some network location $\alpha \in N$. In this paper, we restrict our attention to a particular multicast flow emanating from a source located on network node $\alpha_s \in N$. The multicast routing infrastructure, given any set of receivers $R \subseteq P$, constructs a tree (the *multicast tree*) $T(R) \subseteq L$ rooted at α_s and connecting α_s to the nodes at which users in R reside. In general, the tree $T(R)$ can depend arbitrarily on the subset R, but, for most of this paper (the exception being Section 4), we shall assume that there is a *universal* tree $T(P)$ and that, for each subset $R \subseteq P$, the multicast tree $T(R)$ is merely the minimal subtree of $T(P)$ required to reach the elements in R. This assumption implies that the multicast routing infrastructure chooses, for each user i, a fixed path $T(i)$ from the source to user i and that, for every set R of receivers, the delivery tree is merely the union of these fixed paths: $T(R) = \cup_{i \in R} T(i)$. This approach of building the multicast tree out of a nonlooping[7] set of unicast paths is relatively simple to implement and makes the multicast tree quite stable as receivers join and leave; both of these properties – ease of implementation and stability – are extremely important in practice. We adopt this approach for the bulk of our paper, because it represents the design philosophy embedded in essentially all current multicast routing proposals (see, for example, [1, 4, 5, 13, 27]). However, this approach does impose significant limitations on the structure of $T(R)$, because it forces the data path from the source to a particular user to be independent of which other users are present; in particular, this approach precludes the use of more theoretically familiar structures like Steiner trees. The form of multicast routing we consider here may thus lead to suboptimal trees. We address the possibility of using optimal trees in Section 4.

Each link $l \in L$ has an associated cost $c(l) \geq 0$ that is known by the nodes on each end. The cost of the tree $T(R)$ reaching a set of receivers R is $c(T(R))$, and the overall welfare, or net worth, is $NW(R) = u_R - c(T(R))$, where $u_R = \sum_{i \in R} u_i$ and $c(T(R)) = \sum_{l \in T(R)} c(l)$. Note that the overall welfare is not the sum of the individual welfares and does not depend on the cost shares. The overall welfare measures the total benefit of providing the multicast transmission (the sum of the utilities minus the cost); the cost shares are merely transfers between the receivers and the network but do not change the overall level of welfare. The fact that $T(R) = \cup_{i \in R} T(i)$ ensures that the cost $c(T(R))$ is a nondecreasing and submodular function of the set R: $c(T(R+i)) \geq c(T(R))$ and $c(T(R_1)) + c(T(R_2)) \geq c(T(R_1 \cup R_2)) + c(T(R_1 \cap R_2))$, for all $R, R_1, R_2 \subseteq P$. In Section 4 below, we examine the effect of dropping the restriction $T(R) = \cup_{i \in R} T(i)$.

2.2. Cost-Sharing Mechanisms

[6] The nodes represent routers, and a user's being *resident* at a node means that the node is the first-hop router for that user.

[7] We say that a set of unicast paths is *nonlooping* if the union of any subset of those paths forms a tree.

Cost-sharing mechanisms are defined by the functions $x_i(u)$ and $\sigma_i(u)$. For a given cost-sharing mechanism, define $R(u) = \{i \in P | \sigma_i(u) = 1\}$, the *receiver set*, to be the set of users selected to receive the transmission at a given utility profile u. Similarly, let $W(u) \equiv NW(R(u))$ be the welfare resulting from the mechanism at utility vector u. We consider only strategyproof cost-sharing mechanisms, and so we must have $w_i(u) \geq w_i(u|^i v_i)$, for all u, i, and v_i. We also impose several additional basic requirements:

- No Positive Transfers (NPT): $x_i(u) \geq 0$. This precludes *paying* receivers to receive the transmission. We are only considering situations in which the users pay for the right to participate.
- Voluntary Participation (VP): $w_i(u) \geq 0$. Users are always free to not receive the transmission and not be charged, which would result in an individual welfare of zero; the network can't force a user to be worse off than this baseline option. Note that this implies that $x_i = 0$ whenever $\sigma_i = 0$, *i.e.*, that the set of users who pay is a (not necessarily proper) subset of the set who actually receive the transmission.
- Consumer Sovereignty (CS): For all u, $\sigma(u|^i v_i) = 1$ for sufficiently large v_i. The cost-sharing algorithm cannot arbitrarily exclude any users; the network has to allow users to receive the transmission if they are willing to pay a sufficiently high cost.

There are many strategyproof cost-sharing mechanisms that satisfy the above requirements, some of them quite impractical. For instance, one could send to the highest bidder (breaking ties arbitrarily), deny all other users, and charge the chosen user the second highest reported utility; this would result in all users but one having zero welfare and could result in negative overall welfare if the network costs were sufficiently high. To exclude such alternatives, we find it useful to consider two other requirements:

- Budget-balance: $\sum_{i \in P} x_i(u) = c(T(R(u)))$. This requires that the revenue raised from the receivers cover the cost of the transmission exactly.
- Efficiency: $NW(R(u)) \geq NW(R)$ for all $R \subseteq P$. This requires that the receiver set maximize the overall benefit of the network. A set that maximizes $NW(R)$ is called an *efficient set*.

It is a classical result in game theory [9, 28] that, in this context, these two requirements are mutually exclusive; there are no strategyproof cost-sharing mechanisms that are both budget-balanced and efficient. It is easy to see, and is shown in [22], that there is essentially only one strategyproof cost-sharing mechanism that both satisfies NPT and VP and is efficient: the Marginal Cost mechanism.[8] The MC mechanism is a special case of a more general class, the Vickrey-Clarke-Groves [2, 10, 11, 36] mechanisms, and is defined as follows. Let $R^*(u)$ denote the largest efficient set; this is well defined, because submodularity of the cost function guarantees that the union of two efficient sets is also an efficient set. Let $\sigma_i(u) = 1$, for

[8] More precisely, all such strategyproof mechanisms are welfare-equivalent to the MC mechanism in that the resulting individual welfares w_i are the same. The only degree of freedom in deviating from the MC mechanism is that one can exclude users who don't change the overall welfare and whose own individual welfare is zero.

all $i \in R^*(u)$, and $\sigma_i(u) = 0$, for all $i \notin R^*(u)$; thus, $W(u) = NW(R^*(u))$. The cost shares are given by:

$$x_i = u_i \sigma_i(u) - \big(W(u) - W(u|^i 0)\big) \tag{1}$$

This mechanism gives user i a welfare $w_i = W(u) - W(u|^i 0)$ that represents the marginal contribution to the overall welfare provided by her having nonzero utility for the transmission. Unfortunately, this mechanism is not budget-balanced. It never runs a budget surplus but can run a budget deficit; in fact, in many cases, it raises no revenue at all [22].

While the requirement of efficiency picked out a single natural cost-sharing mechanism, the requirement of budget-balance leaves many possible mechanisms. By imposing the stronger condition of group strategyproofness (no group of users can increase their welfares by lying about their utilities), one can completely characterize the class of possible cost-sharing mechanisms [21, 22]. Each mechanism is defined by a function $f : 2^P \mapsto \Re_{\geq 0}^{|P|}$ with the properties that $\sum_i f_i(R) = c(T(R))$ and $f_i(R + j) \leq f_i(R)$ for all $i, j \in P$. One can use the function f to define $x(u)$ and $\sigma(u)$ in an iterative fashion. First, set $\sigma_i^{(1)}(u) = 1$. At each step k, let $R^{(k)}(u) = \{i | \sigma_i^{(k-1)}(u) = 1\}$, $x_i^{(k)}(u) = f_i(R^{(k)}(u))$, $\sigma_i^{(k)}(u) = 1$ if $u_i \geq x_i^{(k)}(u)$, and $\sigma_i^{(k)}(u) = 0$ if $u_i < x_i^{(k)}(u)$. The sets $R^{(k)}$ form a monotonic sequence, $R^{(k)} \subseteq R^{(k-1)}$, and thus converge after a finite number of steps to some set $\hat{R}(u)$; the resulting values for $x(u)$ and $\sigma(u)$ define the cost-sharing mechanism, and $\hat{R}(u)$ is the receiver set. The Shapley Value mechanism uses the Shapley Value [31, 35] for the function f; the general definition of the Shapley Value [31] applied to our network cost-sharing problem yields the following formula:

$$f_i(R) = \sum_{\tilde{R} \subseteq R - i} \frac{|\tilde{R}|!(|R| - |\tilde{R}| - 1)!}{|R|!} [c(T(\tilde{R} \cup i)) - c(T(\tilde{R}))]$$

This formula is quite forbidding, but it has a simple intuitive explanation: The cost of a link l is shared equally by all receivers who are downstream of the link.

None of these budget-balanced mechanisms maximizes the overall welfare. Out of all of them, the Shapley Value mechanism minimizes the worst-case welfare loss; that is, the quantity $Max_u[NW(R^*(u)) - NW(R(u))]$ produced by the Shapley value mechanism is strictly smaller than the corresponding quantity produced by other mechanisms in this class [22]. If we insist on budget-balance as a requirement, it seems natural to choose the mechanism that minimizes the resulting loss of efficiency; the Shapley Value is thus a natural choice in this class of budget-balanced cost-sharing mechanisms.

2.3. Computational Model and Costs

For a cost-sharing algorithm to be practical, both communication and local computation should require only modest resources, because cost-sharing is not the *raison d'être* of the network —it is done only to support the primary goal of transmitting content to receivers in a financially viable manner. Hence, analyzing the complexity of algorithms for implementing cost-sharing, especially in terms of the

communication burden they impose on the network, is important. By "communication burden," we mean messages transmitted over links in L, not local communication among users resident at the same node. In each round of the algorithm, a node $\alpha \in N$ may receive one message from each of its neighbors in $T(P)$, compute one or more functions of these messages, values supplied by its resident users, and values stored in previous rounds, and then send one message to each of its neighbors in $T(P)$. The resident users may have to execute a local-area network algorithm to accomplish a round of computation, but the algorithm-design problem this raises is orthogonal to the problem we consider here, and its communication burden is taken to be zero.

An instance of a cost-sharing problem has size $n + p + m$, where $n = |T(P)|$, $p = |P|$, and m is the total size of the numerical input $\{c(l)\}_{l \in L} \cup \{u_i\}_{i \in P}$. Ideally, the *total* number of messages sent by a cost-sharing algorithm should be $O(n)$, and the *maximum*, over all $l \in L$, of the number of messages sent over l should be $O(1)$. Note that we care about "hot spots" in bandwidth utilization as well as total bandwidth utilization; for example, an algorithm in which the total number of messages is $O(n)$ but $\Omega(n)$ of the messages "bunch up" on one or a few links is unsatisfactory.

In this paper, our primary focus is communication, rather than local computation, but we do not completely disregard local computation. For example, a mechanism that is NP-hard even to approximate closely is obviously impractical; it could not be implemented efficiently even if communication costs associated with the distributed nature of the inputs and outputs were zero and *a fortiori* cannot be implemented efficiently if these costs are nonzero, as they are in our model.

Similarly, while we focus primarily on the *number* of messages that an algorithm sends, we also require that message *size* be reasonable. It would not be satisfactory, for example, to implement a polynomial-time mechanism as follows: Each node, after receiving a message from each of its children, concatenates all of the received messages, all of the utilities of players resident at it, and the cost of the link connecting it to its parent and then sends the result to its parent; when the root α_s of $T(P)$ has received all of the utilities and link costs, it computes σ and x and sends them to its children so that the needed values can be "peeled off" as they reach the appropriate nodes, and the remaining values can be sent further down the tree. This trivial algorithm would achieve the goal of $O(1)$ messages per link, but the maximum size of a message would be $\Omega(m)$, which is too big. Message size is not a major issue in this paper, but we revisit it in context where appropriate in what follows.

We use the term "network complexity" to capture these four aspects of a cost-sharing algorithm's worst-case performance: the local computational complexity, the total number of messages sent, the maximum number of messages sent over any one link, and the maximum message size. For an algorithm to be practical, all four of these burdens must be modest. Because the study of cost-sharing algorithms (and of algorithmic mechanism design generally) is quite new, we will not give a precise definition of "modest"; we believe that more examples of *prima facie* good algorithms than we have at this early stage should inform such a definition. It

may turn out that polynomial-time local computation,[9] $O(n)$ messages total, $O(1)$ messages over any one link, and maximum message size that is polylogarithmic in n and p and polynomial in the size of the largest $c(l)$ or u_i is the right definition of "feasible network complexity." This is achieved by the algorithm in Section 3 below. On the other hand, there may be mechanism-design problems in which the best that can be achieved is $n \cdot polylog(n)$ messages total, and this may be acceptable in context. Our main goal in this paper is to point out that all four aspects of network complexity are important and that, in our particular problem, there is an algorithm in which all are acceptable.

We do not address any details of message transmission in this model. We assume that messages arrive reliably, within a small bounded time, and arrive in order. It is an open question whether one would obtain fundamentally different network complexity results for the mechanisms considered here if one assumed an unreliable network.

2.4. Related Work

This work lies in the intersection of game theory, theoretical computer science, and networking. Questions of incentives have long been central to game theory, and there is a vast literature on the *mechanism design* or *implementation* paradigm in which resource-allocation mechanisms are designed to achieve the socially desirable outcomes in spite of user selfishness. (See [12] for a review and [26] for an introduction.) Many of these approaches use Nash equilibria (or other notions of noncooperative behavior) rather than strategyproofness: That is, they assume that simultaneous selfish play leads to a self-consistent equilibrium, called a Nash equilibrium, in which no agent can improve her lot by deviating. The *Nash implementation* approach [3, 19] involves designing resource allocation mechanisms with Nash equilibria that yield the socially desirable outcome (such as an efficient and/or budget-balanced and/or fair allocation). In contrast, strategyproofness ensures that no matter how other agents behave – whether selfish, spiteful, or stupid – truthful revelation is the optimal (dominant) strategy for each user. Strategyproofness is much more exacting than Nash implementation, and so Nash implementation can achieve a much wider variety of outcomes. However, [7] argues that strategyproofness may be the only viable approach in the Internet, because one cannot ensure that simultaneous selfish play will reach the traditionally defined notions of equilibrium.

There is a long history of applying game-theoretic techniques to networking problems. Some use incentives as a metaphor for distributed protocol design (*e.g.*, [17]); the elements in the protocol are not selfish, but microeconomic principles are used to achieve coordination between the elements. Others analyze the impact of selfish users on current designs (*e.g.*, [14]), determining the resource allocations achieved

[9] More precisely, each round of local computation at a node α should take time polynomial in the sizes of its local inputs; these inputs may be supplied by users resident at α, may be received as messages from neighbors of α in $T(P)$, or may have been computed and stored in previous rounds. The total time complexity of the computation done at α will obviously also depend on the total number of rounds of computation done by α during the execution of the whole algorithm; ideally, this number will be $O(1)$, with each round of computation triggered by a round of incoming messages.

at Nash equilibrium. Still others adopt the mechanism-design approach and modify network designs to cope with user selfishness (*e.g.*, [6, 16, 30, 32, 33]). There is also significant use of mechanism design in distributed artificial intelligence (*e.g.*, [29, 34]) and in market-based computation (*e.g.*, [37])

Despite their increasing role in some of the more applied areas in computer science, incentives have rarely been an important consideration in traditional theoretical computer science. Typically, users are assumed either to be *cooperative* (*i.e.*, to follow the prescribed algorithm) or to be *adversaries* who attempt to harm other users. In contrast, the selfish users of game theory are neither cooperative nor spiteful; instead they attempt to optimize their own individual welfare. While one cannot assume selfish users will cooperate, one can assume that they will respond to incentives. Thus, one need not cope with the worst-case behavior of a byzantive adversary – only with predictably selfish behavior.

In short, the game-theoretic literature stresses incentives with (usually) little regard for network complexity, while the theoretical computer science literature (usually) focuses on complexity without much consideration of incentives. In one of the first works to merge these two concerns, Nisan and Ronen [23, 24, 25] pose noncooperative allocation problems such as routing and load balancing and analyze the computational complexity of the relevant strategyproof mechanisms. Our work is very much in the same spirit as theirs, except that they consider a centralized setting for the computation, and we address the network complexity in a distributed setting, for the concrete application of multicast cost sharing. The mechanisms we consider would not be interesting in Nisan and Ronen's model. Because their model allows all of the users to send their data to one node, the receiving node could just compute SH or MC cost shares, both of which are polynomial-time computable. SH and MC computations are interesting only when the computational model is distributed and network complexity matters.

3. MARGINAL COST MECHANISM

THEOREM 3.1. *MC cost sharing requires exactly two messages per link. There is an algorithm that computes the cost shares by performing one bottom-up traversal of $T(P)$, followed by one top-down traversal, and this algorithm is optimal with respect to number of messages sent.*

Proof. In order to describe the algorithm, we need the following notation. Let u^α denote the sum of the utilities of the users located at node α, c^α the cost of the link from α to its parent $p(\alpha)$ in the tree $T(P)$, $Ch(\alpha)$ all the child nodes of α in the tree $T(P)$, $V(P)$ all nodes in the tree $T(P)$, $res(\alpha)$ the set of users at node α, and $T^\alpha(P)$ the union of the subtree rooted at α and the link from α to $p(\alpha)$. With this, we can compute $W^\alpha(u)$, which is the welfare (*i.e.*, utilities minus cost) of $T^\alpha(P)$, as follows:

$$W^\alpha(u) = u^\alpha + \left(\sum_{\beta \in Ch(\alpha) | W^\beta(u) \geq 0} W^\beta(u) \right) - c^\alpha.$$

Note that these values can be computed by the bottom-up traversal given in Figure 1 below.[10] Naturally, $\sigma(i) = 1$ (that is, user i is included in the multicast) if $W^\alpha(u) \geq 0$ for all nodes α in the path from user i to the root.

Once the $W^\alpha(u)$'s have been computed, the values of the $\sigma_i(u)$'s – that is, the bits that indicate whether a user i is a member of the efficient set $R^*(u)$ – can be propagated in a top-down traversal.

The cost share $x_i(u)$ for user $i \in R^*(u)$, given by Equation 1, does not require a from-scratch recomputation of $W(u|^i 0)$. For each node α and user $i \in R^*(u)$ at α, let $y_i(u)$ be the smallest $W^\beta(u)$ of any node β in the path from α to the root. (This minimum welfare value might occur at α.) Then there are two cases:

- If $u_i \leq y_i(u)$, then, without user i, the efficient set is the same as it is with user i. That is, $R^*(u) = R^*(u|^i 0)$, and the difference $W(u) - W(u|^i 0)$ is u_i. Therefore, user i must pay $x_i(u) \equiv u_i - (W(u) - W(u|^i 0)) = 0$.
- If, however, $u_i > y_i(u)$, then dropping user i results in the elimination from $R^*(u)$ of a subtree of total welfare $y_i(u)$, and thus user i must pay exactly $x_i(u) = u_i - y_i(u)$.

To see this, note that dropping user i decreases $W^\alpha(u)$ by u_i. If $u_i > y_i(u)$, then there is some lowest (furthest from the source) ancestor α' of α for which dropping user i causes $W^{\alpha'}(u)$ to become negative and $T^{\alpha'}(P)$ to be dropped from $R^*(u)$. Dropping $T^{\alpha'}(P)$ may in turn result in a negative welfare value for some ancestor α'' of α', which would cause $T^{\alpha''}(P)$ to be dropped, and so forth. This "chain reaction" stops after the removal of a minimum-welfare subtree $T^\beta(P)$. On the other hand, if $u_i < y_i(u)$, then the tree structure of $R^*(u|^i 0)$ is the same as that of $R^*(u)$.

Observe that this propagation of $y_i(u)$ can be combined with the propagation of $\sigma_i(u)$, as shown in Figure 2 below.

In the top-down traversal given in Figure 2, we assume that each node α has the "state" from the (earlier) execution of the bottom-up traversal; this consists of the messages that it received from its children, the message that it sent to its parent, and the values σ_i for users i at α (some of which were erroneously, but temporarily, set to 1 and will be corrected in the top-down traversal). The top-down traversal has to convey enough information to allow nodes to compute cost shares using Equation 1 and to correct erroneous σ_i values.

Finally, note that two messages must in fact travel over each link in $T(P)$ if cost shares are to be computed correctly. There are instances in which, for all $\alpha \in V(P)$, cost shares at α depend on utilities at every descendant of α and on utilities and/or link costs between α and the root α_s of $T(P)$. In our model, α can only compute such a share after receiving *some* information from its parent and each of its children (although perhaps not as many bits of information as our algorithm sends). Examples of such instances include those in which $R^*(u) = P$ but setting $u_i = 0$ for any i would cause all users in a subtree rooted at some $\beta \in Ch(\alpha_s)$ not to receive the transmission (because each link joining α_s to one of its children has a very high cost). ■

[10] Johnson, Minkoff, and Phillips [15] use essentially the same formula in their (independently discovered) strong-pruning algorithm for the prize-collecting Steiner-tree problem.

FIG. 1. Bottom-Up Traversal: Computing Welfare Values

At node $\alpha \in V(P)$
  After receiving a message A^β from each child $\beta \in Ch(\alpha)$
    $W^\alpha \leftarrow u^\alpha + (\sum_{\beta \in Ch(\alpha)} A^\beta) - c^\alpha$
    If $W^\alpha \geq 0$ then
    {
      $\sigma_i \leftarrow 1$ for all $i \in res(\alpha)$
      Send W^α to parent $p(\alpha)$
    }
    Else
    {
      $\sigma_i \leftarrow 0$ for all $i \in res(\alpha)$
      Send 0 to parent $p(\alpha)$
    }

FIG. 2. Top-Down Traversal: Computing Membership Bits and Cost Shares

Initialize: Root α_s sends W^{α_s} to each of its children.
For each $\alpha \in V(P) - \{\alpha_s\}$
  After receiving message A from parent $p(\alpha)$
    //Case 1: $T^\alpha(P) \cap T(R^*(u)) = \emptyset$.
    //Set σ_i's properly at α and propagate non-membership downward.
    If $\sigma_i = 0$, for all $i \in res(\alpha)$, or $A < 0$ then
    {
      $x_i \leftarrow 0$ and $\sigma_i \leftarrow 0$ for all $i \in res(\alpha)$
      send -1 to β for all $\beta \in Ch(\alpha)$
    }
    //Case 2: $T^\alpha(P) \cap T(R^*(u)) \neq \emptyset$.
    //Compute cost shares and propagate minimum welfare value downward.
    Else
    {
      $A \leftarrow \min(A, W^\alpha)$
      For each $i \in res(\alpha)$
        If $u_i \leq A$, then $x_i \leftarrow 0$, else $x_i \leftarrow u_i - A$
      For each $\beta \in Ch(\alpha)$
        Send A to β
    }

4. WELFARE-MAXIMIZING MULTICAST TREE

The form of routing used in this paper requires that the actual delivery tree be a subtree of the given *universal* tree that connects the source to all users. While this accurately models the current design philosophy of multicast routing, which empha-

sizes simplicity and stability over optimization, there are other contexts involving tree-like delivery structures in which optimality would become the dominant consideration. One such example is that of installing communication lines connecting several institutions;[11] the situation is static once installed, and thus stability is not an issue, but minimizing cost can be quite important.

In such settings, it is natural to consider the following generalization of the MC mechanism. The node-set N and link-set L form a general digraph; the source α_s is an element of N. *Any* subtree T of (N, L) that is rooted at α_s can be a delivery tree; because, in this extension, we are identifying nodes with "players," such a T defines a receiver set $V(T)$ and a total welfare value that is just the sum of the utilities of nodes in T minus the sum of the costs of links in T. The question is whether, without the restriction used in Section 3 that the delivery tree be a subtree of the universal $T(P)$, it is computationally feasible to find a delivery tree of maximum total welfare.

More formally, let $\mathcal{T}(R)$ denote the set of all trees connecting the source to a receiver set R. Given a receiver set R, we choose the tree $T \in \mathcal{T}(R)$ that minimizes $c(T)$; define $C(R) = Min_{T \in \mathcal{T}(R)} c(T)$. Then, an efficient set $R^* \subseteq N$ is one that maximizes $[u_R - C(R)]$. There is no longer a unique largest efficient set, because the cost function $C(R)$ is not submodular (unlike $c(T(R))$ in our original problem, which is submodular). Let $W(N, L, c, u, r) = Max_{R \subseteq N - \{r\}}[u_R - C(R)]$ be the optimal efficiency. For a given selection of efficient sets $R^*(u)$, Equation 1 defines the cost shares for an MC-like strategyproof mechanism satisfying NPT, VP, and CS. The question is whether we can find a feasible algorithm to implement this mechanism.

Because computing $W()$ is equivalent to the net-worth maximization version of the *Prize-Collecting Steiner-Tree* problem [15], it is NP-hard. We now show that $W()$ is NP-hard to approximate within any constant factor, even if (N, L) is a bounded-degree graph.[12] Therefore, the natural generalization of the MC mechanism that we have defined in this section is apparently computationally infeasible.

THEOREM 4.1. *Let $G = (N, L)$ be a graph, $r \in N$ a distinguished root node, $c : L \to \Re^{\geq 0}$ a cost function on links, and $u : N \to \Re^{\geq 0}$ a utility function on nodes. For $R \subseteq N - \{r\}$, let $C(R)$ be the cost of a minimum-cost tree, the node-set of which contains r and R. Then, for any constant ϵ, $0 < \epsilon < 1$, the function*

$$W(N, L, c, u, r) = \max_{R \subseteq N - \{r\}} (u_R - C(R))$$

is NP-hard to approximate[13] *within ratio ϵ. Moreover, this hardness result holds even if (N, L) is a bounded-degree graph.*

[11] In this context, there is no notion of a "player" that is resident at a node: Each institution corresponds to a single node, and it has a single utility (or "prize") value; either this node will be connected to the delivery tree by the routing infrastructure, or it won't be.

[12] This easy result had been absent from the extensive literature on this problem.

[13] Recall that algorithm A "approximates f within ratio ϵ, $0 < \epsilon < 1$," if, for all x, $\epsilon \leq A(x)/f(x) \leq 1/\epsilon$.

Proof. Let $(x_1, \ldots, x_n, C_1, \ldots, C_m)$ be a SAT instance. Assume without loss of generality that the formula contains a clause $(x_i \vee \bar{x_i})$, for all $1 \leq i \leq n$.

The equivalent graph has four levels:

- The first level contains the root r, with utility ϵ^3.
- The second level contains a node r', with utility zero, connected to r via a link of cost $m \cdot K - n - 1$, where $K \gg m$.
- The third level contains $2n$ nodes, one for each x_i and $\bar{x_i}$, all with utility zero, and all connected to r' via unit-cost links.
- Finally, the fourth level contains m nodes, one for each clause, each of utility K, and each connected via zero-cost links to the level-3 nodes corresponding to the literals it contains.

Any graph of this form has a trivial solution of welfare ϵ^3, containing only the root node r.

To get a solution of welfare greater than ϵ^3, we must include the link (r, r'), which in turn forces the inclusion of all of the level-4 nodes. At most n links from r' to level-3 nodes can be included, because $n + 1$ such links would put us back to welfare ϵ^3. For each i, one of the x_i-node or the $\bar{x_i}$-node must be included, in order to reach the level-4 node corresponding to clause $(x_i \vee \bar{x_i})$. Therefore, a solution of welfare greater than ϵ^3 corresponds to a satisfying assignment; in fact, such a solution will have welfare exactly $1 + \epsilon^3$.

Unsatisfiable formulas correspond to graphs of maximum welfare ϵ^3, on which an ϵ-approximation algorithm for $W()$ would return a value less than or equal to ϵ^2. Satisfiable formulas correspond to graphs of maximum welfare $1+\epsilon^3$, on which an ϵ-approximation algorithm would return a value greater than or equal to $\epsilon(1+\epsilon^3) > \epsilon$. Thus ϵ-approximating $W()$ is NP-hard.

To obtain a hardness result for bounded-degree graphs, we modify the reduction as follows. First, replace the $2n$ links that connect r' to the level-3 "literal nodes" with a binary tree whose root is r' and whose leaves are the $2n$ literal nodes. All of the nodes in this binary tree have utility zero, and all of the links have cost zero, except for the links that connect the literal nodes to their parents; each of these $2n$ links has cost one. Next, assume without loss of generality that each clause C_j either is the disjunction of three literals or is of the form $(x_i \vee \bar{x_i})$. For each literal y, use a binary tree to connect the literal node corresponding to y to all of the nodes corresponding to clauses in which y appears. All of the links in these trees have cost zero, and all of the nodes have utility zero, except for the "clause nodes" themselves, each of which has utility K, as it did in the original reduction. ■

5. SHAPLEY VALUE MECHANISM

SH cost shares can be computed by a brute-force algorithm involving, in the worst case, the exchange of $\Theta(n \cdot p)$ messages total and at least p messages over certain links. We conjecture that this complexity is the best possible and present some evidence for this below. We first, however, explain the brute-force algorithm.

The simplest case of the SH cost-share problem is the one in which all u_i are sufficiently large to guarantee that all of P receives the transmission. (For example, $u_i > c(T(P))$, for all i, would suffice.) For this case the SH cost-shares can be

computed as follows.[14] Do a bottom-up traversal of the tree that determines, for each node α, the number p_α of users in the subtree rooted at α. Then, do a top-down traversal, which the root initiates by sending the number $md = 0$ to its children. After receiving message md, node α computes $md' \equiv \left(\frac{c(l)}{p_\alpha}\right) + md$, where l is the network link between α and its parent, assigns the cost share md' to each of its resident users, and sends md' to each child. Thus, each user ends up paying a fraction of the cost of each link in its path from the source, where the fraction is determined by the number of users sharing this link.

To apply the brute-force algorithm to the general case, we must proceed in an iterative fashion. We initially start, as before, with $R = P$ and compute the cost shares as above. However, in the general case, we cannot assume that $u_i \geq md'$ for all i, and so some users may prefer not to receive the transmission. So, after an iteration, we update R by omitting all users i such that $u_i < md'$, and repeat. The algorithm terminates when no more users need to be dropped. In the worst case, one user will be dropped after each iteration, resulting in a total of p iterations and $\Omega(n \cdot p)$ messages.

THEOREM 5.1. *The brute-force algorithm implements the SH mechanism with* $\Theta(n \cdot p)$ *message exchanges.*

We next show a result strongly suggesting that the brute-force algorithm is essentially optimal, *i.e.*, that such quadratic dependence is inherent.

Lower Bound

We prove our lower bound for a particular class of distributed algorithms. These algorithms start with real variables stored locally at the nodes and proceed by exchanging messages between nodes, interleaved with local computations. Each message is a real number, a *linear combination of the variables at the sending site and the messages received at the sending site at previous steps.*[15] If the coefficients involved in these linear combinations are constant and fixed in advance, then the algorithm is called an *oblivious linear distributed algorithm.*

A more general class of algorithms is the one in which the computation of the next message to be sent from a node is a decision tree with comparisons of linear combinations of the local variables and the messages received; the leaves of the tree then compute linear combinations of the local variables and the previously received messages. We call these the *non-oblivious linear distributed algorithms*, or simply *linear distributed algorithms*, a class that includes all algorithms described in this paper. In such algorithms, the space of all possible variable values of the initial variables is subdivided into convex polytopes, and particular values for the constant coefficients of the linear combinations prevail at each polytope.

[14] This simple case is essentially a distributed version of the linear-time algorithm given in [20].

[15] Synchronization is, of course, crucial in such algorithms. However, as will soon become apparent, we shall be interested in distributed algorithms over *two* sites, in which message exchange can be assumed to proceed in an orderly alternating fashion. Notice also that we assume that each message is a real number with no bounds on size, precision, or other computational attributes, in the spirit of algebraic complexity —a framework that excludes encoding tricks but includes all algorithms presented in this paper and enables our algebraic lower-bounding argument.

We can show the following result, suggesting that, up to constant factors, the brute-force algorithm is optimal with respect to the number of messages sent:

THEOREM 5.2. *There is an infinite family of multicast computations, with n nodes and $O(n)$ users, such that any linear distributed algorithm that implements the SH mechanism requires in the worst case $\Omega(n^2)$ message exchanges.*

Proof. Consider a tree that is a path with nodes α_0 through α_n. The source is α_0, the cost of the link $[\alpha_i, \alpha_{i+1}]$ is c_{i+1}, each of the nodes $\alpha_i, i = 1, \ldots, n-1$ has an agent with utility u_i, while there are n agents at α_n, with utilities $u_{nj}, j = 1, \ldots, n$. For reasons of purely notational simplicity, we assume here that the cost c_i is known only by node α_{i-1} (and not by node α_i); this assumption increases the number of messages that need to be exchanged across each link by at most one. We shall show that $\Omega(n^2)$ messages are necessary. In particular, we shall show that i messages must be exchanged across link $[\alpha_i, \alpha_{i+1}], i = 0, \ldots, n-1$.

Consider the last link $[\alpha_{n-1}, \alpha_n]$. (The general case is very similar and is explained below.) Suppose that the utilities u_i are very large, while all n agents at α_n are excluded one by one at each iteration of the brute-force algorithm. Consider the task of actually *checking* that the link costs and the utilities u_{nj} fall into this case. When the u_{nj} are in decreasing order, this is tantamount to checking the inequalities

$$u_{nj} < \sum_{i=1}^{n} \frac{c_i}{n+j-i} \tag{2}$$

Because we are only interested in the messages across link $[\alpha_{n-1}, \alpha_n]$, we can regard the network as two nodes, α_n and another node α that is formed by merging all the rest. The variables $u_{nj}, j = 1, \ldots, n$ are local to node α_n, while the variables $c_i, i = 1, \ldots, n$ are known to the node α. The assertion of the Theorem follows from the following general result about distributed computation:

LEMMA 5.1. *Any linear distributed algorithm by two nodes for checking the inequalities $Ax + By > a$ (assumed to have a feasible solution), where x is an n-vector known to node I, y is an n-vector known to node II, A, B are $n \times n$ nonsingular matrices known to both, and a is an n-vector known to both, requires n message exchanges.*

Proof of the Lemma: Because we are assuming that linear operations are free and that the matrices A and B are nonsingular, it is not a loss of generality to assume that the inequalities to be checked are $x < y$. It is quite obvious that testing these inequalities requires n messages; the careful proof follows.

Consider an oblivious distributed algorithm checking this with fewer than n messages. We shall identify a polytopal subdivision P of $\Re^{2n}$ as follows: Initially, $P = \Re^{2n}$, and we also set $P_0 = \{(x, y) \in \Re^{2n} : x = y\}$. Whenever a decision of the two decision trees splits P, we select from among the two pieces of P the one that (a) has a nonempty intersection with $\{(x, y) \in \Re^{2n} : x < y\}$ and (b) among those, the one that contains the highest-dimensional piece of P_0. Then the new P_0

is defined as the intersection of the old P_0 and the new P. Let P be the polytopal subdivision at the end of the algorithm.

We call a step in which the dimension of P_0 decreases a *separation;* at a separation, the dimension of P_0 decreases by 1, because an inequality $x_i < y_i$ was checked, for some i. If the separation (that is, the test of the decision tree that caused it) happened at node I, we call it a *I-separation*, otherwise a *II-separation.* Let s_{I} and s_{II} denote the total number of I-separations and II-separations, respectively.

Case 1: Suppose that $s_{\mathrm{I}} + s_{\mathrm{II}} < n$. In this case, P contains a nontrivial P_0, and thus a point with $x_i = y_i$, for some i, and hence P is a polytope that contains a point that satisfies $x < y$ and one that does not. Therefore, it witnesses that the algorithm is wrong.

Case 2: $s_{\mathrm{I}} + s_{\mathrm{II}} \geq n$. Let m_{I} be the number of messages received by I in this branch of the algorithm and m_{II} those received by II. Recall that $m_{\mathrm{I}} + m_{\mathrm{II}} < n$. It follows that either $s_{\mathrm{I}} > m_{\mathrm{I}}$ or $s_{\mathrm{II}} > m_{\mathrm{II}}$; without loss of generality, assume the former. Consider the s_{I} steps at which I suffers its separations; at these steps, the linear tests of I's decision tree were $\hat{x} - \hat{y}$, where $\hat{x}$ and $\hat{y}$ are x and y restricted to s_{I} indices. Now, these linear forms are linear combinations of x, plus the m_{I} messages received by I, and these latter messages are linear combinations of x and y, say $Cx + Dy$ for some $m_{\mathrm{I}} \times n$ matrices C and D. Thus, $\hat{x} - \hat{y}$ can be written as $\hat{x} - \hat{y} = E(Cx + Dy) + Fx$ for some $s_{\mathrm{I}} \times m_{\mathrm{I}}$ matrix E and $s_{\mathrm{I}} \times n$ matrix F. It follows from this equation that $-\hat{y} = EDy$, which is impossible, because matrix ED has rank $m_{\mathrm{I}} < s_{\mathrm{I}}$. This contradiction completes the proof. □

■

We would like to prove two extensions of this lower bound: The first deals with nonlinear algebraic operations and the second with the case in which the costs c_i are known in advance. The first extension should be provable by exploiting some algebraic-geometric understanding of the local structure of the high-dimensional variety that would now be the analogue of $Cx + Dy = b$. The second seems to be true but may require more sophisticated lower-bound techniques.

6. SUMMARY AND OPEN PROBLEMS

In this paper, we considered the problem of sharing the cost of a multicast transmission. We assumed that multicast transmissions incurred per-link network costs and that the strategyproof cost-sharing mechanisms had to satisfy the NPT, CS, and VP conditions described in Section 2.2. Based on the characterization in [22] of such mechanisms that are either budget-balanced or efficient, we focused on two cost-sharing mechanisms: Marginal Cost and Shapley Value. The treatment in [22] focused solely on the game-theoretic aspects of these two mechanisms, and our goal here was to determine whether they could be feasibly implemented on a network. We were thus led to analyze their network complexity, and the apparent gap between the network complexity of MC and that of SH was the topic of our main results.

The MC mechanism is implementable with an algorithm that only requires a single message sent in each direction on each link in the tree $T(P)$. Note that MC has the following property, which we call "nonseparability": The cost shares of an agent can depend on those of every other agent with which it shares a link; that is,

for every i, j with $T(i) \cap T(j) \neq \emptyset$, there are a set $\{c(l)\}_{l \in L}$ of link costs and utility vectors u and v, $u_k = v_k$ for all $k \neq j$, such that $x_i(u) \neq x_i(v)$ or $\sigma_i(u) \neq \sigma_i(v)$. For example, the instances described at the end of the proof of Theorem 3.1 illustrate the nonseparability of MC. Implementing a nonseparable mechanism requires that at least one message traverse each link in $T(P)$. Our results thus imply that, up to constant factors, the Marginal Cost mechanism, which requires exactly two messages to traverse each link (one in each direction), is as easy to implement as any other nonseparable mechanism. Informally, we use the term "minimal" to describe mechanisms that can be implemented with $O(1)$ messages traversing each link.

By contrast, there is an infinite family of cases in which the SH mechanism requires (in the limited computational model we considered) a linear number of messages on a linear number of links. Note that this is roughly the same amount of communication used by the *centralized* approach of sending all the u_i and c_j values to a designated node, computing the resulting cost shares at that designated node, and then sending the x_i and σ_i values back to each node. This centralized approach can be applied to all polynomial-time cost-sharing mechanisms; thus, at least in the limited computational model we considered, no cost-sharing mechanisms impose qualitatively higher communication burdens than SH. Informally, we use the term "maximal" to describe mechanisms that require a linear number of messages to traverse a linear number of links.

Therefore, the two mechanisms we considered are at the opposite ends of the feasibility spectrum. Although the purely game-theoretic analysis in [22] gave little guidance as to which might be more appropriate in practice, the network-complexity analysis here leaves little doubt that only the MC cost-sharing mechanism is feasible for large user populations. This example suggests that network complexity may be an important factor in evaluating candidate cost-sharing mechanisms in distributed systems.

Our study leaves many open questions. We divide them into four categories.

First, we know very little about minimal and maximal cost-sharing mechanisms in general. Natural questions include:

• Are all nonseparable budget-balanced mechanisms (satisfying group strategyproofness, CS, VP, and NPT) maximal? "Nonseparability" as we have defined it here is not necessarily the only requirement to consider. One idea we are trying to capture is that there may be budget-balanced mechanisms that achieve low network complexity trivially by computing outputs $\sigma_i(u)$ and $x_i(u)$ for user i that simply don't depend on most of the inputs $c(l)$ and u_j, $j \neq i$. Some form of "nontriviality" requirement may thus be needed to make the question interesting; "nonseparability" is one form that this requirement can take but not necessarily the only one. In general, the question is whether the budget-balance requirement leads naturally, with a few additional restrictions, to the infeasibility of the mechanism.

• Are there any efficient and strategyproof mechanisms besides Marginal Cost that are minimal? Recall that MC was essentially the only efficient strategyproof mechanism satisfying VP and NPT; thus, this question asks whether relaxing the VP and/or NPT requirements would lead to mechanisms that were less easily implementable than MC.

• What are the game-theoretic properties of strategyproof, minimal mechanisms? Are there game-theoretic properties that are especially compatible with ease of implementation?

Second, the current model is, in many ways, unrealistic; we should move beyond the current model to see which factors significantly change the nature of the results. For example, it is doubtful that network providers would charge per-link for transmissions. While it is not clear what a more realistic charging model would be, it would be of interest to know how the results here would change as the charging model changed. In addition, the NPT requirement – that users are not paid for participating – does not apply to all real-world businesses, particularly e-businesses; as noted above, we would like to determine the network-complexity properties of cost-sharing mechanisms that do not have the NPT requirement.

In a similar vein, we can broaden the class of mechanisms considered by addressing different goals. Our treatment, following [22], only considered budget-balance and efficiency as the two natural goals (in addition to our basic requirements of strategyproofness, CS, VP, and NPT). However, profit maximization is also an important goal for economic mechanisms. While budget-balance and efficiency are compatible with strategyproofness, the naive definition of maximal profit is not; all profit-maximizing mechanisms are equivalent to the one that charges each agent in the efficient coalition u_i and excludes all agents not in the efficient coalition; this is clearly not strategyproof. The open question is how to define a standard for profit maximization that is compatible with strategyproofness. Goldberg *et al.* [8] address this issue quite elegantly for the case in which there are no costs. Their approach does not appear to generalize easily to the case in which users incur different network costs.

Third, one can try to develop a general theory of the network complexity of mechanism-design problems, as Nisan and Ronen [23, 24, 25] have done for the centralized complexity of these problems. It would be useful to be able to prove that particular mechanisms are "complete" or "hard" for the relevant network complexity classes. This will require formal computational models, appropriate notions of "reduction," and other basic ingredients of complexity theory. For all of the Nash implementation and strategyproof mechanisms in the literature, one can ask whether they would be feasibly implementable if the users were on a network. In this paper, the network setting is intrinsic to the statement of the problem, but there will doubtless be situations in which agents involved in some other mechanism interact via the network rather than being centrally located. The question is whether network complexity is likely to be an important issue for these mechanisms, and that will depend greatly on the particular setting. One problem domain in which this might be important is *distributed auctions*; what class of strategyproof auctions have low network complexity?

Fourth, and last, we can leave the game-theoretic considerations behind and focus solely on the network complexity of various distributed functions. Viewing the u_i as general inputs and the x_i as general outputs, one can ask which functions $x_i(u)$ can be feasibly computed on a network. In particular, which other functions $x_i(u)$ are minimal, which are maximal, and which are in between?

ACKNOWLEDGMENT

We would like to thank our colleague Li Li for pointing out an error in an earlier version of this paper.

REFERENCES

1. Ballardie, A. (1993). Core Based Trees (CBT), *in* "Proceedings of Sigcomm '93," pp. 85–95, ACM Press, New York.
2. Clarke, E. H. (1971). Multipart pricing of public goods, *Public Choice* **11**, 17–33.
3. Dasgupta, P., P. Hammond, and E. Maskin (1979). The implementation of social choice rules, *Review of Economic Studies* **46**, 185–216.
4. Deering, S., and D. Cheriton (1990). Multicast routing in datagram internetworks and extended LANs, *ACM Transactions on Computer Systems* **8**, 85–110.
5. Deering, S., D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei (1996). The PIM architecture for wide-area multicast routing, *ACM/IEEE Transactions on Networking* **54**, 153–162.
6. Ferguson, D., C. Nikolaou, and Y. Yemini (1989). An economy for flow control in computer networks, *in* "Proceedings of the 8th Infocom," pp. 100–118, IEEE Computer Society Press, Los Alamitos.
7. Friedman, E., and S. Shenker (1997). "Learning and Implementation in the Internet," preprint. Available at `http://www.aciri.org/shenker/decent.ps`
8. Goldberg, A. V., J. D. Hartline, and A. Wright (1999). "Competitive Auctions and Digital Goods," InterTrust Technical Report 99-01.
Available at `http://www.intertrust.com/star/tr/tr-99-01.html`
9. Green, J., E. Kohlberg, and J. J. Laffont (1976). Partial equilibrium approach to the free rider problem, *Journal of Public Economics* **6**, 375–394.
10. Green, J. and J. J. Laffont (1979). Incentives in public decision making, *in* "Studies in Public Economics," pp. 65–78, North-Holland, Amsterdam, Vol. 1.
11. Groves, T. (1973). Incentives in teams, *Econometrica* **41**, 617–663.
12. Groves, T. and J. Ledyard (1987). Incentive compatibility since 1972, *in* "Information, Incentives, and Economics Mechanisms," pp. 48–111, University of Minnesota Press, Minneapolis.
13. Holbrook, H. and D. Cheriton (1999). IP multicast channels: EXPRESS support for large-scale single-source applications, *in* "Proceedings of Sigcomm '99," pp. 65–78, ACM Press, New York.
14. Hsiao, M.-T. and A. Lazar (1988). A game theoretic approach to decentralized flow control of markovian queueing networks, *in* "Performance '87," pp. 55–74, North-Holland, Amsterdam.
15. Johnson, D. S., M. Minkoff, and S. Phillips (2000). The prize collecting steiner tree problem: theory and practice, *in* "Proceedings of the 11th Symposium on Discrete Algorithms," pp. 760–769, ACM Press/SIAM, New York/Philadelphia.
16. Korilis, Y., A. A. Lazar, and A. Orda (1995). Architecting noncooperative networks, *Journal on Selected Areas in Communications* **13**, 1241–1251.
17. Kurose, J. F. and R. Simha (1989). A microeconomic approach to optimal resource allocation in distributed computer systems, *IEEE Transactions on Computers* **38**, 705–717.
18. Kushilevitz, E. and N. Nisan (1997). "Communication Complexity," Cambridge University Press, Cambridge UK.
19. Maskin, E. (1985). The theory of implementation in Nash equilibrium, *in* "Social Goals and Organization: Essays in Memory of Elisha Pazner," pp. 173–204, Cambridge University Press, Cambridge UK.
20. Megiddo, N. (1978). Computational complexity of the game theory approach to cost allocation for a tree, *Mathematics of Operations Research* **3**, 189–196.
21. Moulin, H. (1999). Incremental cost sharing; characterization by strategyproofness, *Social Choice and Welfare* **16**, 279–320.
22. Moulin, H. and S. Shenker (1997). Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency, to appear in *Economic Theory*. Available in preprint form at `http://www.aciri.org/shenker/cost.ps`

23. Nisan, N. (1999). Algorithms for selfish agents, *in* "Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science," pp. 1–17, Springer-Verlag, Berlin. Lecture Notes in Computer Science, Vol. 1563.

24. Nisan, N. and A. Ronen (1999). Algorithmic mechanism design, to appear in *Games and Economic Behavior*. Extended abstract *in* "Proceedings of the 31st Symposium on Theory of Computing," pp. 129–140, ACM Press, New York.

25. Nisan, N. and A. Ronen (2000). Computationally Feasible VCG Mechanisms, to be presented at "Games 2000." Available in preprint form at `http:/www.cs.huji.ac.il/~noam/vcgbased.ps`

26. Osborne, M.-J. and A. Rubinstein (1994). "A Course in Game Theory," MIT Press, Cambridge MA.

27. Perlman, R., C.-Y. Lee, A. Ballardie, J. Crowcroft., Z. Wang, T. Maufer, C. Diot, and M. Green (1999). "Simple multicast: A design for simple low-overhead multicast," IETF Internet Draft (Work in Progress).

28. Roberts, K. (1979). The Characterization of Implementable Choice Rules, *in* "Aggregation and Revelation of Preferences," pp. 321–348, North-Holland, Amsterdam.

29. Rosenschein, J. and G. Zlotkin (1994). "Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers," MIT Press, Cambridge MA.

30. Sanders, B. (1988). An incentive compatible flow control algorithm for rate allocation in computer networks, *IEEE Transactions on Computers* **37**, 1067–1072.

31. Shapley, L. S. (1953). A value for n-person games, *in* "Contributions to the Theory of Games," pp. 31–40, Princeton University Press, Princeton.

32. Shenker, S. (1990). Efficient network allocations with selfish users, *in* "Performance '90," pp. 279–285, North-Holland, Amsterdam.

33. Shenker, S. (1995). Making greed work in networks, *ACM/IEEE Transactions on Networking* **3**, 819–831.

34. Shoham, Y. and M. Wellman (1997). Economic principles of multi-agent systems, *Artificial Intelligence* **94**, 1–6.

35. Shubik, M. (1962). Incentives, decentralized controls, the assignment of joint costs and internal pricing, *Management Science* **8**, 325–343.

36. Vickrey, W. (1961). Counterspeculation, auctions, and competitive sealed tenders, *Journal of Finance* **16**, 8–37.

37. Walsh, W. and M. Wellman (1998). A market protocol for decentralized task allocation, *in* "Proceedings of the Third International Conference on Multi-Agent Systems," pp. 325–332, IEEE Computer Society Press, Los Alamitos.