Distributed Algorithmic Mechanism Design: Recent Results and Future Directions[*]

Joan Feigenbaum[†]
Yale University
Computer Science Department
New Haven, CT 06520 USA
feigenbaum@cs.yale.edu

Scott Shenker[‡]
ICSI
1947 Center Street
Berkeley, CA 94704 USA
shenker@icsi.berkeley.edu

ABSTRACT

Distributed Algorithmic Mechanism Design (DAMD) combines theoretical computer science's traditional focus on computational tractability with its more recent interest in incentive compatibility and distributed computing. The Internet's decentralized nature, in which distributed computation and autonomous agents prevail, makes DAMD a very natural approach for many Internet problems. This paper first outlines the basics of DAMD and then reviews previous DAMD results on multicast cost sharing and interdomain routing. The remainder of the paper describes several promising research directions and poses some specific open problems.

Categories and Subject Descriptors

F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems; J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Algorithms, Economics, Theory

Keywords

Algorithmic mechanism design, distributed computation, Internet algorithms, multicast, routing

1. INTRODUCTION

Multi-agent systems have been extensively studied in both economics and computer science, but the two communities have approached the topic very differently. In traditional theoretical computer science (TCS), computational agents are typically assumed either to be *obedient* (*i.e.*, to follow the prescribed algorithm) or to be *adversaries* who "play against" each other. On the other hand, the *strategic* agents in game theory are neither obedient nor adversarial. Although one cannot assume that they will follow the prescribed algorithm, one can assume that they will respond to incentives. Thus, the economics literature traditionally stressed incentives and downplayed computational complexity, and the TCS literature traditionally did the opposite. The emergence of the Internet as a standard platform for distributed computation has radically changed this state of affairs: Ownership, operation, and use by many self-interested, independent parties give the Internet the characteristics of an economy as well as those of a computer.

This development requires that these previously separable concerns – incentive compatibility and computational tractability – be jointly addressed. Although many subdisciplines of computer science have a long history of using game theory — such as networking (*e.g.*, [21, 25, 35]), distributed artificial intelligence (*e.g.*, [55, 61]), and market-based computation (*e.g.*, [65]) — the first work in TCS to address incentives and computational complexity simultaneously was Nisan and Ronen's seminal paper [52] on *algorithmic mechanism design* (AMD). This paper put forth a formal model of *centralized* computation that combined incentive compatibility (the "mechanism design" part) with computational tractability (the "algorithmic" part). Feigenbaum, Papadimitriou, and Shenker [20] extended this to *distributed algorithmic mechanism design* (DAMD), in which the same goals of incentive compatibility and computational tractability are present, but, in addition, the agents, the relevant information, and the computational model are all inherently distributed.

The Internet is an arena in which incentive compatibility, distributed computation, and computational complexity are all highly relevant. Thus, we believe that DAMD, with its simultaneous attention to these issues, will be important for understanding our Internet-centric future. This paper is intended to provide a basic overview of DAMD and to identify several promising areas for future research. We start, in Section 2, by providing some necessary background on *mechanism design* (MD), algorithmic and otherwise. In

[*]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.

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

[‡]Supported in part by NSF grants ITR-0081698, ITR-0121555, and ANI-9730162.

Dial-M'02, September 28, 2002, Atlanta, Georgia, USA.

Section 3, we review some previous DAMD results on multicast cost sharing and interdomain routing. The next five sections are devoted to exploring the core technical foundations of DAMD; we discuss the notion of hardness in DAMD problems (Section 4), the role of approximations (Section 5), aspects of strategic models (Section 6), the use of indirect mechanisms (Section 7), and alternative solution concepts (Section 8). We then end the paper in Section 9 with a discussion of several promising applications of DAMD. Throughout Sections 4-9, we pose numerous open problems, some very specific and others quite general, concerning the foundations and applications of DAMD. Additional material about the AMD and DAMD research agendas can be found in, *e.g.*, [50, 51, 54].

2. MD TO AMD TO DAMD

In essence, game theory is the study of what happens when independent agents act selfishly. Mechanism design asks how one can design systems so that agents' selfish behavior results in the desired system-wide goals. The "mechanisms" in this field are output specifications and payments to agents that incentivize them to behave in ways that lead to the desired system-wide result. For example, consider the problem of routing. Agents may be individual routers within a network or entire autonomous *domains*. Each agent incurs a cost when it transports a packet, and this cost is known only to the agent, not to the mechanism designer or to the routing protocol. Each agent is required by the protocol to declare a cost. The system-wide goal is to have the routing protocol choose the true lowest-cost path between any two agents in the network. The mechanism specifies, for each network topology, each sender-receiver pair, and each set of agents' declared costs, a path from sender to receiver and a payment to each agent; the mechanism designer's task is to find a formula for the payments that causes agents to be no worse off by revealing their true costs than they would be by lying about their costs. Such truthful revelation would allow the routing protocol to achieve the system-wide goal of having all the traffic follow lowest-cost paths.

More formally, consider a distributed system in which there is a set of possible outcomes $\mathcal{O}$. Each of the n autonomous strategic agents has a utility function $u_i : \mathcal{O} \to \Re$, where $u_i \in \mathcal{U}$, that expresses its preferences over these outcomes. The desired system-wide goals are specified by a *social choice function* (SCF) $F : \mathcal{U}^n \to \mathcal{O}$ that maps each particular instantiation of agents (who are completely described by their utility functions) into a particular outcome.[1] The problem is that these utilities are known only to the agents, not to the system designer or to any other central administrative entity; thus, one cannot just implement the desired outcome by fiat.

An SCF is *strategyproof* if $u_i(F(u)) \geq u_i(F(u|^i v))$, for all i and all $v \in \mathcal{U}$, where we use the notation $(u|^i v)_i = v$ and $(u|^i v)_j = u_j$, for all $j \neq i$. If F is strategyproof, then no agent has an incentive to lie, and the desired social goals can be achieved by asking agents to reveal their utility functions. Mechanisms in which agents are asked to directly reveal their utility functions are call *direct* mechanisms.

An SCF is *group-strategyproof* if the following holds for all S, u, and u' (where $S = \{i \mid u_i \neq u'_i\}$ is the defecting group): Either $u_i(F(u)) = u_i(F(u'))$, $\forall i \in S$, or $\exists i \in S$ for which $u_i(F(u')) < u_i(F(u))$. That is, if any agent in the group benefits from the group's colluding and lying to the mechanism, then at least one agent in the group suffers.

An important class of problems are those in which the utilities are *quasilinear*, and the outcome space $\mathcal{O}$ factors into a set of system states $\tilde{\mathcal{O}}$ and a set of payment states $\mathcal{P} \subseteq \Re^n$ that represent a vector of payoffs (or charges). At a particular outcome $o = (\tilde{o}, p)$, agent i's utility factors into $u_i(o) = v_i(\tilde{o}) + p_i$, where $v_i : \tilde{\mathcal{O}} \to \Re$ represents his valuations of each of the system states, and p_i is his payment. For such problems, there is a class of strategyproof mechanisms, called Vickrey-Clarke-Groves (VCG) mechanisms [13, 29, 64], that result in the system state that optimizes $\sum_i v_i(\tilde{o})$.

Direct strategyproof mechanisms provide a conceptually simple, if not always ideal (see Section 7), way to achieve strategyproof SCFs. However, there are many cases in which the desired result, *i.e.*, the desired social choice function F, is not strategyproof. To describe how to realize such non-strategyproof SCFs, we now introduce *indirect* mechanisms. Here, one designs a mechanism $< M, S >$, where S is a *strategy space*, and $M : S^n \to \mathcal{O}$ maps vectors of strategies into outcomes.[2] These are called indirect mechanisms, because the agents no longer directly reveal their utilities but instead choose strategies from the space S. This strategy choice is done selfishly, with each agent attempting to maximize its own utility. For a given mechanism M and a given utility vector u, we let the set $C_M(u) \subseteq S^n$ represent all possible strategy vectors that could reasonably result from selfish behavior. This set is called the *solution concept*. Traditional game theory often uses the *Nash-equilibrium* solution concept, *i.e.*, selfish play is assumed to result in strategy vectors in which no agent can unilaterally increase his utility. Other solution concepts include *rationalizable strategies* (agents use strategies that are best responses to rational beliefs about the other agents' strategy choices [10, 57]), *evolutionarily stable strategies* (agents imitate the successful strategies used by others in previous rounds of the game [62]), and *dominant strategies* (agents only choose strategies that, *regardless* of how other agents play, never result in lower payoffs than any other strategy). To date, most of the AMD and DAMD literature uses the dominant-strategy solution concept.

The goal of mechanism design is to define a mechanism M that implements the SCF, *i.e.*, $M(C_M(u)) = F(u)$, for all $u \in \mathcal{U}^n$.

When this condition holds, then selfish behavior by the agents will result in the desired system-wide outcome. In short, the system will be incentive-compatible. There is a large game-theory literature on which SCFs can be achieved for different notions of "incentive compatibility," *e.g.*, for different solution concepts; see Jackson [32] for an overview. With the Nash-equilibrium solution concept, one can design mechanisms to achieve a very wide range of non-strategyproof social choice functions [38].

When $M = F$ and $S = \mathcal{U}$, we reduce to the direct-

[1] More generally, we can consider *social choice correspondences* (SCCs), $H : \mathcal{U}^n \to 2^{\mathcal{O}}$, which map utility vectors into sets of outcomes. For notational simplicity, we discuss only SCFs in this section. In addition, we restrict ourselves to equivalent agents; in general, each agent could have a different set of possible utilities $\mathcal{U}_i$.

[2] Our assumption that all agents are equivalent, made for notational simplicity, renders all strategy spaces the same; in general, we could have different strategy spaces S_i.

mechanism case, and so our preceding discussion applies to direct mechanisms as well. That is, one can achieve non-strategyproof SCFs with direct mechanisms by invoking different solution concepts. For example, one can achieve efficiency and budget balance using the Bayesian-Nash-equilibrium solution concept [7, 14] – something that is impossible using the dominant-strategy solution concept [27, 58].

It is important to note that, although the mechanism is chosen by the system designer, the solution concept is supposed to reflect reality. The solution concept thus depends greatly on the context (*e.g.*, is it a repeated game or a single-shot game, do agents collude, do they know about the other agents, do they know about the other agents' strategic choices, *etc.*). Because the Internet is somewhat different from traditional game-theoretic contexts, the traditional solution concepts may not be sufficient; we shall return to this issue in Section 8.

The game-theory literature on mechanism design does not consider computational and communication complexity, and many of the existence proofs rely on extremely impractical mechanisms. For the mechanism-design approach to have any practical relevance for Internet computation, one must focus on scalable algorithms. That is, the function M must be computable with reasonable computational and communication resources.

Nisan and Ronen [52] initiated the study of AMD by adding computational tractability to the set of concerns that must be addressed in the design of incentive-compatible mechanisms. 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 $M(\cdot)$ is polynomial-time computable. They also provide strategyproof, polynomial-time VCG mechanisms for some concrete problems, including lowest-cost paths and task allocation.

The centralized computational model of [52] is not adequate for the study of Internet computation, where not only are the agents distributed, but so are the resources (*e.g.*, link bandwidth and cache storage) and the computational nodes. Internet-based mechanisms involve distributed algorithms and any measure of their computational feasibility must reflect their distributed nature. In one attempt to address this issue, Feigenbaum, Papadimitriou, and Shenker [20] put forth a general concept of network complexity that requires a distributed algorithm executed over an interconnection network T to be modest in four respects: the total number of messages sent over T (ideally, this should be linear in $|T|$), the maximum number of messages sent over any one link in T (ideally, this should be constant, to avoid "hot spots" altogether), the maximum size of a message, and the local computational burden on agents.

The network-complexity criterion in [20] evaluates the mechanism in isolation based on its *absolute* computation and communication requirements. A *relative* notion of complexity, which we call *protocol compatibility*, is adopted in the work of Feigenbaum, Papadimitriou, Sami, and Shenker [19] on interdomain-routing mechanism design. This measure of complexity does not place absolute limits on what is considered feasible; instead, it requires the mechanism to be a simple extension of a widely deployed, standard Internet protocol. The relevant standardized protocol in [19] is the Border Gateway Protocol (BGP). For a distributed algorithm that computes a mechanism to be considered a simple extension of a standard protocol, it must have the same general algorithmic structure as the standard and must not require substantially more computation, communication, local storage, or any other resource expenditure than the standard, regardless of whether the standard has high or low absolute network complexity. Protocol compatibility addresses two aspects of practical feasibility – computational tractability and deployability[3] – and we expect it to become an increasingly important aspect of DAMD in particular and Internet algorithms in general. In this paper, we use the term *network complexity* generically, encompassing the absolute notion of network complexity used in [20], the relative notion of protocol compatibility used in [19], and other related notions of complexity of Internet computation that will arise in the analysis of future distributed algorithmic mechanisms. Clearly, "network complexity" is not (yet) a well defined term, and we return to this point in Section 4 below. We expect the development of more *prima facie* good (and bad) distributed algorithmic mechanisms to lead to a satisfactory formalization.

3. SOME PREVIOUS DAMD RESULTS

The DAMD approach is relevant to several problems of practical importance. For instance, as we discuss in Section 9, problems of web caching, peer-to-peer systems, overlay networks, and task allocation involve distributed computing by many (possibly) selfish agents. In this section, we focus on two specific scenarios in which DAMD has been applied: multicast cost sharing, which exercises the notion of absolute network complexity, and interdomain routing, which exercises the notion of BGP compatibility.

3.1 Multicast Cost Sharing

The *multicast cost-sharing mechanism-design* problem involves an agent population P residing at a set of network nodes N that are connected by bidirectional network links L. The multicast flow emanates from a source node $\alpha_s \in N$; given any set of receivers $R \subseteq P$, the transmission flows through a *multicast tree* $T(R) \subseteq L$ rooted at α_s and spanning the nodes at which agents in R reside. It is assumed that there is a *universal* tree $T(P)$ and that, for each subset $R \subseteq P$, the multicast tree $T(R)$ is merely the smallest subtree of $T(P)$ required to reach the elements in R. Each link $l \in L$ has an associated cost $c(l) \geq 0$ that is known by the nodes on each end, and each agent i assigns a value v_i to receiving the transmission. A cost-sharing mechanism determines which agents receive the multicast transmission and how much each receiver is charged. We let $x_i \geq 0$ denote how much agent i is charged and σ_i denote whether agent i receives the transmission; $\sigma_i = 1$ if the agent receives the multicast transmission, and $\sigma_i = 0$ otherwise. We use v to denote the input vector $(v_1, v_2, \ldots, v_{|P|})$. The mechanism M is then a pair of functions $M(v) = (x(v), \sigma(v))$. The *receiver set* for a given input vector is $R(v) = \{i \mid \sigma_i = 1\}$. An agent's individual utility is therefore given by the quasilinear form $u_i = \sigma_i v_i - x_i$. The cost of the tree $T(R)$ reaching a set of receivers R is $c(T(R))$, and the overall welfare, also known as *efficiency* or *net worth*, is $NW(R) = v_R - c(T(R))$, where $v_R = \sum_{i \in R} v_i$ and $c(T(R)) = \sum_{l \in T(R)} c(l)$. The overall welfare measures the total benefit of providing the multi-

[3]In practice, straightforward extensions of existing protocols are easier to deploy than *de novo* designs.

cast transmission (the sum of the valuations minus the total transmission cost).

Economic considerations [47] point to two strategyproof mechanisms that are worthy of algorithmic consideration: marginal-cost (MC) and Shapley-value (SH). The MC mechanism is *efficient*, which means that it chooses the receiver set R that maximizes $NW(R)$; let W be the net worth of this welfare-maximizing R. For each $i \in R$, let W^{-i} be the net worth of the receiver set that the MC mechanism would have computed if i had not participated (*i.e.,* if v_i had been set to 0). Then $W - W^{-i}$ measures the gain in overall welfare that results from i's participation. The cost share that MC assigns to i is $x_i \equiv v_i - (W - W^{-i})$. MC is the only strategyproof and efficient mechanism that also has the following two properties:

NPT No Positive Transfers: $x_i(v) \geq 0$, or, in other words, the mechanism cannot *pay* receivers to receive the transmission.

VP Voluntary Participation: $u_i(v) \geq 0$; this implies that $x_i = 0$ whenever $\sigma_i = 0$ and that agents are always free to not receive the transmission and not be charged (by setting $v_i = 0$).

By contrast, the SH mechanism is group-strategyproof and *budget-balanced*, where the latter means simply that $\sum_{i \in R} x_i = c(T(R))$, where R is the receiver set chosen by the mechanism. SH assigns cost shares x_i by dividing the cost $c(l)$ of each link l in $T(R)$ equally among all members of $i \in R$ that are downstream of l. The SH receiver set is the largest $R \subseteq P$ such that $v_i \geq x_i$, for all $i \in R$. A classical result in mechanism design [27, 58] shows that no strategyproof mechanism can be both efficient and budget-balanced.

The MC mechanism has good network complexity: In [20], a distributed algorithm is given that computes the MC receiver set and cost shares by sending just two modest-sized messages over each $l \in L$ and doing two very simple calculations at each node. On the other hand, the SH mechanism has bad network complexity: In [18], it is shown that any algorithm, deterministic or randomized, that computes SH must, in the worst case, send $\Omega(|P|)$ bits over linearly many links.

Before turning to our next representative DAMD problem, *i.e.*, interdomain routing, we say a few words about why efficiency and budget balance are natural mechanism-design goals. Efficiency arises naturally as a design goal in the scenario in which the network is owned and operated by society at large, and multicast delivery may be subsidized, *e.g.*, via taxation, if the cost-sharing mechanism runs a deficit; here, the MC mechanism is a natural one to use, because it maximizes the overall welfare of the society as a whole and ensures (because it's strategyproof) that, once the collective choice has been made to charge for multicast delivery in this fashion, no single agent can cheat the group. Budget balance arises naturally as a design goal if the prices charged for multicast delivery must be set by competition among service providers. Competing providers could not charge more than their real costs, because they would be undercut, nor could they charge less than their real costs, because they would go out of business. These are the two scenarios considered in the work of Moulin and Shenker [47], which provides the economic foundation for [20] and most of the subsequent work on computational aspects of multicast cost sharing. By contrast, in the scenario in which the multicast delivery is done by a monopoly content owner, profit maximization is the natural mechanism-design goal. Fiat *et al.* [22] provide several novel cost-sharing mechanisms for this scenario.

Finally, we note that, in the problem as we have stated it here, the potential receivers are strategic, but the network (*i.e.*, the universal multicast tree $T(P)$) is obedient. In particular, the network nodes are neither in cahoots with nor conspiring against their resident agents, and the various subnetworks are not competing with each other or with the network as a whole. This is an accurate model of the real-world multicasting scenarios discussed above, in which $T(P)$ is operated by society at large, by a service provider with competitors, or by a monopoly content owner. Even in this simplest possible strategic model, determining the inherent network complexity of natural mechanisms is nontrivial. There may be other multicasting scenarios in which more complex strategic models are needed; we return to this issue in Section 6 below.

Although the multicast cost-sharing problem has been quite useful in establishing the basic conceptual foundations of DAMD, it is neither realistically formulated[4] nor of pressing importance. Interdomain routing, our next example, is both more realistic and more important.

3.2 Interdomain Routing

The Internet is comprised of many separate administrative domains or *Autonomous Systems* (ASs). Routing between these domains – *i.e.*, interdomain routing – is currently handled by the Border Gateway Protocol (BGP). There has been much research on routing in general and BGP in particular, but most of it takes a traditional protocol-design approach.[5] Recently, Feigenbaum, Papadimitriou, Sami, and Shenker [19] focused on DAMD issues inherent in interdomain routing.

The basic incentive problem involves transit traffic, *i.e.*, traffic neither originating from nor destined to the AS that is currently carrying the packets. For the overall efficiency of the network, packets should travel along shortest or, more generally, lowest-cost paths (LCPs). These optimal paths would typically, in general networks, cut across several ASs. However, carrying transit traffic is a burden that ASs would prefer to avoid. The basic problem is simple: Overall system efficiency is maximized when ASs accept transit traffic, but individual domains are happiest when they carry no transit traffic at all.

In the model of Feigenbaum *et al.* [19], which is an extension of an earlier (centralized) LCP-mechanism model proposed by Nisan and Ronen [52] and studied further by Hershberger and Suri [30], 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. Furthermore, the model also assumes that, to compensate for these incurred costs, each AS is paid a *price* for carrying transit traffic. The goal is to maximize network efficiency by routing packets along the LCPs. Standard routing pro-

[4]For example, the per-link cost model, although appealing and widely used, does not accurately capture network costs in most situations. Unfortunately, we are unaware of an alternative, well-validated cost model.

[5]By this we mean that the participating entities are assumed to be obedient, and so incentive issues can be ignored.

tocols (such as BGP) can compute LCPs given a set of AS costs.[6] However, under many pricing schemes, an AS would be better off lying about its costs;[7] such lying would cause traffic to take non-optimal routes and thereby interfere with overall network efficiency.

To prevent this, one needs the pricing scheme to be strategyproof, so that ASs have no incentive to lie about their costs. The pricing scheme should also have the reasonable property that ASs that carry no transit traffic at all receive no payment. It is shown in [19] that there is only one strategyproof pricing scheme with this property; it is a member of the VCG family. Moreover, a BGP-compatible distributed algorithm is given that computes these prices. This algorithm requires only minor and straightforward modifications of the BGP computational model given by Griffin and Wilfong [28]. Specifically, the algorithm in [19] requires a small constant-factor increase in both the table sizes and the message sizes of BGP, but it does not require any new messages or any new infrastructural or computational capability; in particular, all messages are still sent between neighbors in the AS graph.[8] Similarly, the local computation done by a node in each stage (*i.e.*, between receiving an updated table from a neighbor and, if necessary, sending an update to each of its neighbors) is the same order of magnitude as the BGP local-computation time.[9]

The results on multicast cost sharing and interdomain routing represent the two most successful applications to date of DAMD to practical network problems. In Sections 4-8, we turn our focus away from specific applications and towards the foundational underpinnings of DAMD. Each of these five sections fleshes out a fundamental aspect of DAMD: "Hard" vs. "easy" DAMD problems, the role (and meaning) of approximation, strategic models, indirect mechanisms, and solution concepts. In section 9, we return to specific Internet-based problems in which DAMD may be applicable. The general discussions to follow are augmented by a series of open problems representing both open-ended and focused research issues that warrant further study.

4. HARD AND EASY DAMD PROBLEMS

The central mission of TCS is to determine which problems are easy and which are hard in relevant computational models. In the Turing-machine model of centralized computation, the (crude) distinction is between polynomial-time solvable problems and those that are NP-hard. In the PRAM model of parallel computation, it is between those problems that are in NC and those that are P-hard. One of the major goals of this study of DAMD foundations is to develop the tools needed to classify relevant problems as easy or hard "to compute incentive-compatibly on the Internet" and to find more natural examples of both hard and easy DAMD problems.

Informally, a DAMD problem can be considered "easy" if it can be solved in a manner that is both incentive-compatible and computationally tractable. The technical definitions of incentive compatibility and computational tractability will depend on the particular problem under consideration.

The discussion in Section 3.1 shows that *welfare-maximizing multicast cost sharing* is easy when strategyproofness is the incentive-compatibility requirement, and low absolute network complexity is the computational-tractability requirement. The first open problem is to determine how general this result is. Recall that the MC mechanism is the only strategyproof and efficient mechanism that satisfies NPT and VP. If we remove the NPT and VP requirements, then we have the entire family of VCG mechanisms at our disposal. How many of these have reasonable network complexity?

Open Problem 1. *Fully characterize the set of easy welfare-maximizing multicast cost sharing problems.*

Of course, we are interested in far more than just multicast cost sharing, and one of the central DAMD challenges is the search for additional examples.

Open Problem 2. *Design good distributed algorithmic mechanisms to show that natural problems of interest are easy.*

While easy problems are a field's "successes," hard problems often lead to a deeper understanding of an approach's fundamental limitations. Thus, we are interested in how to define hardness in the DAMD context. Superficially, a problem is *hard* if it cannot be solved in a manner that satisfies both the incentive-compatibility and the computational-tractability requirements. There will be many problems for which this cannot be done; NP-hard problems, for example, cannot be solved in a computationally tractable manner (unless P=NP), and there are no efficient, strategyproof, and budget-balanced solutions to general cost-sharing problems. However, we are not interested in hardness *per se* but rather in hardness that results from the interplay of incentive compatibility and computational complexity. Thus, a more useful distinction is made by defining a DAMD problem to be *canonically hard* if each of these two requirements can be satisfied individually, but they cannot be satisfied simultaneously. Canonical hard problems will help us understand the fundamental nature of hardness in DAMD, as opposed to hardness that results solely from computational issues or solely from incentive issues.

Budget-balanced multicast cost sharing, under a few natural incentive-compatibility restrictions, is canonically hard. Here, the computational-tractability requirement is low absolute network complexity, as it is for welfare-maximizing multicast cost sharing. The incentive-compatibility conditions include the aforementioned group strategyproofness, NPT, and VP and the following two additional requirements:

[6]BGP does not currently consider general path costs; it simply computes *shortest* AS paths in terms of number of AS hops. However, BGP could be trivially modified so that it computes LCPs; in what follows, we assume that this modification has been made.

[7]Lying could increase an AS's total welfare by either attracting more traffic, and thereby increasing revenue, or increasing the price, or decreasing the costs incurred. In particular, if an AS declared an infinite cost, it would carry no transit traffic at all, and thus not incur any related transit costs.

[8]The tables in [19] contain both LCPs (as do BGP tables) and costs and prices.

[9]Although it can be done in a BGP-compatible fashion, we do not expect the pricing scheme of [19] (or any similar scheme) to be deployed in the near future. This first work on DAMD for interdomain routing is not an attempt to "solve a BGP problem." Rather, it is an attempt to study algorithmic mechanism design for routing in a computational model that is faithful to the Internet-centric motivation for the study; this was not done in the earlier papers [52, 30] on routing mechanisms.

CS Consumer Sovereignty: For given $T(P)$[10] and link costs $c(\cdot)$, there exists some κ such that $\sigma_i(v) = 1$ if $v_i \geq \kappa$; this condition ensures that the network cannot exclude any agent who is willing to pay a sufficiently large amount, regardless of other agents' valuations.

SYM Symmetry: If i and j are at the same node or are at different nodes separated by a zero-cost path, and $v_i = v_j$, then $x_i = x_j$.

The SH mechanism defined in Section 3.1 is the natural group-strategyproof, budget-balanced mechanism to consider, for reasons discussed at length by Moulin and Shenker [47], but it is only one of several group-strategyproof, budget-balanced mechanisms in the literature that have properties NPT, VP, CS, and SYM; see [16, 33] for more examples. It is shown in [18] that no group-strategyproof, budget-balanced multicast cost-sharing mechanism that satisfies conditions NPT, VP, CS, and SYM can have low absolute network complexity.

Thus, for this problem, one cannot simultaneously meet the computational-tractability requirement and the incentive-compatibility requirement. However, one can meet each requirement independently. The SH mechanism satisfies the incentive-compatibility requirement, and one can easily obtain budget-balanced cost-sharing "mechanisms" with low absolute network complexity if incentive issues are ignored. For instance, in one bottom-up pass of $T(P)$, one can compute $V = \sum_{i \in P} v_i$ and $C = \sum_{l \in L} c(l)$. If $C > V$, no one receives the transmission, and the mechanism does one top-down pass to inform all members of P that this is the outcome; if $C \leq V$, everyone receives the transmission, and the mechanism does one top-down pass to communicate the cost share $(C \cdot v_i)/V$ to agent i, for all $i \in P$.

Group-strategyproof, budget-balanced multicast cost sharing with the additional restrictions of NPT, VP, CS, and SYM is the only canonically hard DAMD problem that has been identified so far. To gain a greater understanding of DAMD, we need many more examples.

Open Problem 3. *Find more DAMD problems that are canonically hard.*

These informal descriptions of what we mean by "easy" and "hard" suffice for the analysis of some examples, but a formal framework is needed if we are to go beyond examples and develop a full-fledged "complexity theory of Internet computation."

Open Problem 4. *Define the computational models and computational resources needed to formalize "network complexity," both absolute and relative, and other relevant measures of DAMD complexity. Develop the appropriate notions of "reduction" to show that certain problems are hard or complete for the relevant complexity classes.*

The preceding discussion considered hardness of DAMD problems. One can also consider hardness of AMD problems by using notions of computational tractability that are appropriate in a centralized computational model. However, we are not aware of a canonically hard AMD problem.[11] All unsuccessful attempts to devise computationally tractable, centralized algorithmic mechanisms that we are aware of fail either because incentive compatibility is unattainable (*e.g.*, budget-balanced and welfare-maximizing cost sharing) or because computational tractability is unattainable (*e.g.*, NP-hard welfare maximization in combinatorial auctions) but not because of the interplay of the two. An interesting open question is whether such canonically hard AMD problems exist.

5. APPROXIMATION

Given that some DAMD problems are hard, it is natural to ask whether *approximate* versions of these DAMD problems are easy. In order to study this question, we must decide what it means to "approximate" a mechanism. For concreteness, we first discuss approximations of multicast cost-sharing mechanisms, one of which we know to be canonically hard and for which we already have the necessary terminology and notation.

Recall that a multicast cost-sharing mechanism is a pair of functions (σ, x). One may be tempted to define an approximation of the mechanism as a pair of functions (σ', x') such that σ' approximates σ well (for each v, these are characteristic vectors of subsets of P; so, we may call σ' a good approximation of σ if, for each v, the Hamming distance between the vectors is small), and x' approximates x well (in the sense, say, that, for some p, the L^p-difference of $x(v)$ and $x'(v)$ is small, for each v). However, the mechanism (σ', x') may not have the desired game-theoretic properties. For example, if (σ, x) were strategyproof but (σ', x') were not strategyproof, agents might misreport their valuations to the approximate mechanism. Thus, even if (σ, x) and (σ', x') were, for each v, approximately equal as pairs of functions, the resulting equilibria might be very different, *i.e.*, $(\sigma'(v'), x'(v'))$ might be very far from $(\sigma(v), x(v))$, where v' is the reported valuation vector when using the approximate mechanism (σ', x').

Thus, we first consider "approximate mechanisms" that retain the strategic properties (*e.g.*, strategyproof or group-strategyproof) of the mechanisms that they are approximating. In addition, if the original mechanism has some property, such as budget balance or efficiency, that does not relate to the underlying strategic behavior of agents but is an important design goal of the mechanism, then the approximate mechanism must approximate that property well.

The SH mechanism is group-strategyproof, budget-balanced, and, among all mechanisms with these two properties, the unique one that minimizes the worst-case efficiency loss.[12] Therefore, in a distributed computational model (where SH is canonically hard [18]), one should strive for a group-strategyproof mechanism that has low network complexity

[10]For brevity, we often use $T(P)$ to denote four components of a multicast cost-sharing problem instance: the node-set N, the link-set L, the locations of the agents, and the multicast-source location α_s.

[11]The question of whether there are any such problems was brought to our attention by Eric Friedman.

[12]The efficiency loss of a mechanism M on a particular problem instance I is the difference between the optimal net worth of I and the net worth realized by M. The SH mechanism minimizes the worst-case loss in the following sense. For each group-strategyproof, budget-balanced M and each instance size k, there is a worst-case instance $I_{M,k}$, *i.e.*, one for which efficiency loss $L(I_{M,k})$ is largest; for all k, SH achieves the minimum, over all group-strategyproof, budget-balanced M, of $L(I_{M,k})$, and it is the only such mechanism that does so.

and is approximately budget-balanced and approximately efficiency-loss minimizing in the worst case. "Approximately budget-balanced" can be taken to mean that there is a constant $\beta > 1$ such that, for all $c(\cdot)$, $T(P)$, and v:

$$(1/\beta) \cdot c(T(R(v))) \leq \sum_{i \in R(v)} x_i(v) \leq \beta \cdot c(T(R(v)))$$

Similarly, the statement that mechanism M is "approximately efficiency-loss minimizing in the worst case" can be taken to mean that there is a constant $\gamma > 1$ such that, for all k, the worst-case efficiency loss of M on instances of size k is at most γ times the worst-case efficiency loss of SH on instances of size k.

Progress has been made on the network complexity of approximate SH mechanisms, but the problem has not been solved completely. The SSF (for "scaled step function") mechanism given in [5] is group-strategyproof and fails to achieve exact budget balance and exact minimum worst-case efficiency loss by bounded amounts, but the bounds are not constant factors; therefore, SSF is not an approximation to SH by the definition given above, but it is a step towards such an approximation.

Open Problem 5. *Is there an approximate SH mechanism whose network complexity is as good as that of SSF? Is there one that achieves the network-complexity lower bound given in [18] for group-strategyproof, approximately budget-balanced mechanisms? If not, is there another group-strategyproof, budget-balanced mechanism that can be approximated in a manner that meets this lower bound?*

The notion of "approximating a mechanism M" that we have discussed so far (and that is used in [5] to study the SH mechanism) is, roughly, "retain the strategic properties of M but approximate the other mechanism-design goals." In what follows, we will call these *strategically faithful* approximations – they retain the strategic properties of the original mechanism exactly and approximate one or more of its other properties. This type of approximation is studied in the economics literature as well (*e.g.*, [37]), but not for the purpose of reducing computational or communication costs.

An important and open research issue is to explore alternative notions of approximation, as well as to design computationally tractable, strategically faithful approximations for more DAMD problems. Several alternative notions have been put forth, and we briefly review three of them here.

Approximation of equilibria is considered in the game-theory literature independent of distributed-algorithmic concerns.[13] For example, Schummer [60] and Parkes *et al.* [56] consider *ϵ-dominance*. A strategy vector $(s_1, \ldots, s_n)$ is an ϵ-dominant equilibrium if, for every agent i, every strategy t_i, and every set of other players' strategies $(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n)$, the inequality $u_i(M(t_1, \ldots, t_{i-1}, s_i, t_{i+1}, \ldots, t_n)) + \epsilon \geq u_i(M(t_1, \ldots, t_{i-1}, t_i, t_{i+1}, \ldots, t_n))$ holds – so, s_i may not be a best possible strategy, but it is always within ϵ of best possible, regardless of what other agents do. In the AMD context, one also insists that, for all i, the computation of s_i is tractable, given the information available to agent i, and that the computation of $M(\cdot)$ is tractable, given a strategy vector $(s_1, \ldots, s_n)$.

[13] One can consider the *virtual implementation* literature [39, 1], in which mechanisms produce lotteries over outcomes, to be a form of approximation. We don't discuss virtual implementation here.

Nisan and Ronen [53] were the first to address the question of approximate computation in AMD. They considered VCG mechanisms in which optimal outcomes are NP-hard to compute (as they are in combinatorial auctions). They pointed out that, if an optimal outcome is replaced by a computationally tractable, approximately optimal outcome, the resulting mechanism may no longer be strategyproof. The above discussion of how we should define "approximating the SH mechanism" and why approximating the pair of functions (σ, x) is not sufficient is based on the analogous observation in the DAMD context. Nisan and Ronen [53] develop the notion of *feasibly dominant strategies* and *feasibly strategyproof mechanisms*. For each agent i in such a mechanism, there may exist instances in which there is a strategy s_i that would be better for him than truth telling; however, in all instances, computational or informational limitations (or both) make it infeasible for agent i to find any such s_i. Because *all* agents are unable to find any strategies that are better than truth telling, the mechanism is "feasibly strategyproof." A broad class of situations in which NP-hard VCG mechanisms have feasibly strategyproof approximations is given in [53]. This approach is not directly applicable to SH-mechanism approximation for two basic reasons. First, SH is not a VCG mechanism. Second, the issue in SH-approximation is not polynomial-time approximation of an NP-hard optimization problem in a centralized computational model but rather low-network-complexity approximation of an intrinsically high-network-complexity (but polynomial-time computable) mechanism in a distributed computational model. However, the notion of feasible strategyproofness may have a distributed-computational analog and/or an analog that is applicable to non-VCG mechanisms.

A third approach to approximation is suggested by one of the questions studied in [5]. The MC mechanism is strategyproof, but it is not group-strategyproof, as is the SH mechanism to which it is being compared. How serious a drawback is this? This question is formalized in [5] as follows: (1) For which instances of the MC mechanism (*i.e.*, which trees, link costs, and valuation-vectors $(v_1, \ldots, v_n)$) can the agents collude successfully? (2) On the instances that are subject to manipulation by coalitions, what forms do the successful coalitions take? (3) What is the effect of this strategic manipulation, *e.g.*, how many agents receive the transmission that would not have received it in the absence of collusion, and how many wind up paying less than they would have in the absence of collusion? The results in [5] are negative; essentially, most instances of MC are subject to devastating manipulation by a wide class of coalitions. However, the approach warrants further study, because it may be applicable to other mechanism-design problems. Informally, we say that a mechanism is *tolerably manipulable* with respect to group strategyproofness if it is not group-strategyproof, but the groups that can strategize successfully are fully characterizable, and their effects on the overall performance of the system are deemed to be tolerable. Similarly, one can consider mechanisms that are tolerably manipulable with respect to other relevant solution concepts. Tolerably manipulable mechanisms may offer significant practical advantages in terms of network complexity.

Open Problem 6. *Thoroughly investigate the notion of approximation in the DAMD context. In particular, explore the applicability and the limitations of computation-*

ally tractable ϵ-dominance, strategically faithful approximation, feasible strategyproofness, and tolerable manipulability, and formulate new definitions of approximation if they are needed.

Open Problem 7. *Devise approximations for the canonically hard DAMD problems identified in the answer to Open Problem 3.*

Finally, there are approximation questions in mechanism design that are not computational in nature. Recall that there are no strategyproof multicast cost-sharing mechanisms that are both efficient and budget-balanced [27, 58]. One could conclude from the results in [20, 18, 47] that MC is the only strategyproof mechanism that can be practically deployed on the Internet. This is a disheartening conclusion, because MC can be arbitrarily budget-imbalanced.

Instead of abandoning budget balance altogether, one could seek a compromise via a strategically faithful mechanism. That is, although it is known that no strategyproof mechanism can achieve both exact efficiency and exact budget balance, it is conceivable that one could be exactly efficient and approximately budget-balanced, exactly budget-balanced and approximately efficient[14], or approximately both. Unfortunately, this more modest goal is also unattainable; it is shown in [18] that no strategyproof multicast cost-sharing mechanism that satisfies NPT, VP, and CS can be both approximately efficient and approximately budget-balanced. Thus, the most promising approach to practically deployable cost-sharing mechanisms appears to be SH approximation, as discussed above.

This discussion suggests a more general question that deserves further study.

Open Problem 8. *How do the characterization results for strategyproof mechanisms change if the nonstrategic design goals need only be approximated?*

Characterization results that should be considered include the Moulin-Sprumont characterization of strategyproof mechanisms with single-peaked utilities [45, 63] and the Barbera-Jackson characterization of strategyproof exchange markets [8].

6. STRATEGIC MODELS

In this section, we look more carefully at an aspect of DAMD that we have thus far oversimplified. It need not be the case that all parties in a distributed, algorithmic mechanism are simply selfish maximizers. In many realistic scenarios, there are (at least) four types of entities that participate in the protocol:

- *Obedient nodes* are correctly functioning computers. They have no independent strategic goals and simply do what they are programmed to do.

- *Faulty nodes* are incorrectly functioning computers. They also have no independent strategic goals, but they suffer from, *e.g.*, hardware- or software-bugs or misconfigurations.

- *Strategic nodes* are the selfish agents of game theory. Each has utility maximization as its goal.

- *Adversarial nodes* are the various sorts of "enemies" found in cryptographic-protocol theory; they range from "honest but curious" enemies (who follow the protocol when called upon to send messages to other parties but may use what they learn from the protocol, possibly in collusion with other cheaters, to learn something that they are not supposed to know) to Byzantine enemies (who may deviate from the prescribed protocol in arbitrary ways). In particular, mechanism designers cannot assume that the goals of an adversarial node are captured by a utility function; in economic terms, such a node may be "irrational."

Collusion is another important aspect of the strategic model. In some contexts the faulty, adversarial, or strategic behaviors of nodes are coordinated. Thus, one must ask, *e.g.*, whether or not various subsets of the agents can collude before or during the execution of the protocol and whether the coalitions are static or dynamic. For simplicity, we do not discuss collusion explicitly in the rest of this section, but it is an important issue about which many open issues remain.

Recall that there are two aspects to mechanisms: The strategic aspect involves the choice of inputs to the mechanism (the strategies), and the computational aspect involves the calculation of the mechanism M (that is a function of the strategies). These two aspects are cleanly separated in scenarios in which the computations are carried out on obedient nodes, and the strategic nodes merely supply their inputs to this computational infrastructure. This is the case in, *e.g.*, [52, 30], where the mechanism is executed on an obedient, centralized computational device that is distinct from the strategic agents who supply the inputs; similarly, it is the case for previous work on distributed multicast cost-sharing mechanisms [5, 3, 18, 20, 22, 33, 41], where the mechanism is executed on an obedient, distributed computational device (*i.e.*, the multicast tree) that is distinct from the strategic agents (who are resident at various nodes of the tree but not in control of those nodes).

However, the situation is significantly more complicated when the computational and strategic aspects become intertwined. For instance, an important issue not resolved in [19] is the need to reconcile the strategic model with the computational model. On the one hand, the problem formulation in [19] captures the fact that ASs may have incentives to lie about costs in order to gain financial advantage and provides a strategyproof mechanism that removes these incentives. On the other hand, it is these very ASs that carry out the distributed algorithm 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?

Mitchell *et al.* [43] have observed that, 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 in [19] can be modified so that all forms of cheating are detectable. Remaining open problems include:

Open Problem 9. *Can the protocol in [19] be modified to detect all forms of cheating without the addition of public-key infrastructure (or any other substantial new infrastruc-*

[14] *Approximately efficient* has the obvious meaning: There is a constant γ, $O < \gamma < 1$, such that the mechanism always chooses a receiver set R' with $NW(R') \geq \gamma NW(R)$, where R is the net-worth-maximizing set chosen by MC.

tural or computational capability) to the BGP-based computational model?

Open Problem 10. *Can digital signatures (or, more generally, cryptographic protocol-design techniques) always be used to convert a distributed algorithmic mechanism in which some of the parties must be assumed to be obedient into one with a more realistic strategic model?*

Mitchell and Teague [44] have used cryptographic techniques for multicast cost-sharing protocols. For example, they consider the case in which the nodes of the multicast tree are strategic rather than obedient, and the strategic goal of a node is to obtain the transmission for its resident agents at lower prices than those agents would have to pay if the nodes obediently executed the protocol given in [20] for the MC mechanism.

Open Problem 11. *Thoroughly investigate the interplay between strategic models and computational models in DAMD. In particular, develop realistic strategic models for a variety of DAMD problems, including problems in which one or more of the players are adversarial or faulty. If possible, develop general techniques for converting distributed algorithmic mechanisms in which some of the parties must be assumed to be obedient into ones in which all parties are realistically strategically modeled.*

Web caching may be a particularly interesting DAMD challenge with respect to Open Problem 11, because all types of agents may be present.

7. INDIRECT MECHANISMS

Most of the work to date on algorithmic mechanisms in the TCS community has focused on strategyproof, direct mechanisms. The underlying premise of this approach is that agents will voluntarily reveal their private information if it can be proven that lying does them no good in the situation addressed by this particular mechanism-design exercise. We question this premise. Indeed, the TCS community generally questions this premise, which it did not invent but rather inherited from the economics community. Revelation of private information may be in an agent's best interest in the particular game at hand, but it may be unacceptable in the broader context.

For example, in the interdomain-routing mechanism of [19] discussed in Section 3.2 above, ASs are expected to reveal their internal per-packet transit costs, and conventional economic wisdom would have it that they'd be willing to do so, because the mechanism is strategyproof. However, this seems unrealistic: Revealing its true transit costs may reveal details about an AS's internal network that it wants to keep private for reasons that have nothing to do with near-term transit-traffic revenues.

More fundamentally, the real mechanism-design goal is not to convince agents to reveal their private inputs but rather to compute the desired result that depends on these inputs. The economics literature does not emphasize the fact that these are distinct goals, but the distinction a major focus of the TCS literature. The theory of *secure, multiparty function evaluation* (SMFE), developed by the cryptographic-research community, shows that functions can often be computed in such a way that nothing about agent i's private input need be revealed to agent j (except what is logically implied by the outcome and agent j's private input). In economic terms, the SMFE approach would lead to indirect mechanisms, because agents would not be revealing their utilities but instead would be using strategies drawn from some other strategy space. For an overview of SMFE, see Goldreich [26].

One cannot always apply SMFE techniques "off the shelf" to DAMD. In particular, one often cannot "compose" a direct distributed algorithmic mechanism with a standard SMFE protocol, for several fundamental reasons:

- The strategic models may be different. Some standard SMFE techniques (*e.g.*, [9, 11]) apply to networks in which at least a constant fraction of the agents are obedient; the other agents are often assumed to be Byzantine adversaries. Although one usually does not have to design distributed mechanisms for Byzantine adversarial agents, one often has to assume that *all* of the agents will act strategically – *none* can be assumed to be obedient.

- Standard SMFE techniques for transforming an arbitrary multi-agent protocol into one that keeps agents' inputs private and computes the same output produce protocols with unacceptably high network complexity. In particular, the required total number of messages may grow quadratically (or worse) as a function of the total number of agents. Sometimes special-purpose, SMFE protocols with low network complexity are obtainable, but, if these are to be found for DAMD problems of interest, they will have to be designed on a case-by-case basis; no general SMFE results guarantee their existence.

- Some of the standard building blocks of protocols in the SMFE literature (notably secret sharing) assume that each agent knows the *set* of all agents participating in the protocol and can refer to each of them using a unique ID. Clearly this is not the case in all DAMD problems; in particular, it is not the case in the multicast cost-sharing problem, where two agents resident at different nodes of the multicast tree must be assumed to be ignorant of each other's existence.

The agent-privacy issue was first raised in an early paper of Nisan [50], as was the potential applicability of SMFE techniques. However, there has not yet been substantial progress toward a general theory of distributed algorithmic mechanisms that keep agents' inputs private. Thus, it remains an important open problem.

Open Problem 12. *Explore agent privacy in specific DAMD problems of interest. More generally, devise new building blocks for SMFE protocols that are applicable in the DAMD context, where all agents can be strategic (i.e., none need be obedient or adversarial), low network complexity is crucial, and the set of participating agents is unknown to each individual agent.*

In addition to the standard SMFE literature [26], work that might be relevant to Open Problem 12 includes but is not limited to the papers of Naor and Nissim [48], Naor, Pinkas, and Sumner [49], and Dodis, Halevi, and Rabin [15].

Thus far, our motivation for considering indirect mechanisms has been the desire to preserve agents' privacy, and we have observed that low network complexity and agent privacy may be hard to achieve simultaneously. It is important to note, however, that indirect mechanisms have not been shown to have inherently higher network complexity than direct mechanisms.

Open Problem 13. *Are there DAMD problems for which all direct mechanisms have bad network complexity but at least one indirect mechanism has good network complexity?*

Indirect mechanisms might have other advantages as well, such as enabling one to trade off between agent computation and mechanism computation or to avoid worst-case running times on many instances [55]. They might also enable approximation. These benefits have been demonstrated in the context of combinatorial auctions and should be explored in other contexts as well.

8. SOLUTION CONCEPTS

To date, most of the DAMD work in the theoretical computer science community has focused on the dominant-strategy solution concept. One of the justifications for this focus is that agents resident in different (and possibly competing) Internet domains would not even necessarily be aware of each other's existence, much less be able to observe enough about each other's actions to make the *best responses* needed for Nash equilibria and other solution concepts that are more mainstream in the economics literature. This justification may not be valid for some of the DAMD problems now under investigation. For example, it may not be valid for interdomain routing, where the agents are the domains themselves and, in the course of running BGP, acquire partial but significant information about the global AS graph. It is possible that enough "common knowledge" is acquired by the strategic agents to make dominant-strategy solutions not the only sensible option for interdomain routing; similar reasoning may apply to other DAMD problems.

Open Problem 14. *Which solution concepts are most interesting for DAMD? In particular, what is the "right" solution concept for interdomain routing?*

The game-theory literature contains a very rich set of solution concepts to which we cannot possibly do justice here. For example, one can consider "repeated-play" settings in which each agent knows its own payoff function but not those of others; as the game continues, agents learn more about the payoff functions of others, adapt their own strategies, and reach an equilibrium through repeated play. The resulting solution concept depends on how the agents learn. If agents are *adaptive learners*, according to the Milgrom-Roberts definition [42], then the play asymptotically heads towards the serially undominated set. If the learning algorithms are *calibrated* in the sense of Foster and Vohra [23], then the solution concept is the set of correlated equilibria.

Internet-based agents typically have very little information even about their own payoff functions and know still less about those of other agents. All they know is which strategy they played and what the resulting payoff was. In addition, the payoff functions change over time in unpredictable and sudden ways, reflecting the dynamic nature of the Internet infrastructure. Furthermore, play is highly asynchronous, because agents adapt their strategies at different rates. New definitions of learning and new solution concepts that attempt to capture these aspects of Internet-based, repeated games have been proposed by, *e.g.*, Erev and Roth [17] and Friedman and Shenker [24].

Which, if any, of these solution concepts and game-theoretic definitions of learning will play a central role in the emerging theory of DAMD is a wide open question. Moreover, we have yet to understand the relationship between the solution concept and the network complexity of mechanisms designed for that solution concept. Are there solution concepts that typically lead to mechanisms with lower network complexity? Are these solution concepts realistic in a network setting?

Open Problem 15. *Fully explore the relationship between solution concepts and network complexity.*

9. INTERNET APPLICATIONS OF DAMD

In this section, we turn our attention from general and foundational issues in DAMD to specific mechanism-design challenges now faced by Internet researchers. For the sake of brevity, we only discuss four possible application areas – web caching, peer-to-peer file sharing, overlay networks, and distributed task allocation – and we do not attempt to provide complete references to the vast literatures on each of these subjects.

9.1 Web Caching

Web caches are an important tool for enhancing the performance of web access; they are used to eliminate hot spots in the network and to reduce access latencies. A web-caching architecture provides the framework for a set of collaborating caches to interact with each other and serve a client community. A wide variety of caching architectures have been proposed; their common intent is to achieve overall system efficiency, and their common assumption is that caches are obedient. This assumption may be valid when the entire caching infrastructure belongs to the same administrative entity, but, when the caching infrastructure spans administrative boundaries, the caches might deviate from the protocols to maximize their individual welfare.

There are two fundamental incentive issues in caching. First, when considering a single cache, the utility of the requesting clients would be maximized by caching the frequently requested pages that provide the highest user utility. However, the only way that a cache can learn about these valuations is from the clients themselves. One needs strategyproof mechanisms to elicit truthful valuations from clients.

Second, caches have limited resources (bandwidth and storage) and incur a cost for storing a page or serving a client request. When there are several caches collaborating, but they are managed by separate economic entities, there is a question of how to design mechanisms to distribute the caching load. If the caches are not reimbursed for the cost of serving pages, they have an incentive to manipulate other caches into serving the pages and thereby avoid bearing the operating costs. If the caches receive payments for serving pages, they might "compete" with each other to serve the highest-value pages, thereby duplicating each other's content, leaving some important pages uncached, and yielding sub-optimal performance. Thus, the system needs to provide incentives designed to have the caches report their true operating costs.

Open Problem 16. *Develop distributed algorithmic mechanisms for caching in which clients are induced to reveal their true preferences, and caches are induced to implement the optimal resource allocation. Demonstrate that these caching mechanisms can be built as scalable extensions to existing distributed caching protocols and provide provable performance guarantees.*

Some aspects of this problem are addressed in, *e.g.*, [34].

9.2 Peer-to-Peer File Sharing

Peer-to-peer (P2P) file-sharing networks, *e.g.*, Gnutella, KaZaA, and Freenet, are the *ne plus ultra* of autonomous distributed systems; each machine belongs to a different user, and there is no central administrative authority. Thus, incentive issues are likely to be very important to the future of P2P technology; this provides a great opportunity for DAMD research.

P2P file-sharing represents a shift from purely commercial content-distribution systems (*e.g.*, content providers paying CDNs to distribute their content) to a "gift economy," in which individual users offer up their resources – content, access bandwidth, storage, and CPU – for the greater good. However, initial studies show that there is a serious "free rider problem" [2]. It may be that, without some systematic way of providing incentives for users to share their files, these P2P systems will become increasingly centralized, with only a few commercially supported nodes offering to share files.

One initial attempt to provide incentives to participants is the MojoNation P2P system, which awards users "mojo" for offering resources [40]. "Mojo" can be used to gain priority access when the system is overloaded.

While the original P2P systems operate as gift economies, MojoNation can be thought of as a barter economy, in which mojo is exchanged in return for services, but no money is transferred. The exchange of money would allow for a much wider range of possible economic mechanisms. The question is whether this extension to monetary exchanges would result in superior performance.

Open Problem 17. *What are the performance characteristics of the equilibria that result from barter P2P systems? In particular, are they optimal, competitive with optimal, or highly suboptimal?*

Open Problem 18. *Can one design* monetary *P2P systems that provide better performance than purely barter P2P systems? Can one characterize, in simple models, the possible outcomes achievable with both kinds of P2P economies?*

Most of the recent models of P2P systems consider a collection of identical nodes. Although this is theoretically appealing, it is contradicted by preliminary measurement studies suggesting that there is a very wide range of node capabilities (in bandwidth, CPU, and disk) in P2P systems [59]. Wide heterogeneities may lead to significantly increased efficiency in P2P systems; that is, the highly capable nodes can act as semi-centralized repositories.

Open Problem 19. *How does the performance of the equilibria of P2P systems, both barter and monetary, change when the user population exhibits extreme variability in node capability?*

The above questions address mostly the extent to which users share files with other P2P users. But in fully decentralized P2P systems, nodes function both as caches of shared files and as routers for queries destined for other nodes. In that sense, the P2P-incentive issues are a union of the issues in routing and caching. However, it isn't clear how to model the incentive aspects of P2P routing, and this is also an open question worthy of study.

9.3 Application-layer overlay networks

Many distributed systems form an application-layer network out of their constituent nodes. For instance, in many P2P file-sharing systems, each node has a set of neighbors to whom it forwards queries. There have also been many proposals for application-layer networks to perform unicast routing (*e.g.*, [4]) and multicast routing (*e.g.*, [12]). These overlay networks are formed by algorithms that assume nodes are obedient and ignore their own incentives. Clearly, obedience may not be in a node's best interest. In the case of P2P file sharing, nodes would want to be close to others who share lots of files (so that their own queries could be answered quickly) but far away from others that generate lots of queries (to minimize the time they spend processing them). In routing, a node would want to minimize the maximal distance to other nodes but would also not want to carry much traffic, and so it would prefer not to be on many LCPs. This poses the question of what kinds of networks would result if users were selfish and chose their neighbors accordingly.

Open Problem 20. *What are the characteristics of overlay networks formed by selfish users?*

We expect the answer to depend on the particular system, *e.g.*, selfishly constructed overlay file-sharing networks will likely be different from selfishly constructed overlay routing networks.

One of the purposes of these overlay networks is to choose routes that improve end-to-end latency and availability. The overlay network can be seen as a "selfish" entity, picking LCPs for its traffic without concern for the overall network's performance. What happens if we have many different overlay networks? It may be that the advantages of overlay networks are undermined by the result of competition among them.

Open Problem 21. *What is the result of competition among many "selfish" overlay routing networks? Can a well designed distributed algorithmic mechanism incentivize behavior that preserves the observed benefits of overlays in the presence of competition?*

9.4 Distributed task allocation

Instead of P2P file sharing, in which users share their storage capacity, or application-layer routing, in which users share their communication capacity, many users can participate in a CPU-intensive task to share their CPU capacity. We have already seen many users work together on tasks such as integer factoring, signal processing, and protein folding. However, these examples depend on voluntary user actions, and therefore many users may not want to participate. Furthermore, in the future, many users may use weak, mobile computational devices, and therefore service providers may be set up so that users can offload CPU-intensive tasks to them. For a large task, it may be beneficial to collaborate and distribute a task among multiple "CPU-service providers." In their original paper [52], Nisan and Ronen proposed a centralized task-allocation mechanism called *MinWork*.

Open Problem 22. *Design distributed task-allocation mechanisms for efficient sharing of the CPU resources of many users or service providers. Take into account the differences among the resources, e.g., storage, communication, and CPU.*

10. ACKNOWLEDGEMENTS

We thank Dirk Bergemann, Eric Friedman, Arvind Krishnamurthy, Noam Nisan, Christos Papadimitriou, David Parkes, Rahul Sami, and Richard Yang for many stimulating discussions about this work. We are also grateful to seminar participants at DIMACS, Stanford, and NEC for their comments.

11. REFERENCES

[1] D. Abreu and A. Sen, "Virtual Implementation in Nash Equilibrium," *Econometrica* **59** (1991), pages 997-1022.

[2] E. Adar and B. Huberman, "Free Riding on Gnutella," in *First Monday*, `http://www.firstmonday.dk/issues/issue5_10/adar/index.html`, October 2000.

[3] M. Adler and D. Rubenstein, "Pricing Multicast in More Practical Network Models," in *Proceedings of the 13th Symposium on Discrete Algorithms*, ACM Press/SIAM, New York/Philadelphia, pages 981–990, 2002.

[4] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient Overlay Networks," in *Proceedings of the 18th Symposium on Operating System Principles*, ACM Press, New York, pages 131–145, 2001.

[5] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Approximation and Collusion in Multicast Cost Sharing," submitted. `http://www.cs.yale.edu/homes/jf/AFKSS.ps`

[6] A. Archer and E. Tardos, "Frugal path mechanisms," in *Proceedings of the 13th Symposium on Discrete Algorithms*, ACM Press/SIAM, New York/Philadelphia, pages 991–999, 2002.

[7] K. Arrow, "The Property-Rights Doctrine and Demand Revelation under Incomplete Information," in **Economics and Human Welfare**, Academic Press, New York, pages 23–39, 1979.

[8] S. Barbera and M. Jackson, "Strategy-proof exchange," *Econometrica* **63** (1995), pages 51–87.

[9] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation," in *Proceedings of the 20th Symposium on Theory of Computation*, ACM Press, New York, pages 1–10, 1988.

[10] B. Bernheim, "Rationalizable Strategic Behavior," *Econometrica* **52** (1984), pages 1007–1028.

[11] D. Chaum, C. Crépeau, and I. Damgaard, "Multiparty Unconditionally Secure Protocols," in *Proceedings of the 20th Symposium on Theory of Computation*, ACM Press, New York, pages 11–19, 1988.

[12] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," in *Proceedings of ACM Sigmetrics '00*, ACM Press, New York, pages 1–12, 2000.

[13] E. Clarke, "Multipart pricing of public goods," *Public Choice* **11** (1971), pages 17–33.

[14] C. d'Aspremont and L.-A. Gérard-Varet, "Incentives and Incomplete Information," *Journal of Public Economics* **11** (1979), pages 25–45.

[15] Y. Dodis, S. Halevi, and T. Rabin, "A Cryptographic Solution to a Game-Theoretic Problem," in *Advances in Cryptology – Crypto 2000*, Lecture Notes in Computer Science, volume 1880, Springer, Berlin, pages 112–130, 2000.

[16] B. Dutta and D. Ray, "A concept of egalitarianism under participation constraints," *Econometrica* **57** (1989), pages 615-635.

[17] L. Erev and A. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *American Economic Review* **88** (1998), pages 848–881.

[18] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker, "Hardness Results for Multicast Cost Sharing," Yale University Technical Report YALEU/DCS/TR1232, June 2002, `http://ftp.cs.yale.edu/pub/TR/tr1232.ps`

[19] 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.

[20] J. Feigenbaum, C. Papadimitriou, and S. Shenker, "Sharing the Cost of Multicast Transmissions," *Journal of Computer and System Sciences* **63** (2001), pages 21–41.

[21] D. Ferguson, C. Nikolaou, and Y. Yemini, "An Economy for Flow Control in Computer Networks," in *Proceedings of the 8th Infocom*, IEEE Computer Society Press, Los Alamitos, pages 100–118, 1989.

[22] A. Fiat, A. Goldberg, J. Hartline, and A. Karlin, "Competitive Generalized Auctions," in *Proceedings of the 34th ACM Symposium on Theory of Computing*, ACM Press, New York, pages 72–81, 2002.

[23] D. Foster and R. Vohra, "Calibrated Learning and Correlated Equilibrium," *Games and Economic Behavior* **21** (1997), pages 40–55.

[24] E. Friedman and S. Shenker, "Learning and Implementation on the Internet," `http://www.icir.org/shenker/decent.ps`

[25] R. Gibbens and R. Kelly, "Resource Pricing and the Evolution of Congestion Control," *Automatica* **35** (1999), pages 1969–1985.

[26] O. Goldreich, "Secure Multi-party Computation (working draft, Version 1.1)," 1998. `http://philby.ucsd.edu/cryptoloib/BOOKS/oded-sc.html`

[27] J. Green, E. Kohlberg, J. and Laffont, "Partial equilibrium approach to the free rider problem," *Journal of Public Economics* **6** (1976), pages 375–394.

[28] T. Griffin and G. Wilfong, "An analysis of BGP convergence properties," in *Proceedings of SIGCOMM '99*, ACM Press, New York, pages 277–288, 1999.

[29] T. Groves, "Incentives in teams," *Econometrica* **41** (1973), pages 617–663.

[30] 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.

[31] M.-T. Hsiao and A. Lazar, “A game theoretic approach to decentralized flow control of Markovian queueing networks,” in *Performance ’87*, North-Holland, Amsterdam, pages 55–74, 1988.

[32] M. Jackson, “A Crash Course in Implementation Theory,” *Social Choice and Welfare*, **18** (2001), pages 655–708.

[33] K. Jain and V. Vazirani, “Applications of approximation to cooperative games,” in *Proceedings of the 33rd Symposium on the Theory of Computing*, ACM Press, New York, pages 364–372, 2001.

[34] T. Kelly, Y-M. Chan, S. Jamin, and J. MacKie-Mason, “Biased Replacement Policies for Web Caches: Differential Quality-of-Service and Aggregate User Value,” in *Proceedings of the 4th International Web Caching Workshop*, pages 1–10, 1999.

[35] Y. Korilis, A. Lazar, and A. Orda, “Architecting Noncooperative Networks,” *Journal on Selected Areas in Communications* **13** (1995), pages 1241–1251.

[36] J. Kurose and R. Simha, “A Microeconomic Approach to Optimal Resource Allocation in Distributed Computer Systems,” *IEEE Transactions on Computers* **38** (1989), pages 705–717.

[37] P. McAfee, “A Dominant-Strategy Double Auction,” *Journal of Economic Theory* **56** (1992), pages 434–450.

[38] E. Maskin, “Nash Equilibrium and Welfare Optimality,” *Review of Economic Studies* **66** (1999), pages 23–38.

[39] H. Matsushima, “A New Approach to the Implementation Problem,” *Journal of Economic Theory* **45** (1988), pages 128-144.

[40] J. McCoy, `http://www.mojonation.net/`

[41] A. Mehta, S. Shenker, and V. Vazirani, “Profit-Maximizing Multicast Pricing by Approximating Fixed Points,” submitted. `http://www.cc.gatech.edu/people/home/aranyak/multicast.ps`

[42] P. Milgrom and J. Roberts, “Adaptive and Sophisticated Learning in Normal Form Games,” *Games and Economic Behavior* **3** (1991), pages 82–100.

[43] J. Mitchell, R. Sami, K. Talwar, and V. Teague, Private communication, 2001.

[44] J. Mitchell and V. Teague, Private communication, 2002.

[45] H. Moulin. “Strategyproofness and Single-peakedness,” *Public Choice* **35** (1980), pages 437–455.

[46] H. Moulin. “Incremental cost sharing: characterization by strategyproofness,” *Social Choice and Welfare* **16** (1999), pages 279–320.

[47] H. Moulin and S. Shenker, “Strategyproof Sharing of Submodular Costs: Budget Balance Versus Efficiency,” *Economic Theory* **18** (2001), pages 511–533.

[48] M. Naor and K. Nissim, “Communication-Preserving Protocols for Secure Function Evaluation,” in *Proceedings of the 33rd Symposium on Theory of Computing*, ACM Press, New York, pages 590–599, 2001.

[49] M. Naor, B. Pinkas, and R. Sumner, “Privacy-Preserving Auctions and Mechanism Design,” in *Proceedings of the 1st Conference on Electronic Commerce*, ACM Press, New York, pages 129–139, 1999.

[50] N. Nisan, “Algorithms for Selfish Agents: Mechanism Design for Distributed Computation,” in *Proceedings of the 16th Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, volume 1563, Springer, Berlin, pages 1–17, 1999.

[51] N. Nisan, “Rationality as a Paradigm for Internet Computing,” Invited Talk at the U. C. Berkeley Workshop on Theory of Computation & the Sciences, `http://www.cs.huji,il/~noam/rationality.ppt`.

[52] N. Nisan and A. Ronen, “Algorithmic mechanism design,” *Games and Economic Behavior* **35** (2001), pages 166–196.

[53] N. Nisan and A. Ronen, “Computationally feasible VCG mechanisms,” in *Proceedings of the 2nd Conference on Electronic Commerce*, ACM Press, New York, pages 242–252, 2000.

[54] C. Papadimitriou, “Algorithms, Games, and the Internet,” in *Proceedings of the 33rd Symposium on Theory of Computing*, ACM Press, New York, pages 749–753, 2001.

[55] D. Parkes, “Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency,” PhD Thesis, University of Pennsylvania, 2001.

[56] D. Parkes, J. Kalagnanam, and M. Eso, “Achieving Budget-Balance with Vickrey-Based Payment Schemes in Exchanges,” in *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, Morgan Kaufman, San Francisco, pages 1161–1168, 2001.

[57] D. Pearce, “Rationalizable Strategic Behavior and the Problem of Perfection,” *Econometrica* **52** (1984), pages 1029–1050.

[58] K. Roberts, “The Characterization of Implementable Choice Rules,” in **Aggregation and Revelation of Preferences**, North-Holland, Amsterdam, pages 321–348, 1979.

[59] S. Saroiu, P. Gummadi, and S. Gribble, “A Measurement Study of Peer-to-Peer File Sharing Systems,” in *Proceedings of Multimedia Computing and Networking*, SPIE Press, Bellingham, pages 156–170, 2002.

[60] J. Schummer, “Almost-dominant Strategy Implementation,” `http://www.kellogg.nwu.edu/faculty/schummer/ftp/research/esp.pdf`

[61] Y. Shoham and M. Wellman, “Economic Principles of Multi-Agent Systems,” *Artificial Intelligence* **94** (1997), pages 1–6.

[62] J. Smith, **Evolution and the Theory of Games**, Cambridge University Press, Cambridge, 1982.

[63] Y. Sprumont, “The division problem with single-peaked preferences: A characterization of the uniform rule,” *Econometrica* **59** (1991), pages 509–519.

[64] W. Vickrey, “Counterspeculation, auctions, and competitive sealed tenders,” *Journal of Finance* **16** (1961), pages 8–37.

[65] W. Walsh and M. Wellman, “A Market Protocol for Decentralized Task Allocation,” in *Proceedings of the 3rd International Conference on Multi-Agent Systems*, IEEE Computer Society Press, Los Alamitos, pages 325–332, 1998.