Rationality and Self-Interest in Peer to Peer Networks

Jeffrey Shneidman and David C. Parkes
Harvard University
Division of Engineering and Applied Sciences
{jeffsh, parkes}@eecs.harvard.edu

Abstract

Much of the existing work in peer to peer networking assumes that users will follow prescribed protocols without deviation. This assumption ignores the user's ability to modify the behavior of an algorithm for self-interested reasons.

We advocate a different model in which peer to peer users are expected to be rational and self-interested. This model is found in the emergent fields of Algorithmic Mechanism Design (AMD) and Distributed Algorithmic Mechanism Design (DAMD), both of which introduce game-theoretic ideas into a computational system. We, as designers, must create systems (peer to peer search, routing, distributed auctions, resource allocation, etc.) that allow nodes to behave rationally while still achieving good overall system outcomes.

This paper has three goals. The first is to convince the reader that rationality is a real issue in peer to peer networks. The second is to introduce mechanism design as a tool that can be used when designing networks with rational nodes. The third is to describe three open problems that are relevant in the peer to peer setting but are unsolved in existing AMD/DAMD work. In particular, we consider problems that arise when a networking infrastructure contains *rational* agents.

1 Introduction

Imagine running an auction over a large peer to peer network. You are the auctioneer and have three directly connected neighbors. You send out an announcement advertising the auction and ask that it be globally propagated. You then sit back, expecting many bids, and are surprised when you only receive three bids – one from each neighbor. What happened? Your three neighbors understood that it was in their best interest not to forward the initial announcement. Assuming each of your neighbors wanted to win the auction, it made sense for them to limit the possible competition.

This trivial example (which shall be used for the remainder of the paper) captures the problem of rationality in peer to peer networks. Perhaps the key defining characteristic of a peer to peer network is that one cannot distinguish between strategic nodes and the network infrastructure. Nodes, representing rational users, sometimes will deviate from a suggested protocol in order to better their own outcome. This paper considers notions of rationality and self-interest in the context of peer to peer networks. Section 2 explores evidence for the existence of rationality in peer to peer networks and discusses common approaches for dealing with this strategic node behavior. Section 3 reviews some of the requisite mechanism design tools and terminology. Section 4 introduces different node types (the strategic agents) to motivate why mechanism design is applicable to the peer to peer setting. Section 5 describes open problems in DAMD that are particularly relevant to peer to peer settings. Section 6 reviews related work and finally in Section 7, we conclude with our future work in this area.

2 Rationality in Peer to Peer Systems

In the auction example given above, we can speculate that all three bidders had positive utility for winning the auction. *Utility* is a numeric value that agents assign to a particular outcome based on their preference for that outcome. The game-theoretic approach of mechanism design assumes that a *rational* node plays a *strategy* to maximize its own expected utility. A node's equilibrium strategy depends on the information that it has about the preferences and behavior of other nodes, and in some cases may require some effort to calculate.

It is not hard to find evidence of rational behavior in existing peer to peer networks. There is interesting work documenting the "free rider" problem [1] and the "tragedy of the commons" [9] in a data centric peer to peer setting. In this situation, rational users free ride and consume a resource but do not produce at the same level of their consumption.

An additional example can be found in the context of peer to peer search: consider the rational users in simple file sharing networks who refuse to relay other node queries to conserve their own bandwidth. The Kazaa peer to peer file sharing network client supports a similar behavior, allowing powerful nodes to opt-out of network support roles that consume CPU and bandwidth [15]. Our auction example is another case where self-interested message passing and computation presents a challenge.

Although it has not been labeled as such, rational behavior has occurred in computational peer to peer settings as well. One perverse example occurred when users of Seti@Home (a peer to peer computation project) modified their client software to make it appear as if they were doing more work than was actually occurring. These users placed a high utility on their ranking in a leader board that recorded the "computation units contributed" for the Seti@Home project. The scoring system did not prevent these rational players from increasing their utility by modifying the behavior of their software [11].

What should be the response to the problems created by user rationality in peer to peer systems? One common approach has been to ignore rationality problems and hope for the best. Rational users can free ride on systems, create software designed to subvert mechanisms, and generally place themselves into selfishly advantageous situations. This sounds bad, and this paper argues that it is bad, but one should not ignore the fact that this is how many peer to peer systems work. One reason why these systems may work is that there can be enough obedient users following a given protocol, even when it might be rational not to do so. For instance, they might download an obedient client program instead of writing or acquiring a more expressive client. Alternatively, existing systems may work because there are enough "rational" users that maximize their expected utility by the enjoyment of providing a common good. This altruistic behavior is outside of typical game-theoretic models.

Another approach is to limit the effect that a rational user can have on the execution of a system. In peer to peer networks, a node can be required to run some part of a distributed mechanism. If one assumes that users cannot modify the execution of the mechanism (but can only strategize about their inputs), then the rationality problem is much easier. With this assumption, for instance, one would not need to worry about nodes refusing to pass messages in this paper's initial auction example. This is what Perrig et al. [19] can achieve with the help of specially trusted auction hardware.

Another approach comes from failure handling in distributed algorithms, where running nodes are classified into two basic types: correct and faulty. One goal of a distributed algorithm is to identify and ignore faulty behavior. In this model, a rational node that deviates from a protocol is considered faulty. However, ignoring rational nodes is not practical in systems like Gnutella, where in one study 70% of nodes seemed to free ride, thereby acting rationally [1]. It seems suboptimal to use traditional distributed systems techniques that detect and ignore faulty nodes. Doing so creates other problems as well such as network vulnerability, an idea explored in Adar & Huberman [1].

None of these approaches seems optimal, and this leads one naturally to consider designing networks with self-interest in mind. Mechanism design can help here by incentivizing rational nodes to perform as a network designer might intend.

3 A Brief Introduction to Mechanism Design

This section gives a brief introduction to economic mechanism design. Classical mechanism design concepts are covered in more detail in introductory and more advanced game theory textbooks [5, 8].

The idea in mechanism design (MD) is to define the strategic situation, or "rules of the game", so that the system as a whole exhibits good behavior in equilibrium when self-interested nodes pursue self-interested strategies. Formally, a *mechanism* is a specification of possible player strategies and a mapping from the set of played strategies to outcomes. This paper's initial auction example is a mechanism implementation in action: players choose their bids and the auctioneer computes the outcome according to the rules of the mechanism. Mechanism design can be thought of as *inverse* game theory – where game theory reasons about how agents will play a game, MD reasons about how to design games that produce desired outcomes. For instance, in the auction problem, we seek a mechanism that will provide incentives that result in neighbors forwarding the bids and bid announcements, and in nodes implementing the appropriate auction rules.

MD assumes that the players feed their calculated strategies to a special obedient *center* that performs the mechanism calculation and declares the outcome. However, one problem with traditional MD is that many mechanisms are computationally infeasible. For instance, it can be hard for a player to calculate a best strategy. It can be even more difficult for a mechanism implementation to calculate the outcome.

An emerging field, borne out of the theoretical computer science and artificial intelligence communities, deals with *tractable* mechanism design. Algorithmic Mechanism Design (AMD) addresses the computational complexity of the mechanism infrastructure and attempts to construct mechanisms that produce desired outcomes while retaining computational feasibility [16]. Research in mechanism design has also focused on the design of mechanisms that reduce the computational complexity of agent participation (e.g. [18]).

Distributed Algorithmic Mechanism Design (DAMD) is an even newer construction [7]. Whereas AMD is concerned with a centralized implementation, DAMD assumes that a mechanism is carried out via a distributed computation. This more accurately models situations like the Internet and peer to peer networks, where agents and resources are distributed.

In these settings, using obedient centers might not be feasible for a number of reasons, including issues of trust, privacy, and complexity:

- For instance, a node offering to perform a mechanism computation might have a vested interest in the outcome and silently change the result. Distributing a mechanism over many nodes with vested interest is more realistic in real-world networks when the presence of obedient nodes cannot be guaranteed. If the presence of *some* obedient nodes can be assumed, or when tools outside of mechanism design like cryptography or redundancy [2, 21] are available, distributing a mechanism may allow a robustness to cheating that is not possible in a centralized mechanism.

- Furthermore, because of network topology, a node running in a network with a centralized mechanism may have to reveal its strategy to the center via other competing nodes. For instance, a point to point transmission between a node and the center may actually occur via many hops over strategizing nodes. In distributed mechanisms, this problem is made explicit. One way of addressing this problem is to make truthful revelation be an equilibrium strategy (e.g. [6]).

- Finally, a centralized mechanism implementation might be NP-hard. Distributed mechanisms can sometimes achieve better complexity results than centralized mechanisms, and may even be piggybacked on existing network protocols [6].

In all forms of mechanism design, the mechanism designer is aiming to create a *good mechanism*. Mechanisms can be good in different ways: they can be efficient, strategy proof, incentive compatible, budget balanced, etc. (For space considerations, we hope to pique your interest but defer to other good sources [18] for a complete discussion of these properties.) A famous example of a good mechanism (with a center) that you may have used is the second-price sealed-bid auction.[1]

A well designed mechanism will incentivize rational nodes into behaving according to a designer's wishes. A poorly designed mechanism will fail to exact the expected behavior; this paper's auction scenario is a trivial example.

A more subtle problem can occur if the mechanism designer does not build mechanisms with strategies that are simple to compute. For instance, it is reasonable to suppose that different nodes have different computational capabilities in a peer to peer network. If a node is constrained by computational or communication limits, it may fail to pick a strategy that maximizes its expected utility. In game theory, this node is known as a *bounded-rational* player.

Finally, mechanisms can be modeled as one-shot or repeated, and behavior that may not seem rational in the short term (a starving graduate student slaving over a paper submission, when sleep would yield a higher utility) is (hopefully) rational in the long term.

[1] If you have used Yahoo! Auctions or Ebay, you have seen this flavor of mechanism at work. For instance, if the current bid for an Enron Ethics Manual [22] is 10, and you tell the auctioneer you are willing to pay 15, your bid will be recorded as 11 (assuming a minimum bid increment of 1). If only one other bid for 13 is received before the auction closes, you win the auction and pay 14 (the second highest price + the minimum bid increment.) The second price sealed-bid auction (for which William Vickrey won the 1996 Economics Nobel Prize) has the nice property that the best (dominant) strategy a bidder can choose is to declare truthfully his/her value for owning the item. This mechanism also happens to be efficient, individual rational, and weakly budget balanced, further nice properties that make Vickrey auctions at least theoretically popular. See Sandholm [20] for a concise overview of various auction types and a discussion of why the interesting Vickrey auction is not used very often in real life.

4 Node Types in Peer to Peer Systems

Mechanism design should be viewed as a tool in the network designer's repertoire to help deal with rational nodes. When additional node types are present, additional tools are needed. Distributing the mechanism implementation creates additional challenges [7] that need to be addressed with outside techniques like redundancy or cryptography.

Feigenbaum & Shenker give an enumeration of node types in a network based on node intention [7]. Our listing differs slightly in order to stress how the actual mechanism affects the behavior of strategizing nodes, and to explore techniques for dealing with these strategizing node types.

First, we should mention two classes of non-strategizing node types described in the distributed systems literature:

Correct/Obedient nodes correctly follow a given protocol.

Faulty nodes have been classified according to their side effects and severity. Elements in this classification set include failstop (a node stops working), send/receive omission (a node drops messages), and Byzantine failure (a node can act arbitrarily) [12].

Economic mechanisms do not affect either correct or faulty nodes, since these node types do not strategize. Correct nodes need no special handling. Typical techniques for dealing with faulty nodes include using redundancy and cryptographic signing to detect and ignore broken nodes [12]. Second, we define two classes of nodes that can strategize about their behavior:

Rational nodes aim to maximize their expected utility from participation, given their beliefs about their environment, including the types of other nodes and the network topology. Rational nodes can exhibit behaviors normally attributed to the *adversarial nodes* found in cryptographic-protocol theory. Namely, rational nodes may use information learned from participation in a protocol to refine their own strategy. Rational nodes also can change or drop messages from other players, and can change or replace local algorithms as part of a utility maximizing strategy.

Irrational nodes behave strategically but do not follow a behavior modeled by the mechanism designer. They behave irrationally with respect to the mechanism. For example, these nodes might have utility functions that depend on more than just their own preferences. Anti-social nodes [3], for instance, prefer strategies that hurt other nodes even when it means reducing their own economic utility.[2] Alternatively, the bounded-rational nodes introduced in Section 3 might be unable to act rationally if the strategy calculation is too onerous.

Both of these node types are affected by the mechanism. The grand goal of the mechanism designer is to build a mechanism that enables strategizing nodes to act rationally, and incentivizes rational nodes to behave well. Irrational nodes can sometimes be brought into the rational node class by changing a poorly-designed mechanism. For instance, simplifying a mechanism might allow nodes that were bounded-rational to compute rational strategies in the new mechanism. Other types of irrational nodes might be made rational by designing repeated mechanisms instead of one-shot mechanisms.

5 Mechanism Design for Peer to Peer Networks

In this section, we highlight some open problems in DAMD that are especially apparent in peer to peer settings. In doing so, we also explore other techniques that can be used with MD to create systems that are proactive in dealing with the problems created by rational nodes.

Open Problem #1: What effect does network topology have on message passing in a centralized mechanism running on a peer to peer network? What about in a decentralized mechanism?

It seems that network topology can have both positive and negative effects on the ability to implement mechanisms with strategic nodes.

Monderer and Tennenholtz [14] examine the effects of topology on a centralized mechanism when nodes are not connected directly to a center, but instead must pass messages through other rational nodes. In this environment, it might be rational for infrastructure nodes to drop or change messages from their neighbors, as in this paper's initial auction example. They show that redundant

[2] In the real world, companies accept lower profits in attempts to drive out competition. While their behavior may seem irrational in the short term, in the grand scheme of things, their behavior makes sense.

message passing, weak cryptography, and a biconnected topology can be used to provide incentives for nodes to forward messages and implement a suggested protocol.

Yet, attempting to apply the same ideas to non-biconnected network leads to a negative result. These graphs can contain rational nodes at articulation points. The rational behavior of these nodes is enough to break the mechanism.

One way to circumvent this negative result when the network topology is known is to use a digital signing scheme. The node implementing the centralized mechanism can send signed messages to all participants and request signed return receipts. Because of the strong cryptography, if an articulation point drops or changes one of these messages, the bad behavior can be caught and the deviating node can be punished.

In the distributed setting, rational infrastructure nodes must pass messages *and* perform computation to determine the outcome of a mechanism. The problem is identified in Feigenbaum et al. [6] as "the need to reconcile the strategic model with the computational model." As an example, the authors give a strategyproof incentive compatible distributed mechanism for computing lowest-cost paths in an interdomain routing setting. But, the model assumes that nodes obediently compute their part of the distributed mechanism. This issue of implementing mechanisms is critical in peer to peer networks when infrastructure nodes might be required to participate in the mechanism even when they are not disinterested in the outcome.

Open Problem #2: What are the bounds on the guarantees that mechanism design can provide in a distributed setting, and what is the minimum set of helper technologies that must be employed in concert with DAMD ideas in distributed networks?

When a mechanism is to be computed by strategic agents, DAMD techniques may need to be augmented to ensure that agents perform computations correctly. One approach is to use cryptographic signing techniques [7] to detect cheating. However, a heavy-handed cryptographic approach may not be desirable. Are there other techniques that a designer can use? Another approach may be to use network redundancy with catch-and-punish techniques to ensure compliance [14, 21]. A third approach, mentioned earlier, is to assume some obediency in the network either in the form of specialized hardware [19] or dedicated designer-trusted nodes. (See Open Problem #3.) Reputation systems is an economic area that might be useful in heuristically strengthening mechanisms. Paralleling Monderer and Tennenholtz [14], we are interested in understanding when clever partitioning of a mechanism across nodes can be useful.

Are these alternate techniques valid on all topologies? If one imagines these alternate techniques as the foundation upon which MD should be built, how few helper technologies can one employ?

Open Problem #3: How can assumptions about the distribution (but not the identity) of various node strategy types help to create mechanisms with good properties?

Researchers might take advantage of the fact that some nodes in a peer to peer system do appear to be obedient [1]. Designers can use this idea when creating mechanisms that might need to rely on obedient nodes. For example, a minimum number of obedient nodes are sometimes required in auction protocols (e.g. [10]). Alternatively, one could use these obedient agents to check the behavior of other nodes if the system implements catch-and-punish schemes to enforce good behavior. The system designer can also consider injecting a limited number of obedient nodes into the system to make the mechanism design problem easier.

6 Related Work

The last section of Feigenbaum & Shenker [7] considers applications of DAMD, two of which are peer to peer systems and overlay networks. They pose several open questions, including how rational agents might affect network topology formation. In this paper, we are additionally concerned about how network topology affects mechanism implementation.

There have been some heuristic approaches to addressing these problems, including the introduction of a "barter economy" currency [13] to induce proper node behavior. Mechanism design ideas may provide a useful toolkit to analyze these barter economies.

There has been considerable work in auction protocols. A protocol by Brandt [2] uses costly cryptography to remove the need for either an auctioneer or any obedient nodes. This is actually a side effect of work that concentrates on privacy-preservation in auctions. The work assumes a totally connected physical graph, which probably is not realistic in most peer to peer settings.

Finally, there are several topical papers on "peer to peer auctions" that basically ignore notions of rationality [17]. Similarly, recent papers on economic models for resource scheduling in scientific Grid computing have not explored issues of rationality [4].

7 Conclusions

Rationality is a concern in peer to peer networks, where, in a very real sense, the users are the network. The central challenge in turning to ideas from economics is to provide incentives for nodes to follow protocols that provide the network as a whole with good system-wide performance. Mechanism design, while not a silver bullet to address all network implementation problems, is a good tool to use when designing robust systems.

Our future work in this area will explore the open problems presented in this paper in an attempt to minimize the amount of external tools used to successfully implement distributed mechanisms. To this end, our current focus is on how redundancy helps the mechanism designer [21]. We are exploring how to build a real peer to peer resource allocation system using mechanism design as inspiration for dealing with rational nodes.

8 Acknowledgements

We thank Joan Feigenbaum for her continuing conversations on DAMD and related topics. We also thank Danni Tang for her assistance while putting together this paper.

References

[1] Eytan Adar and Bernardo Huberman. Free Riding on Gnutella. *First Monday*, 5(10), October 2000.

[2] Felix Brandt. A Verifiable, Bidder-Resolved Auction Protocol. In *Proceedings of the 5th International Workshop on Deception, Fraud and Trust in Agent Societies*, pages 18–25, 2002.

[3] Felix Brandt and Gerhard Weiß. Antisocial Agents and Vickrey Auctions. In *Pre-proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, pages 120–132, 2001.

[4] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. Economic Models for Management of Resources in Peer-to-Peer and Grid Computing. In *Proceedings of the SPIE International Symposium on The Convergence of Information Technologies and Communications (ITCOM)*, August 2001.

[5] Prajit K. Dutta. *Strategies and Games*. The MIT Press, 1999.

[6] Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, and Scott Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *Proceedings of the 21st Symposium on Principles of Distributed Computing*, pages 173–182, New York, 2002. ACM Press.

[7] Joan Feigenbaum and Scott Shenker. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, New York, 2002. ACM Press.

[8] Drew Fudenberg and Jean Tirole. *Game Theory*. The MIT Press, 1991.

[9] Garrett Hardin. The Tragedy of the Commons. *Science*, 162:1243–1248, 1968. Alternate Location: `http://dieoff.com/page95.htm`.

[10] Michael Harkavy, J. D. Tygar, and Hiroaki Kikuchi. Electronic Auctions with Private Bids. In *3rd USENIX Workshop on Electronic Commerce*, pages 61–74, September 1998.

[11] Leander Kahney. Cheaters Bow to Peer Pressure, 2001. `http://www.wired.com/news/technology/0,1282,41838,00.html`.

[12] Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, 1996.

[13] Jim McCoy. Mojo Nation Responds. `http://www.openp2p.com/pub/a/p2p/2001/01/11/mojo.html`.

[14] Dov Monderer and Moshe Tennenholtz. Distributed Games: From Mechanisms to Protocols. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI)*, pages 32–37, 1999.

[15] Sharman Networks. Kazaa Guide: Supernode FAQ, 2003. `http://www.kazaa.com/us/help/faq/supernodes.htm`.

[16] Noam Nisan and Amir Ronen. Algorithmic Mechanism Design. In *Proceedings of the 31st ACM Symposium on Theory of Computing*, pages 129–140, 1999.

[17] E. Ogston and S. Vassiliadis. A Peer-to-Peer Agent Auction. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2002.

[18] David C. Parkes. *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency (Chapter 2)*. PhD thesis, Univesity of Pennsylvania, May 2001. `http://www.eecs.harvard.edu/~parkes/pubs/ch2.ps`.

[19] Adrian Perrig, Sean Smith, Dawn Song, and J. Doug Tygar. SAM: A Flexible and Secure Auction Architecture Using Trusted Hardware, 1991. Submitted Manuscript.

[20] Tuomas Sandholm. Limitations of the Vickrey Auction in Computational Multiagent Systems. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS)*. AAAI Press, 1996. Menlo Park, CA.

[21] Jeffrey Shneidman and David C. Parkes. Using Redundancy to Improve Robustness of Distributed Mechanism Implementations, 2003. Working Paper. Poster version to appear at ACM Conference on Electronic Commerce EC'03.

[22] Smithsonian to Enshrine Enron Ethics Manual. `http://ca.news.yahoo.com/020227/5/k6vd.html`.