Economic Behavior of Peer-to-Peer Storage Networks

Andrew C. Fuqua, Tsuen-Wan "Johnny" Ngan, and Dan S. Wallach
Department of Computer Science, Rice University
{andrewf,twngan,dwallach}@cs.rice.edu

Abstract

Peer-to-peer networks introduce a new area of interplay between computer science and economics. Designers of such systems must firmly understand the incentives, preferences, and decision space of participating agents in order to decide the policies and make the system function as well as possible. This paper models the economic behavior of agents in a peer-to-peer storage network. From the model, it becomes clear that agents have single-peaked preferences for a system-wide parameter that defines the reliability of the storage network. Consequently, the system designer may implement a mechanism to elicit opinions for this parameter (knowing that they will be truthfully revealed) and set the system-wide value to some socially optimal level, or agents with similar preferences may cluster together to form a p2p network closer to their preferences.

1 Introduction

Peer-to-peer (p2p) networks provide a new platform for distributed applications, allowing users to share their computational, storage, and networking resources with their peers to the benefit of every participant. Most p2p system designs focus on traditional computer science problems including scalability, load-balancing, fault-tolerance, and so forth. However, many systems tend to assume that all users in the system are running "official" software, whereas users have a clear self-interest in modifying their software if it allows them to consume the network's resources without contributing any of their own. P2p systems must be designed to take user incentives and rationalities into consideration [6, 10, 14]. Given rational user behavior, we can then study the overall economic behavior of such systems. How will these systems evolve over time? How should the parameters of the system, such as the degree of object replication, be chosen when different users have different ideas about these parameters' optimal values?

While such parameters could be declared by a central authority, the total utility of the system could possibly be increased if individual utility is taken into account. This paper addresses this problem by considering a game theoretic model of p2p storage networks. We consider the preferences, utility functions, and constraints of the agents in the model. This allows us to analyze the economic behavior of the agents and suggest policies for system administrators. Such a model is most relevant to p2p systems, such as distributed backup systems (e.g., Pastiche [2]), where storage, not network bandwidth, is the limiting resource.

The rest of this paper is organized as follows. Section 2 gives a background on p2p storage networks. Section 3 uses utility theory to analyze the economics of such networks, and Section 4 discusses how this effects the decisions of system administrators and participants. In Section 5, we present simulation results to support our assumptions in the analysis. Finally, we conclude in Section 7.

2 Background

This paper considers economic incentives that will apply in virtually any p2p storage system. However, in order to be concrete in our discussions, we will focus on PAST [12].

PAST is a storage system built on top of a structured overlay and can be viewed as a distributed hash table (DHT). Each stored item in PAST is given a 160 bit handle and replicas of an object are stored at the k live nodes whose nodeIds are the numerically closest to the object's handle. PAST maintains the invariant that the object is replicated on k nodes, regardless of node addition or failure. The handle is built from a cryptographically secure hash (e.g., SHA-1) applied to the data being stored. As such, the handle has sufficient information for the holder of the handle to verify that the actual document has not been modified in transit. PAST is implemented using Pastry [11], a p2p routing substrate that scales well to large numbers of nodes, providing efficient mechanisms to locate the nodes closest to a desired handle as well as gracefully handling when new nodes arrive and leave the system.

Pastry, like other structured p2p overlays, assumes that nodeIds are assigned randomly and uniformly from the 160-bit space of possible identifiers. Attackers who can choose nodeIds can compromise the integrity of Pastry or any other structured p2p overlay. Even when they cannot choose nodeIds, they may still be able to mount “Sybil” attacks if they can obtain a large number of legitimate nodeIds easily [3]. Such attacks can be prevented only by limiting the attacker’s ability to join the network multiple times. Castro et al. [1] consider several approaches to accomplish this, although the only robust approach they identify requires a trusted central authority (CA) to issue entrance permits. Aside from issuing such permits, the CA is otherwise uninvolved in the operation of the p2p network, limiting the damage that can be caused if the CA is offline. The CA is assumed to serve the common good and all members of the p2p network must fully trust the CA. As such, the CA can potentially be extended for other operations requiring a globally trusted authority.

Like traditional file systems, PAST performance degrades when the system is operating near its full capacity. The probability of successfully inserting a document on the system also decreases. However, unlike traditional file systems, users cannot simply purchase larger disks for their local computer to increase system capacity; they must somehow convince remote computers to reserve more storage space. P2p storage systems generally need a notion of *fairness* in that a node should only get as much remote storage as it contributes its own local storage for the use of others. Ngan et al. [8] consider architectures to meet that goal; requiring nodes to publish auditable records of their usage and allowing nodes to audit their peers’ records anonymously gives nodes incentives to report their usage truthfully. Building on those results, we consider the properties of the economic system that develop when “cheating” has been rendered technically infeasible.

H_i:	L_i	S_i	$N_i = k_i S_i$	$R_i = \lambda k_i S_i$

H_i:	total hard drive space of the agent
L_i:	agent’s private local space
S_i:	local copies of documents the agent stores in the network
N_i:	reciprocal space the agent uses for locally storing remote documents
k_i:	agent’s replication factor
R_i:	additional local space the agent is required to contribute
λ:	overhead rate

Figure 1: Hard drive space usage of an agent.

3 Model

We begin modeling an agent of a node *i* by partitioning its hard drive space as shown in Figure 1 to implement a “fairness” policy, as described in Section 2. Each agent may choose its own *replication factor* k_i, while the *overhead rate* λ is a system-wide constant. Our agent wishes to store S_i units of data in the network. In reciprocity, the agent must make available $N_i = k_i S_i$ units of space for the use of remote nodes plus an additional overhead $R_i = \lambda k_i S_i$. Thus, including L_i units of private, unshared data, the total disk space usage is

$$\begin{aligned} H_i &= L_i + S_i + N_i + R_i \\ &= L_i + (1 + k_i(\lambda + 1)) S_i \end{aligned} \tag{1}$$

The constant λ defines an important aspect of how the system will behave. Higher values of λ increase the efficiency of finding a node with free space to absorb storage requests, at the cost of lower effective capacity in the p2p storage network.

3.1 Agent preferences

Initially, we assume that all agents consider λ to be an exogenous variable. The agent then needs only to decide how much of its personal data should be archived on the network (and, thus, how much space it must make available for remote storage). Thus, an agent primarily cares about L_i, S_i (see Figure 1) and $p_i(\lambda)$: the probability of successfully storing a document. While an agent’s preferences might vary with changes in other values, λ is the only value that must be agreed by all agents. Therefore, we will focus on the importance of λ. We start by modeling the utility function of node i to be $U_i(L_i, S_i, p_i(\lambda))$. We expect U_i to be a three good Cobb-Douglas utility function because the utility-maximizing values of the arguments should all be non-zero.[1] For instance, an extremely large amount of completely unreliable remote space (i.e., $p_i(\lambda) = 0$) would be useless. In general, zeros for any argument to the utility function represent degenerate cases (e.g., when an agent is providing no space for remote storage) that are uninteresting in practice.

From equation (1), $L_i = H_i - c_i S_i$ where $c_i = 1 + k_i(\lambda + 1)$. We can now write the utility function as

$$\begin{aligned} & U_i(L_i, S_i, p_i(\lambda)) \\ &= L_i^\alpha S_i^\beta (p_i(\lambda))^\gamma \qquad \text{where} \quad \alpha + \beta + \gamma = 1 \\ &= (p_i(\lambda))^\gamma \left((H_i - c_i S_i)^\alpha S_i^\beta \right) \qquad (2) \end{aligned}$$

$$\begin{aligned} & \log\left((H_i - c_i S_i)^\alpha S_i^\beta \right) \\ &= \alpha \log(H_i - c_i S_i) + \beta \log S_i \qquad (3) \end{aligned}$$

Since $(p_i(\lambda))^\gamma$ is a constant with respect to the other arguments, maximizing equation (2) or (3) will also maximize U_i. The first order conditions for maximizing equation (3) imply that

$$\frac{\beta}{S_i} = \frac{\alpha c_i}{H_i - c_i S_i} \quad .$$

By rearranging the terms, we have

$$\frac{\beta}{S_i} = \frac{(\alpha + \beta) c_i}{H_i} \quad ,$$

and thus

$$S_i = \frac{\beta H_i}{c_i(\alpha + \beta)} \qquad \text{and} \qquad L_i = \frac{\alpha H_i}{\alpha + \beta} \quad . \qquad (4)$$

We can now express an agent's preferences through an indirect utility function $v_i(\lambda)$, which gives the highest utility available to an agent for a given λ. By substituting (4) into (2), we obtain $v_i(\lambda) =$

$$(p_i(\lambda))^\gamma \left(\frac{\alpha H_i}{\alpha + \beta} \right)^\alpha \left(\frac{\beta H_i}{\alpha + \beta} \right)^\beta \left(\frac{1}{1 + k_i(\lambda + 1)} \right)^\beta .$$

[1]For background reading in economics, consult the utility theory or consumer behavior section of a microeconomics textbook such as Varian [15, Ch. 7].

All terms without λ are constant and can thus be collectively represented by m.

$$v_i(\lambda) = \frac{(p_i(\lambda))^\gamma}{(1 + k_i(\lambda + 1))^\beta} \cdot m \qquad (5)$$

Before further analyzing the properties of v_i we will model the reliability of the system with the function $p_i(\lambda)$.

3.2 Reliability function: $p_i(\lambda)$

Recall that $R_i = \lambda k_i S_i$ and the space R_i provides a buffer which helps maintain free space that is approximately uniformly distributed throughout the network. It is clear that for a constant S_i, a larger buffer (or equivalently a larger λ) lowers the failure rate when storing objects in a p2p storage network. Section 5 shows that the distributions of the reliability function can be approximated by

$$p_i(\lambda) = 1 - e^{-t\lambda}$$

for some fixed parameter t, where t depends negatively on the average size of the files stored. As the amount of free space in the system decreases, larger files will become more difficult to store than smaller files. Conversely, as λ increases, the probability of success will increase, but with diminishing returns. Other variables will affect the general reliability of the system, including the replication factor k_i and how many times a failed storage request might be retried before the system aborts the request. As a general rule, additional reliability can always be achieved at a cost of additional storage and/or communication overhead.

3.3 Properties of the indirect utility function

Now that we have a model for $p_i(\lambda)$, we can examine the properties of $v_i(\lambda)$ as in equation (5). We wish to show that preferences with respect to λ are single-peaked. That is, $v_i(\lambda) = (1 - e^{-t\lambda})^\gamma / (1 + k_i(\lambda + 1))^\beta$ has a single maximum.

In Figure 2 we plot $v_i(\lambda)$ against λ with different parameters. It provides only visual evidence of the single-peakedness of $v_i(\lambda)$. A formal proof is shown in Appendix A.

Figure 2 was plotted using varied parameter values, but for most reasonable values, the peaks of v_i occur for $\lambda \in [0.3, 1.5]$. When γ was given an extraordinarily high value, the peak of $v_i(\lambda)$ shifted well to

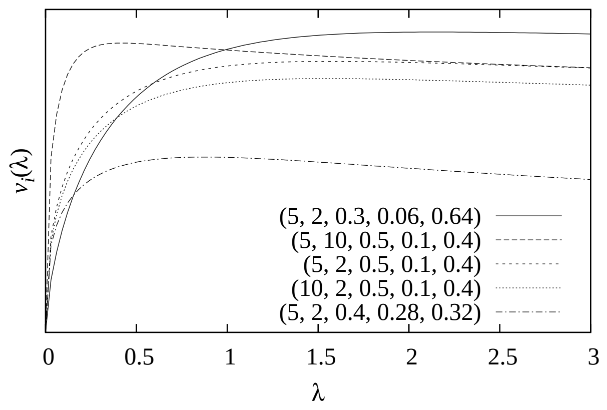

Figure 2: The single-peakedness of $v_i(\lambda)$ with different values of $(k_i, t, \alpha, \beta, \gamma)$.

the right. Since γ is the weight in the utility function for the probability of successfully storing a file, it follows that the utility-maximizing λ for nodes desiring high remote storage reliability would be larger than the λ for nodes that primarily value having more space available for local storage. We also observe that the indirect utility function is always strictly concave up to the peak, which means that there is decreasing marginal utility of λ.

4 Implications

In this section, we use our model to reason about rational behaviors for both agents and administrators of p2p storage systems.

4.1 Agent participation

Assuming that several storage networks exist, each with possibly differing λ_s values, agents will join the storage network whose parameters best suit the agent's own preferences. It is unlikely that an agent will find a λ_s which exactly equals its own λ^*, but with several storage systems to choose from, agents can evaluate and rationally choose to participate in the system closest to their preferences. (The alternative, of course, is to refuse to participate.)

As a corollary to the above observation, agents with similar preferences for λ will tend to band together. By clustering in this fashion, agents contribute their resources to form a system with the desired level of reliability and, in effect, create a public good. For example, an agent with a preference for low overhead (and reliability) would not join a high reliability network because the agent would rather allocate its disk space for its private use rather than to the extra reserve space (i.e., reliability) mandated by the higher λ. In other words, disutility is created by joining a non-optimal network. If the disutility is large enough the agent will refuse to participate at all, thus creating a market for another storage system that better suits the needs of that agent.

4.2 Administration

P2p systems are fundamentally designed to limit or entirely remove the role of centralized administration. Regardless, the presence of such administration can help provide a rendezvous point for new agents to determine which of many existing storage networks best suit the agent's preferences.

We also argue that a central administrator can conduct surveys of agent preferences, allowing for one or more networks to be defined to best match the expressed preferences of individual agents. If only a single system is to be established, economic voting principles suggest that the median of the agents' revealed preference variables should be chosen by the administrator, as this implies that no majority wants to increase or decrease the chosen value [15, Section 23.6]. Agents will truthfully reveal their preferences if they know the administrator employs this policy [16]. If the administrator finds that establishing clusters would benefit participants, it can partition the voting agents roughly according to their preferences and choose the median of the revealed preferences of the agents in each partition. Truthful revelation increases the likelihood that the administrator will define a storage system closer to that agent's optimal preferences, whereas lying means the agent might find itself assigned to a storage system that actually increases its disutility.

Of course, once agents join a system and begin exchanging data with one another, a mechanism such as the auditing described by Ngan et al. [8] becomes necessary to guarantee that no agent is free riding. The central administrator does not need to be involved in this auditing process, except perhaps acting as a "court" to which a node might bring evidence of another node's free riding behavior.

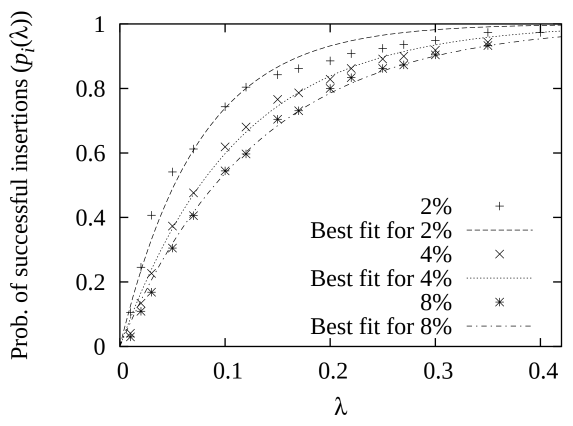

Figure 3: Probability of successful object insertions with different file sizes and varying λ values. The curves are of the form $1-e^{-t\lambda}$, fitting the data points by varying t. The percentage indicates the size of the document to be inserted as a fraction of the average storage capacity one agent contributes.

5 Reliability vs. overhead

In section 3, we modeled storage reliability $p_i(\lambda)$, as a function of the overhead rate λ. This section presents simulation results to determine the exact shape of $p_i(\lambda)$.

We constructed a PAST system with 10,000 nodes, each contributing storage space chosen from a truncated normal distribution from 2 to 200GB, with an average of 50GB. First, each node stores as many document as their quota permits. Then we attempt to store additional documents into the overlay network and record the probability of an eventual storage success. The result, as well as the best fit curves of the form $1-e^{-t\lambda}$, are shown in Figure 3. As expected, the figure shows that $p_i(\lambda)$ increases with smaller file sizes and higher λ. It also shows that $1-e^{-t\lambda}$ is a close approximation of $p_i(\lambda)$ measured by our simulations.

6 Related work

The field of Mechanism Design (MD) has existed formally for thirty years. It's goal to design the rules of interaction between agents so that their selfish, or self-interested, behavior produces some outcome deemed desirable by the designer. Classically, MD has been applied to auction theory, among other economic systems. More recently, as the view of computers as agents has become more prevalent, MD has also been applied in computational settings [16].

Nisan and Ronen [9] applied MD to solve some problems that might arise from agents manipulating algorithms to serve their own interest. Distributed Algorithmic Mechanism Design [6] applies MD specifically in a distributed setting and has as goals both computational tractability and incentive compatibility. It has been used to solve network problems related to multicast transmissions [5], efficient routing [4], and most recently p2p systems [14].

We described a voting process where agents reach agreement on parameters for their shared system. The game-theoretic aspect of voting is an active research area for both economics and artificial intelligence. Voting and decision-making of distributed agents is discussed in Sandholm [13].

Golle et al. [7] modeled centralized p2p systems with small incremental payments between agents. They proposed several payment mechanisms and analyzed how various user strategies reach equilibrium within a game theoretic model.

7 Conclusion

This paper presents an economic model of the resources and preferences of agents in p2p storage networks. By analyzing the model, specifically the indirect utility function, we observe that an agent has a single-peaked preference for the storage overhead rate λ. This implies that agents with similarly optimal λ values will have an incentive to cluster together and to reveal their preferences to a centralized administrator who can orchestrate this clustering. We expect this clustering will also work for other system parameters, including the degree of object replication.

Acknowledgments

We would like to thank Hervé Moulin and Terence Kelly for many helpful discussions.

This work is supported by NSF Grant CCR-9985332 and Texas ATP grant #03604-0053-2001 and by gifts from Microsoft and Schlumberger.

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer over-

lay networks. In *Proc. 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec. 2002.

[2] L. P. Cox and B. D. Noble. Pastiche: Making backup cheap and easy. In *Proc. 5th Symposium on Operating Systems Design and Implementation*, Boston, MA, Dec. 2002.

[3] J. R. Douceur. The Sybil attack. In *Proc. 1st Int'l Workshop on Peer-to-Peer Systems*, Cambridge, MA, Mar. 2002.

[4] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proc. 21st ACM Symposium on Principles of Distributed Computing*, New York, NY, 2002.

[5] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1), Aug. 2001.

[6] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proc. 6th Int'l Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1–13, Atlanta, GA, Sept. 2002.

[7] P. Golle, K. Leyton-Brown, I. Mironov, and M. Lillibridge. Incentives for sharing in peer-to-peer networks. In *Proc. 3rd ACM Conf. on Electronic Commerce*, Tampa, FL, Oct. 2001.

[8] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *Proc. 2nd Int'l Workshop on Peer-to-Peer Systems*, Berkeley, CA, Feb. 2003.

[9] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proc. 31st ACM Symposium on Theory of Computing*, Atlanta, GA, May 1999.

[10] C. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd ACM Symposium on Theory of Computing*, pages 1–5, Hersonissos, Crete, Greece, July 2001.

[11] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object address and routing for large-scale peer-to-peer systems. In *Proc. IFIP/ACM Int'l Conf. on Distributed Systems Platforms*, pages 329–350, Heidelberg, Germany, Nov. 2001.

[12] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. 18th ACM Symposium on Operating Systems Principles*, pages 188–201, Chateau Lake Louise, Banff, Canada, Oct. 2001.

[13] T. Sandholm. Distributed rational decision making. In G. Weiß, editor, *Multiagent Systems: A Modern Appraoch to Distributed Artificial Intelligence*, chapter 5. The MIT Press, 1999.

[14] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. In *Proc. 2nd Int'l Workshop on Peer-to-Peer Systems*, Berkeley, CA, Feb. 2003.

[15] H. R. Varian. *Microeconomic Analysis*. W.W. Norton & Company, New York, NY, 3rd edition, Mar. 1992.

[16] H. R. Varian. Mechanism design for computerized agents. In *Proc. First Usenix Workshop on Electronic Commerce*, New York, NY, 1995.

Appendix A

In this appendix we prove the following theorem on the single-peakedness of $v_i(\lambda)$.

Theorem. *For positive constants k_i, t, β, and γ, the function*

$$v_i(\lambda) = \frac{(1-e^{-t\lambda})^\gamma}{(1+k_i(\lambda+1))^\beta}$$

is single-peaked w.r.t. λ for $\lambda \geq 0$.

Proof. We prove by showing that for $\lambda \geq 0$, the sign of the first derivative of $v_i(\lambda)$ changes exactly once, and it is from positive to negative. For notational convenience, let $z = k_i + 1$. Then

$$\begin{aligned}
v_i(\lambda) &= \frac{(1-e^{-t\lambda})^\gamma}{(k_i\lambda+z)^\beta} \\
\frac{dv_i(\lambda)}{d\lambda} &= \frac{1}{(k_i\lambda+z)^{2\beta}}\big[(k_i\lambda+z)^\beta\gamma(1-e^{-t\lambda})^{\gamma-1}te^{-t\lambda} \\
&\qquad - (1-e^{-t\lambda})^\gamma\beta(k_i\lambda+z)^{\beta-1}k_i\big] \\
&= \frac{(1-e^{-t\lambda})^\gamma}{(k_i\lambda+z)^\beta}\left[\gamma te^{-t\lambda}(1-e^{-t\lambda})^{-1}\right. \\
&\qquad \left. - \beta k_i(k_i\lambda+z)^{-1}\right] \\
&= v_i(\lambda)\left[\gamma t(e^{t\lambda}-1)^{-1} - \beta k_i(k_i\lambda+z)^{-1}\right] \\
&= v_i(\lambda)(e^{t\lambda}-1)^{-1}(k_i\lambda+z)^{-1} \\
&\qquad \cdot \left[\gamma t(k_i\lambda+z) - \beta k_i(e^{t\lambda}-1)\right]
\end{aligned}$$

Let $\Phi(\lambda) = \gamma t(k_i\lambda+z) - \beta k_i(e^{t\lambda}-1)$. Since $v_i(\lambda)$, $(e^{t\lambda}-1)^{-1}$, and $(k_i\lambda+z)^{-1}$ are all positive, we only need to show that $\Phi(\lambda)$ changes its sign exactly once, and the sign changes from positive to negative. First we note that $\Phi(0) = \gamma tz > 0$. The derivative of $\Phi(\lambda)$ is

$$\Phi'(\lambda) = k_i t(\gamma - \beta e^{t\lambda}) \ .$$

Solving $\Phi'(\lambda) = 0$ for λ, we obtain $\lambda = \frac{1}{t}\ln(\frac{\gamma}{\beta})$. It is easy to verify that $\Phi'(\lambda) > 0$ for $\lambda < \frac{1}{t}\ln(\frac{\gamma}{\beta})$, and $\Phi'(\lambda) < 0$ for $\lambda > \frac{1}{t}\ln(\frac{\gamma}{\beta})$. Thus, $\Phi(\lambda)$ is positive at $\lambda = 0$, increasing until $\lambda = \frac{1}{t}\ln(\frac{\gamma}{\beta})$ (if $\gamma > \beta$), and then decreasing. In other words, $v_i(\lambda)$ is single-peaked for $\lambda \geq 0$. □