On designing incentives-compatible peer-to-peer systems

Tsuen-Wan “Johnny” Ngan Animesh Nandi Atul Singh Dan S. Wallach
Peter Druschel

Department of Computer Science, Rice University

1 Introduction

Peer-to-peer (p2p) systems allow participants to share their computational, storage, and networking resources to the benefit of every participant. This cooperative sharing gives participants access to an abundance of resources they could not afford individually. It also enables organic scaling as the system evolves, while requiring no dedicated infrastructure beyond network connectivity.

Most existing p2p system are designed to address issues such as scalability, load-balancing and fault-tolerance, but they assume that all participants in the system follow the protocols and observe the system’s fair use policies. However, in a system with open or loosely controlled membership, participants have a 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 participant incentives and rationalities into consideration [4, 8, 11] and provide appropriate mechanism to ensure participants are fairly sharing their resources. Ideally, we would like to design a system where nodes, acting selfishly, behave collectively to maximize the common welfare. When such a system has no centralized authority with total knowledge of the system making decisions, this becomes a distributed algorithmic mechanism design (DAMD) problem [4]. DAMD is a current area of study which combines computer science with incentive-compatible mechanism design in the economics literature. It provides a useful framework for considering p2p systems and many researchers are currently studying this approach [2, 3, 7, 11, 6].

In this paper, we sketch the design of two practical incentives-based mechanisms to ensure fair sharing of resources in p2p systems. One mechanism addresses storage as the resource of interest, while the other considers network bandwidth. The mechanisms are fully decentralized and require no consensus or synchronization among the participating nodes. Participants act autonomously, yet the mechanisms dictate that fairly contributing resources is in the best interest of each participant.

2 Fair sharing of storage

In this section, we sketch the design of an incentives-based mechanism useful in p2p applications where storage space (i.e., free disk space) is a limited commodity. Our mechanism is based on the idea that nodes are required to maintain and publish accounting records, and that other nodes can audit those records at any time. Of course, nodes have no inherent reason to publish their records accurately or to audit other nodes. Our mechanism creates natural incentives to perform these tasks accurately.

We assume the existence of a public key infrastructure and strong node identities; this can be achieved by having a trusted authority issue nodeId certificates, as described in Castro et al. [1]. The authority’s sole purpose is to issue certificates that bind a nodeId to a public key; it is not involved in other transactions.

Usage records: Every node maintains a *usage record*, digitally signed, which is available for any other node to read. The usage file has three sections:

- the *advertised capacity* of the storage this node is contributing to the system;
- a *local list* of (nodeId, handle) pairs, containing the identifiers and sizes of all objects that the node is storing locally on behalf of other nodes; and
- a *remote list* of handles of all the objects the system is storing on behalf of the node, with their sizes.

Together, the local and remote lists describe all the credits and debits to a node’s account. A node is allowed to store new objects into the network as long as its advertised capacity minus the sum of its remote list times a system-wide constant $\mu < 1$, is positive. Since the entries in local/remote lists have to be matched, it is impossible to create credits without making the usage record unbalanced. By increasing the advertised capacity, a node can store more objects in the system, but it also has make an equal amount of space available. By adding matched pairs in the local list of one node and the remote list of another, the credit is transferred from the latter node to the former.

When a node A wishes to store an object F_1 on another node B, B can fetch A’s usage record to verify that A is under quota. Then, two records are created: A adds F_1 to its remote list and B adds (A, F_1) to its local list. Of course, A might fabricate the contents of its usage record to convince B to improperly accept its objects.

Incentives: We must provide incentives for A to tell the truth. To game the system, A might normally attempt to either *inflate* its advertised capacity or *deflate* the sum of its remote list. If A were to increase its advertised capacity beyond the amount of disk it actually has, this might attract storage requests that A cannot honor, assuming the p2p storage system is operating at or near capacity; nodes have no incentive to provide any more storage to the network than is required of them by the network [5]. A might compensate by creating fraudulent entries in its local list, to claim the storage is being used. To prevent fraudulent entries in either list, we define an auditing procedure that B, or any other node, may perform on A.

Normal audit: If B detects that F_1 is missing from A's remote list, then B can feel free to delete the object[1]. After all, A is no longer "paying" for it. Because an audit could be gamed if A knew the identity of its auditor, anonymous communication is required, and can be accomplished by contacting the audited node through a random intermediate node, similar to Crowds [9]. So long as every node that has a relationship with A is auditing it at randomly chosen intervals, A cannot distinguish whether it is being audited by B or any other node with objects in its remote list. We refer to this process as a *normal audit*.

Random audit: Normal auditing, alone, does not provide a disincentive to inflation of the local list. For every entry in A's local list, there should exist an entry for that file in another node's remote list. To verify this, all nodes in the p2p overlay perform *random auditing*. With a lower frequency than the normal audits, each node chooses a node at random from the p2p overlay. The auditor fetches the usage record and verifies it against the nodes mentioned in that record's local list. Assuming all nodes perform these random audits on a regular schedule, every node will be audited, on a regular basis, with high probability.

Recall that usage files are digitally signed by their node. Once a cheating anchor has been discovered, its usage file is effectively a *signed confession* of its misbehavior! An auditor can present this confession to other interested nodes, e.g., nodes that store objects on behalf of the cheater, who can independently verify the cheater's guilt and delete his objects.

Our mechanism ensures that users have an incentive to fairly contribute resources to the system, since they risk having their objects deleted otherwise. Moreover, participants have an incentive to perform auditing to keep the system's storage utilization at or below μ, thus maintaining the system's performance.

Ensuring that data is stored: In any p2p storage systems, it is imperative to ensure that nodes are actually storing the objects they claim to store. We ensure this using a *challenge* mechanism. For each object a node is storing, it periodically picks a node that stores related objects (e.g., replicas or erasure coded fragments of the same source object), and notifies all holders of related objects that it is challenging that target. Then it randomly selects an extent of the object and a random key, and queries the target for a keyed hash of the extent. The target may retrieve the object from the holders of related objects, but any such request during a challenge would cause the challenger to be notified, and thus able to restart the challenge for another object.

3 Fair sharing of bandwidth

Next, we sketch a mechanism to enforce fair sharing of bandwidth in p2p systems. Unlike storage, bandwidth is a short-lived resource and there is no equivalent to punishing a cheater by deleting its stored objects. Punishment must instead take the form of degrading a cheater's service. Unlike the previous mechanism, our fair-share mechanism for bandwidth does not require a PKI or certified node identities.

Credit and debt: A node maintains two variables for each node with which it has a *relationship*: the number of objects (or bytes or some other globally-specified block size) sent to the node and the number of objects received from the node. The difference of these two numbers expresses the *debt* or *credit* that a node has with its peer. The total number of objects that a node has received from a peer measures the *confidence* that a node has in its peer.

Pairwise trade: The debt and confidence values can be used by good nodes to discriminate freeloaders from other good nodes, allowing them to refuse service to freeloaders. A simple policy is to set a *debt threshold*. Requests from a node are honored unless the debt exceeds the threshold. This debt threshold is increased dynamically as a function of the confidence value, giving more slack to peers that have performed well over time.

In general, a node has no incentive to serve an object to a node with which it has no prior relationship. Imagine a mature node (having lived in the system for a long time) A that has credit with many nodes. However, if A wants to read an object from a node Z, which does not happen to owe A anything, how can A leverage the credit it already has to obtain the object? We solve this problem by *transitive trading*.

Transitive trade: A transitive trade allows a node to take advantage of its earned credit to obtain objects from nodes which might otherwise refuse to serve it. It works by identifying a *debt-based path* from itself to a node

[1] In practice, B should give A a grace period as A might be facing a transient failure.

that has the desired object. In a debt-based path, each node in the path has credit with the next node, i.e., it has a relationship with, and is below the debt threshold of the next node. Locating such debt-based path can be achieved efficiently in a structured overlay network like Pastry [10], by looking up the desired object using debt as the proximity metric.

Once a debt-based path is identified, the following simple protocol ensure a secure exchange of data and credit along the trading chain. Assume that *Z* has an object that *A* wishes to download. The object is broken into a number of fixed-size blocks. *A* routes a message through the trading chain to *Z* requesting a specific block. *Z* transmits the block directly over the underlying network to *A*. *A* then transmits subsequent block requests along the same trading chain. Intermediate nodes can incrementally adjust their debt and credit variables when they forward these requests. If any party in the trade refuses to pass along the control traffic, then the data traffic will stop as well; the party dropping the request will get at most one "free" block.

Node bootstrapping: When a new node joins the system, it has no debts and no credits. Unless other nodes in the system behave altruistically, nobody will honor requests from the new node. Granting an initial debt threshold is a form of altruism in our mechanism, where nodes honor a few object requests from nodes with which they have no prior relationship. Moreover, it is assumed that new nodes contribute some content or are asked to store content on behalf of others. Requests for these objects then allows the new node to establish credit with other nodes. Once two peers have established a relationship, one can show that setting a debt threshold proportional to the square root of the confidence value limits the damage freeloaders can do while minimizing denied requests among well-behaved peers who experience a temporary asymmetry in their request patterns.

Relationship throttling: In a large system with many nodes, a freeloader could exploit altruism in the system by continuously forming new relationships, abandoning them as soon as they have reached the debt limit. Thus, it is necessary to limit the rate at which nodes can establish new relationships. Observe that mature nodes with sufficient credit can easily find debt-based paths to any other node. As such, they have no incentive to establish new relationships and they generally refuse them. On the other hand, new nodes *want* to acquire credit to redeem later; they have an incentive to establish new relationships, even at the risk of being exploited. This narrows the potential target for freeloaders to newly joining nodes.

Simulations show that as a result of our mechanism, freeloaders quickly experience significantly lower quality of service (in terms of the number of tries and the length of time it takes them to obtain a desired object) than well-behaved nodes. While we cannot completely eliminate freeloading by patient nodes, the mechanism creates a strong incentive for nodes to contribute resources in order to receive acceptable service.

4 Discussion

We have sketched two mechanism that provide nodes in a p2p system with natural incentives to share their storage and bandwidth resources. The mechanisms are quite general and can be applied to many p2p storage or content distribution system in which storage or bandwidth are contented resources. Our mechanisms are simple and require no distributed consensus or synchronization. The storage mechanism incurs no additional operations in the critical path of a p2p storage system; simulations show that the background traffic required for auditing is modest. The principal cost of the bandwidth mechanism is in discovering debt-based paths. Simulations show that a structured overlay network can discover such path at a small additional cost during the object lookup. A full experimental evaluation of our mechanisms is in progress. We conclude that incentives-based mechanisms are an interesting and effective means of enforcing policies in large-scale, decentralized systems.

References

[1] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Proc. OSDI'02*, Boston, MA, Dec. 2002.

[2] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[3] L. P. Cox and B. D. Noble. Samsara: Honor among thieves in peer-to-peer storage. In *Proc. SOSP'03*, Bolton Landing, NY, Oct. 2003.

[4] 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*, Atlanta, GA, Sept. 2002.

[5] A. C. Fuqua, T.-W. J. Ngan, and D. S. Wallach. Economic behavior of peer-to-peer storage networks. In *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, CA, June 2003.

[6] 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.

[7] T.-W. J. Ngan, D. S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In *Proc. IPTPS'03*, Berkeley, CA, Feb. 2003.

[8] C. Papadimitriou. Algorithms, games, and the internet. In *Proc. 33rd ACM STOC*, pages 1–5, Hersonissos, Crete, Greece, July 2001.

[9] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[10] 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.

[11] J. Shneidman and D. Parkes. Rationality and self-interest in peer to peer networks. In *Proc. IPTPS'03*, Berkeley, CA, Feb. 2003.