Selfish Caching in Distributed Systems: A Game-Theoretic Analysis Write-up

Odysseas Papapetrou, Tutor: Prof. Gerhard Weikum

December 30, 2004

Abstract

Original Document: Selfish Caching in Distributed Systems: A Game-Theoretic Analysis [6], from B. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou and J. Kubiatowicz

The authors analyze the caching problem in P2P systems when the peers behave selfishly, trying to maximize their own gain. They model the problem using a game-theoretic approach, and show that the game reaches a Nash Equilibrium. Following, they enhance the system with a payment methodology, which *tunes* the system toward a better solution.

1 Introduction

The nodes in distributed systems are usually characterized from a common goal, and follow protocols for achieving that goal. As such, the protocols may easily become optimized toward a better, more efficient, overall strategy. However, with the recent advances in decentralized P2P systems, existence of protocols in the application layer is not straight-forward. The peers may behave *selfishly*, that is, *deny obeying any protocols*, trying to maximize their own profit, and minimize their own cost. Namely this can introduce an important extra load in the system, usually referred as *price of anarchy*, and reduce the overall system performance.

The authors of this work study the problem of caching in a distributed P2P system, when the nodes behave selfishly. They model the above problem using a game-theoretic approach and prove that the system reaches a Nash Equilibrium. Later, they introduce a *payment model* which can *tune* the system to a better solution.

This introduction is followed by a short description of related work. We then give some basic notions on game theory. In section 4 we describe how our problem can be modeled and processed as a game. Then, in section 5 we enhance our problem with payments in order to *tune* to better solutions. Finally, we conclude with some discussion on this work and future plans.

2 Related Work

This work combines two distinct research areas: (a) peer to peer systems, and (b) caching systems. Both the subjects are widely studied in the past, and there have been many commercial applications of the research findings.

Peer-to-peer systems have recently attracted a considerable attention from the CS community. Their ability to robustly distribute and efficiently solve difficult problems by employing the computing power of many desktop PCs without central coordination, makes them very attractive for many scientific applications. The most prominent use of P2P systems is for information sharing [2, 1]. Another use of P2P systems is for building distributed file systems, such as OceanStore [7] and Pangaea [17].

One of the most difficult problems in P2P systems is optimization. In most of these systems, peers can behave selfishly, putting their own benefit over the social benefit, and over any community protocol. The most dominant practice for optimizing these systems is by the use of incentives. The whole idea of incentives is simple: *since the peers do not want to follow good behavior, find a way to make them realize the benefit.* For example, Golle et al. [10], studying on file-sharing P2P systems, propose, among others, to reward the participants for sharing-uploading data to the system with a micro-payment mechanism. For the same problem, in many modern P2P file-sharing systems such as eMule [1], the participants get rewarded for offering to the system (i.e. uploading or sharing files) by getting a better downloading bandwidth. Buragohain et al. in [5] propose another way of providing incentives to the peers to make them offer to the system. The proposal is based in game theory, and leaves out of the Nash Equilibrium any peers that do not offer enough to the system (these peers will not be satisfied on joining the system).

Caching on the other hand is one of the most common approaches in attacking performance-related problems in Internet. Furthermore, caching can be used for improving availability and reliability. However, most of the caching schemes are usually centralized. For example, NetCache [3] and Squid [4] are usually deployed in a dedicated machine (or a number of machines) and strictly serve the needs of an organization.

Having a distributed caching system is also studied in the past. Squirrel [12] uses a P2P technique to create a decentralized caching system. Squirrel enables the machines (players) to share their local web browser caches, in order to efficiently form a P2P caching system. The Squirrel protocol uses Pastry [16] for implementing P2P distributed hashing and efficient routing. However, Squirrel, as most of the other distributed caching systems, assumes perfect obedience of the peers to the protocol. As such, a well-designed protocol may optimize the system [11, 13, 15, 14, 9].

3 Game Theory

Game Theory offers a method of modeling and processing problems which include a number of independent components. Each component is modeled as a player in a multi-player game where the players have a specific set of goals to complete i.e. a number of resources to acquire. The set of the possible actions of each player is well-defined from the beginning of the game, and each action is usually accompanied by a cost. The players try to satisfy all their goals with the minimum possible cost (in some cases, the problem becomes a minimum-cost, maximum profit problem).

Game theory does not make any assumptions for the coordination of the players. More specifically, we assume that each player is completely free to follow his own strategy, without necessarily communicating, coordinating, or, at any other way, respecting any other player *(unless such rules are defined from the game setup)*. Being selfish, the players do not try for the best result for the community (i.e. choose a configuration that will minimize the social cost). At the course of the game the players can dynamically adopt their strategy, according to their opponents' strategies, always seeking their own minimal costs.

We will now describe some basic notions in game theory, which we will use while modeling the selfish caching problem:

Rational players: Each component in the multi-component system is called a player. By the word rational we emphasize that the players will only select an action if that action will be for their own profit i.e. minimize their costs (according to their current knowledge).

The possible actions: Each player has a number of possible actions to do, in order to succeed his goals. Each action is usually accompanied by a cost, which is known to the player beforehand.

The player's strategy: The strategy is a rule-like representation of the players behavior on the game i.e. if (buying-cost$\leq$renting-cost) then buy it else rent it. The strategy of each player defines his contribution to the game. The players dynamically adopt their strategies (according to other players' strategies), seeking the minimum personal cost for themselves.

The personal cost: The personal cost for each player is the sum of all the costs that occur from the player's strategy.

The social cost: Social cost is the sum of all the personal costs for all the players. Namely this is the cost we would like to minimize in our system.

The social optimum: The social optimum is the setup (may be more than one) that achieves the minimum possible social cost in our game (not necessarily an equilibrium and not necessarily reachable according to the rules of the game).

A Nash Equilibrium: A Nash Equilibrium is the state of the system where no player can benefit by changing his strategy, supposed that the other

players keep their strategies unchanged (all the players have reached a local minimum). Intuitively, this is the state of the game where every player can no longer minimize his cost, without first the other players changing their strategy ⇒ none of the players change their strategies any more.

The price of anarchy The price of anarchy is defined as the ratio of the worst Nash Equilibrium social cost (the maximum cost that can be a Nash Equilibrium for the system) to the cost of the social optimum

4 Modeling the Basic Game

Our problem includes a number of distributed peers (players) and a number of documents. Each peer needs to have access to some of the documents. Accessibility in the documents can be provided in two ways: (a) the peer can fetch the document every time it is needed from another peer that already cached the document, or (b) the peer can decide to cache the document itself, and keep it updated. Each peer must decide which of the two approaches will be cheaper for each document it needs, and implement its strategy in order to minimize its overall cost.

More formally, our game is defined as follows:

Players: Each peer is a player in the game.

Possible actions: The peers have only two possible actions. They can either fetch a document whenever it is needed or cache it locally.

The players' strategy: Each peer's strategy is defined as the documents which the peer decides to cache locally. Each peer decides for the strategy which, according to the expected requests and costs for each resource, minimizes the cost for the peer.

The strategy profile: A strategy profile $S = (S_1, S_2, ..., S_n)$ is the set of the decided strategies for each peer

The personal cost: The personal cost for each peer is the sum of the cost for caching the documents that are to be cached from the peer according to its strategy and the cost for fetching all the other documents whenever needed.

The social cost: Social cost is the sum of all the personal costs for all the peers. $(p_1, p_2, ..., p_n)$.

The configuration of a document: A configuration X of a document d is defined as the set of the peers that decide to cache d.

Our setup also includes the following rules:

1. Each peer is aware of the expected requests per resource (this can be done by history-log analysis).

2. Each peer is aware of the cost for acquiring each resource from a remote source.

3. Each peer is aware of the placement cost for each resource (that is, the cost for caching and maintaining a fresh copy of a resource).

4. The peers do not have a capacity limitation, that is, each peer can cache an unlimited number of resources.

We will now show how is the cost calculated for each peer. We will also transform our problem to a simpler problem, for matters of easier handling, and show how we can reach to a Nash Equilibrium.

4.1 The Cost Model

The whole game is actually motivated by the effort of each peer to reduce its personal cost, while still having all the needed resources. Thus, the most important part of the game, which actually defines the course of the game is the cost for each peer to acquire a resource. We have the following two cases:

- The peer caches the resource locally. In this case, the cost is called *placement cost*. The placement cost of peer i to cache resource j is noted by a_{ij} and defined as follows:

$$a_{ij} = k_{1i} + k_{2i} \frac{UpdateSize_j}{ObjectSize_j} \frac{1}{T} P_j \sum_k w_{kj}$$

where
k_{1i} is the installation cost
k_{2i} is the relative weight between the maintenance cost and the installation cost
$Updatesize$ is the size of an update
$ObjectSize$ is the size of an object
T is the update period
P_j is the number of writes over the number of reads
w_{kj} is the(expected) number of requests of peer k for document j

While the placement cost can be correctly calculated with the above formula, we will now simplify the problem by assuming that the placement cost for all the document and peer combinations is stable and equals to α. So, for the rest of this work, we will assume

$$a_{ij} = \alpha \ \forall \ i : peer, j : document$$

- The peer fetches the resource from the nearest remote cache. In this case, the cost for the peer i to fetch resource j is equal to the distance between the peer and the nearest remote cache that includes the resource, noted by $d_{il(i,j)}$, where $l(i,j)$ is the peer *nearest* to i that caches document j. The term nearest always refers to the fetching cost distance (the peer with minimum fetching cost distance from i).

According to this cost model, the solution of the game (the social optimum) can be mathematically found by Integer Programming.

Single-object game: Since peers do not have a maximum capacity, we can see the game of caching for each of the resources as a different game [8]. We will solve the caching problem for each of the resources, and then combine the solutions to reach to a solution for the multi-object game. More accurately, a Nash equilibrium for the multi-object game is the cross product of the equilibria for the single-object games. Thus, from now on, we will study the easier, single-object (basic) game. The cost of each peer in the single-object game is now simplified in the following:

$$C_i(S) = \alpha S_i + w_{ij} d_{il(i,j)}(1 - S_i)$$

4.2 Analysis

It should be clear by now that our game somehow "evolves". In each step of the game, each peer may decide to cache or de-cache a resource depended on the decisions-strategies of the other peers. We are interested to see if the game reaches a stable situation, where none of the peers is interested to change its strategy (cache or de-cache any resources) supposed that all other peers keep their strategies stable. This situation is called Nash Equilibrium.

Each game can have more than one Nash Equilibria. Each Nash Equilibrium denoted with (S_i^*, S_{-i}^*) specifies a configuration X (that is, a collection of peers which locally cache the resource). The set of all pure strategy Nash Equilibrium configurations is denoted with ε. We know that for all configurations that belong to ε, a peer caches a document iff the placement cost is less or equal to the cost occurred for fetching the document each time it is needed from the nearest peer that caches the document (or else, the peer could achieve a lower cost by altering its strategy and fetching the resource from a near cache). More formally, for any Nash Equilibrium configuration

$$X \in \varepsilon \Leftrightarrow \ \forall i \in N,\ \exists j \in X \ s.t.\ d_{ji} \leq \alpha \ and \ \forall j \in X,\ \neg \exists k \in X \ s.t.\ d_{kj} < \alpha$$

where N denotes the set of peers and X is a configuration.

We can easily show that there exist at least one Nash Equilibrium in the basic game. In order to produce one Nash Equilibrium we do the following:

1. `Ignore nodes with zero demand`
2. `Calculate` $\beta_x = \frac{\alpha}{w_{xj}} \forall x : peer$
3. `Order list of` β_x `ascending`
4. `Cache the document on the peer with lower` β_x `in the list`

5. Remove all the peers from the list that can get
   the object from the new cache for cost lower than α
6. Repeat from 4 until the list is empty

When the above algorithm finishes, all the peers are pleased, and the system stabilizes. This is because all the peers that did not cache can get the document from a cache for cost less than the placement cost (α) and all the peers that cached the document will have more personal cost when de-caching the document.

We now want to evaluate our Nash Equilibrium solutions, and see how close they are to the social optimum solution. For the evaluation, we will use the social cost definition, and check how much worse are our Nash Equilibria from the social optimum. The social cost of a strategy profile (the set of strategies for all the peers) for the single-object game is defined as the sum of the costs for each peer, that is:

$$\sum_{i=0}^{n-1} C_i(S)$$

where $C_i(S)$ is the cost incured by server i. The social optimum strategy profile is the strategy profile that minimizes the social cost (not necessarily a Nash Equilibrium) and the social optimum cost is the cost of that strategy profile.

A Nash Equilibrium does not necessarily suggest the configuration resulting to the social optimum. There are even cases where the social optimum is not a Nash Equilibrium of the basic game. For example, in the system shown in Figure 1, the social optimum is when peer 1 decides to cache the system (the social cost will be $\alpha + 5$). However, such a configuration cannot be a Nash Equilibrium, since peer 1 can fetch the resource for $\alpha - 2$ instead of caching it for α.

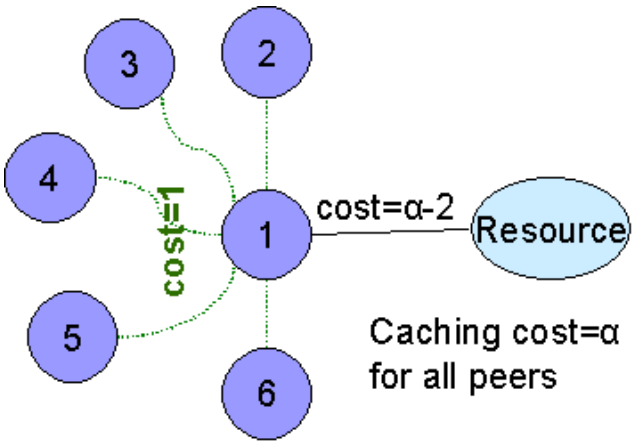

Figure 1: The Social Optimum is not always a Nash Equilibrium in the basic game

We can also show that the Price of Anarchy depends on the network topology. For example, in Figure 2 the PoA is 1, while in Figure 3, the PoA is $\frac{3(\alpha-1)}{2\alpha}$ (in our examples, for simplification purposes, we assume that all nodes have the same demand $w_{ij} = 1$ for our document). More specifically, the PoA is bounded by $O(n^{\frac{D}{D+1}})$, where D are the dimensions of the topology, and is asymptotically approaching $O(n)$ for high-dimensional network topologies.

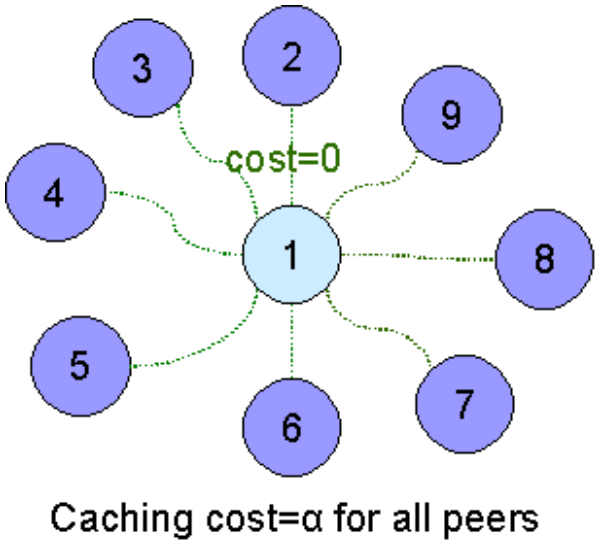

Figure 2: Star Topology: Peer 1 caches the resource. The Price of Anarchy is 1

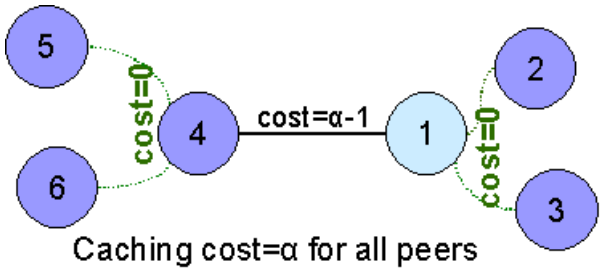

Figure 3: If peer 1 (or 2 or 3) caches the resource, none of the peers 4, 5, 6 decide to cache the resource in any Nash Equilibrium. They always get the resource from peer 1. The Price of Anarchy is $\frac{3(\alpha-1)}{2\alpha}$

5 Enhancing the Game with a Payment Model

We have seen that the Nash Equilibrium in the basic game does not necessarily succeed a low price of anarchy. In fact, the basic game can lead to very inefficient solutions. In order to try to reach to better solutions, we will now enhance the problem with a payment model. Again, the problem will be modeled using game theory. Then, we will evaluate the new solutions and compare them with the solutions of the basic game.

5.1 Modeling the payment-enhanced game

In the basic game, each peer needing a resource had only two options, a) to fetch the resource from a remote source, or b) to keep a local copy - cache of the resource. As such, the game did not encourage any other cooperation of the peers, which could potentially lead to better solutions.

The cause of inefficient solutions in the basic game was under-supply. Since the peers did not collaborate, they did not share the placement costs. Thus, the unshared placement cost was usually higher from the fetching cost for the majority of the peers. Now, in order to encourage cooperation we introduce a third option for the peers. The peers now can also *bid* to a neighboring peer, to cache the resource for them.

The bidding mechanism is simple. Each peer that needs access to a resource and does not have strong incentives to cache it locally (i.e. fetching is less expensive that caching) can promise some money to a neighboring peer (we call this action "bidding"), in order to persuade the second peer to cache it, and then get it from there. Then, when the second peer receives a sufficient amount of bids to cache a resource, it can decide to cache the resource and collect the bids (the bids on a peer are collected iff the peer decides to cache).

The payment-enhanced game adds three more decisions to the peers. The peers must first decide on their *threshold to cache*. The threshold to cache is the minimum value of the bids received from a peer, over of which, it is more profitable for the peer to cache the document locally (and collect the bids) rather than fetch it each time it is needed. In order to be profitable for the peer to cache instead of fetching the document (our peers still behave selfishly), each peer decides the threshold to cache, so that the following is true:

$$PlacementCost - ThresholdToCache \leq FetchingCost$$

Each peer also needs to decide which peer to bid on, and the maximum bid it is able to offer to each neighbor so that it can still have a profit compared to fetching the document from another cache or caching it locally. The best neighbor for bidding on would be the nearest neighbor (with the minimum distance from the peer). Furthermore, in order for the peer to have profit on bidding, the maximum bid would be:

$$dist(Peer, BestNeighbor) + MaximumBid \leq min(Fetching, PlacementCost)$$

Having added the bidding mechanism in the basic game, we need to redefine some of our notations. The strategy now for each peer (for the simplified single-document game) is a triple of the following form: $(v_i, b_i, t_i) \in (N, \Re_+, \Re_+)$, where
v_i is the peer to bid on for caching the document,
b_i is the value of the bid to v_i,
t_i is the threshold for caching the document.

The result of the payment-enhanced game must include not only the configuration for each document, but also the bids that are done. So, for each peer (for the simplified single-document game) the result of the game is now:$(I_i, v_i, b_i, R_i) \in (\{1|0\}, N, \Re_+, \Re_+)$, where
I_i is whether the peer decided to cache the document or not,
v_i is the peer to bid on for caching the document,
b_i is the value of the bid to v_i,
R_i is the bids the peer receives if it cached the document.

The personal cost for each peer now (always for the simplified single-document game) is also adjusted to include the biddings:

$$C_i(S) = \alpha I_i + w_{ij} d_{il(i,j)}(1 - I_i) + b_i I_{v_i} - R_i I_i$$

where α is the placement cost, I_i is whether peer i decided to cache the document or not, I_{v_i} is whether the peer v_i (the peer that i bids on to cache the document) cached the document or not, w_{ij} is the expected request for document j from peer i, $d_{il(i,j)}$ is the distance between the peer and the nearest remote cache that includes the resource, where $l(i, j)$ is the peer *nearest* to i that caches document j, b_i is the bid value made from peer i to peer v_i for caching the document, and finally, R_i is the sum of the bids collected from peer i in order to cache the document.

Since the bids received from all the peers and the bids made from all the peers are always zero-sum, the social cost in the payment game is not affected.

5.2 Analysis

All the Nash Equilibria of the basic game are Nash Equilibria of the payment game. This is true since the payment game can simulate the basic game in the following way:

```
select any equilibrium of the basic game
for each peer that caches the document in the basic game
  set caching threshold=0
for each peer that does not cache the document
  set caching threshold=a
for all peers
  set maximum bid=0
```

We can also show that the price of anarchy in the payment game is at least equal to the price of anarchy in the basic game. Since all the Nash Equilibria of the

basic game are also Equilibria in the payment game, this means that, the worst Nash Equilibrium in the basic game is also an Equilibrium in the payment game. Furthermore, the optimal solution in the basic game is the same to the optimal solution in the payment game (since the topology did not change). This results to a price of anarchy (PoA) for the payment-enhanced game to be at least equal to the price of anarchy in the basic game.

Furthermore, the payment-enhanced game can have worse Nash Equilibria than the basic game. Figure 4 presents an example of such an equilibrium. In the example, all the dashed lines represent connections which carry cost equal to 1 (distance between the two peers is 1). The cost of all the other connections is shown in the picture, and the cost for caching is α for all the peers. In the payment-enhanced game, peers 1 and 10 can bid on peers 8 and 9 a value of $\frac{\alpha}{4}$ in order to persuade them to cache the document. This will persuade both peer 8 and 9 to cache the document, but will lead to a Nash Equilibrium which is not a Nash Equilibrium in the basic game (peers 8 and 9 can not possibly cache the document at the same time in the basic game since they have distance less than α). However, the new Nash Equilibrium has worse social cost than any possible social cost in the basic game for this network topology.

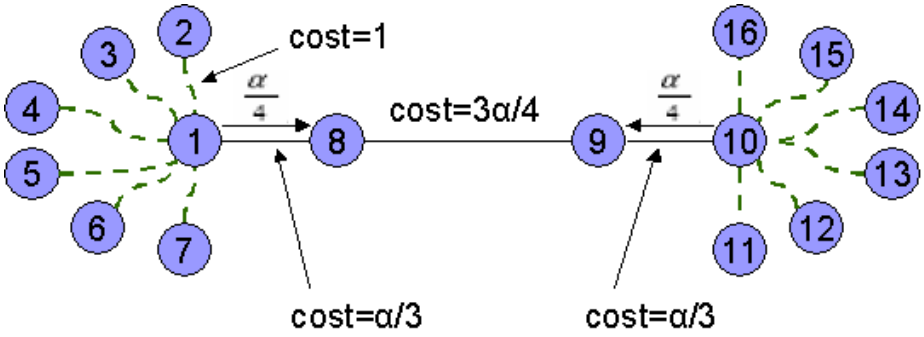

Figure 4: The payment-enhanced game may have worse Nash Equilibria than the basic game

However, we can show that we can always implement the social optimal configuration in the payment-enhanced game. This was not the case in the basic game i.e. the social optimal configuration was sometimes not a Nash Equilibrium in the basic game (Figure 1). More precisely, any social optimal configuration can be implemented now by distributing the difference of the placement cost to the fetching cost for each peer that caches the document in the social optimal as bids in the the peers that access the document from that peer. This will give the necessary incentive in the right peers to cache, since $PlacementCost - Bids \leq FetchingCost$. Furthermore, no other peer will cache since it can get the document from an existing cache for less cost than caching locally (or else the initial social optimal configuration would not be optimal).

Furthermore, the Optimistic Price of Anarchy (OPoA) in the payment-enhanced game is always 1. This is true since the social optimal configuration is always implementable in the payment-enhanced game. This makes the payment-enhanced approach promising for further work.

Simulations: Theory gives contradicting evaluations for the payment game. As we already presented, while the payment-enhanced game has always Optimistic PoA = 1 (which means that the social optimal is always implementable), we can sometimes get worse Nash Equilibria than the basic game.

In order to get a better insight of the payment-enhanced game, the authors perform large-scale simulations of both the games in varying network topologies, with varying number of peers. From the simulations, the payment-enhanced game seems to tune to better solutions compared to the basic game. More specifically, the PoA in the basic game is usually higher than the respective PoA in the payment-enhanced game for the same network topology. Furthermore, the payment-enhanced game gives incentives to cache, thus usually creates more replicas than the respective basic game (which usually reduces the PoA, since the PoA is due to under-supply).

6 Discussion

This work is an important step toward modeling and *tuning* non-cooperative P2P systems. With the recent advances in P2P systems and Internet, non-cooperative distributed systems where the peers behave selfishly are more realistic and more useful than the cooperative ones. Specifically non-cooperative caching is well-suited for P2P systems, and can actually be used in many distributed applications. This work does not only offer on selfish caching for P2P systems. The study, while focusing on the caching problem, presents an interesting approach for optimizing many other P2P applications.

However, the authors make many assumptions, and thus impose important constraints on the selfish caching problem in order to model and solve it as a game. More specifically, the following assumptions are made:

Server Congestion: Server Congestion is ignored in the problem. While the optimal cost-oriented solution in some setups can be to cache the documents in some central peers (i.e. in a star topology), this overloads these peers, and can cause severe problems. The overall system cost is importantly reduced in such cases. However, this is not taken under consideration in this work. This problem (alone) is investigated in [18].

Placement Cost: The authors assume that the placement cost for all the peers is equal. This importantly simplifies the game modeling but is non-realistic for the caching application. The game-theoretic approach can easily be used for studying the problem with varying placement cost.

Caching Capacity: The authors use peers with no capacity. While this assumption is essential for enabling the game-theoretic approach (otherwise,

we could not simplify our problem to a single-document caching problem), it can make the study not suitable for realistic systems.

Stable demand of all peers: We assume that all peers have the same demand. Enhancing the approach to enable varying demands over all peers is possible with game theory.

7 Conclusions and Future Work

The authors of this work used a game-theoretic approach to study the selfish caching problem. They initially model the selfish-caching problem as a non-cooperative game, where the peers can either cache or fetch the documents from another cache. Investigation on the basic game shows that the price of anarchy can be high. Then, they enhance the basic game with a payment option, which encourages cooperation, and gives incentives to cache. The enhanced game sometimes has even higher PoA than the basic game. However, the enhanced game always has Optimistic PoA equal to 1 (the social optimum can always be implemented).

The authors propose future enhancements on this work. First, they want to remove some of the assumptions made in the payment system (section 6). For example, they want to study the problem where the peers have varying placement costs and demands (we assumed that demand and placement costs are the same for all the peers), or when the peers have a maximum caching capacity. Furthermore, they want to take server congestion into account, which is completely ignored in the current system, and can lead to completely different solutions.

The authors also propose more study on the aggregation effect, when several peers are grouped and studied as one. They believe that the aggregation effect may reach to better Nash Equilibria. Moreover, the authors propose bigger simulations, with different network topologies, and want to study if forcing some distance constraints can reduce the PoA.

References

[1] eMule Peer-to-peer file sharing system, available at http://www.emule-project.net.

[2] Gnutella P2P File Sharing, available at http://www.gnutella.com/.

[3] NetCache Network Cache Appliance, available at http://www.netapp.com/.

[4] SQUID Web Proxy Cache, available at http://www.squid-cache.org/.

[5] Chiranjeeb Buragohain, Divyakant Agrawal, and Subhash Suri. A Game Theoretic Framework for Incentives in P2P Systems. In *3rd International Conference on Peer-to-Peer Computing*, 2003.

[6] B. Chun, K. Chaudhuri, H. Wee, M. Barreno, C. Papadimitriou, and J. Kubiatowicz. Selfish Caching in Distributed Systems: A Game-Theoretic Analysis. In *PODC*, 2004.

[7] J. Kubiatowicz et al. OceanStore: An Architecture for Global-scale Persistent Storage. In *ASPLOS*. ACM, 2000.

[8] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The Complexity of Pure Nash Equilibria. In *ACM STOC*, 2004.

[9] Michel X. Goemans and Martin Skutella. Cooperative facility location games. In *Symposium on Discrete Algorithms*, pages 76–85, 2000.

[10] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *ACM Conference on Electronic Commerce*, 2001.

[11] Steven Gribble, Alon Halevy, Zachary Ives, Maya Rodrig, and Dan Suciu. What Can Databases Do for Peer-to-Peer? In *WebDB Workshop on Databases and the Web*, 2001.

[12] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: A decentralized peer-to-peer web cache. In *21st ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC)*, 2002.

[13] B. Ko and D. Rubenstein. A Distributed Self-stabilizing Protocol for Placement of Replicated Resources in Emerging Networks. In *ICNP*. IEEE, 2003.

[14] V.N. Padmanabhan L. Qiu and G.M. Voelker. On the Placement of Web Server Replicas. In *INFOCOM*. IEEE, 2001.

[15] B. Li, M.J. Golin, G. F. Italiano, and X. Deng. On the Optimal Placement of Web Proxies in the Internet. In *INFOCOM*. IEEE, 1999.

[16] Antony Rowstron and Peter Druschel. Pastry: Scalable distributed object location and routing for large-scale peer-to-peer systems. In *International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[17] Y. Saito, C. Karamanolis, M. Karlsson, and M. Mahalingam. Taming Aggressive Replication in the Pangaea Wide Area File System. In *USENIX OSDI*, 2002.

[18] Subhash Suri, Csaba D. Toth, and Yunhong Zhouy. Uncoordinated Load Balancing and Congestion Games in P2P Systems. In *Third International Workshop on Peer-to-Peer Systems*, 2004.