Uncoordinated Load Balancing and Congestion Games in P2P Systems*

Subhash Suri[1], Csaba D. Tóth[1], and Yunhong Zhou[2]

[1] Department of Computer Science, University of California, Santa Barbara, CA 93106
{suri, toth}@cs.ucsb.edu
[2] Hewlett-Packard Laboratories, 1501 Page Mill Road, Palo Alto, CA 94304
yunhong.zhou@hp.com

Abstract. In P2P systems, users often have many choices of peers from whom to download their data. Each user cares primarily about its own response time, which depends on how many other users also choose that same peer. This interaction is best modeled as a game among self-interested agents, which we call *uncoordinated load balancing*. The players in this game are the rational and strategic users who are free to act in their own self-interest. We describe some of our recent work on this problem, and propose several new research directions, including analyzing Nash equilibria under general latency functions, a cost to switch servers, settings where user groups are dynamic, as well as the complexity of finding Nash solutions, and incentives for peers to be truthful in revealing their load.

1 Introduction

In Peer-to-peer (P2P) systems, data are often replicated to enable a high level of availability and fault tolerance. As a result, users typically have choice of many hosts from whom to download their data. Each user is selfish and strategic, and wants to minimize its response time. On the other hand, when serving data to multiple users, a host must share its limited bandwidth among those users. Therefore, the latency (response time) experienced by a user when downloading data from a host depends on *how many other users are connected to that host*. Different hosts may have different speeds. We assume that the response time is inversely proportional to the speed of the host, but *grows linearly* with the total number of users connected to a host. (All our results generalize to the case where the response time grows as the pth power of the load.)

Each user independently trying to maximize its utility is essentially engaged in a game with other users, which we call *uncoordinated load balancing*. Unlike traditional load balancing, however, users are not interested in optimizing the social welfare (e.g., *total* response time). Instead, each user has its own private objective. The stable outcomes of these interactions are the Nash equilibria—outcomes in which no user can improve its utility by switching unilaterally. In general, Nash equilibrium solutions can be much worse than the centralized outcomes, and Papadimitriou [8] has coined the term "price of anarchy" to denote the *worst-case ratio between a Nash outcome and the*

* Research of the first two authors was partially supported by NSF grants CCR-0049093 and IIS-0121562.

social optimum. We describe several problems and results concerning the price of anarchy for uncoordinated load balancing, and explore many associated algorithmic and structural questions. We begin by describing our model for the load balancing game.

2 Model and Results

An instance of the load balancing game is modeled as a bipartite graph G between a set U of n users (data requesting peers) and a set V of m hosts (data hosting peers). We will interchangeably use the terms users and clients, and hosts and servers. An edge (i, j) indicates that user i can obtain its data from node j. A *peer matching* is a (many to one) mapping $M : U \to V$ that assigns each user to a host.[3] That is, each user is matched to exactly one host, but a host can be matched to multiple users (or none).

Different hosts can have different speeds. Suppose a host j has speed σ_j and is matched to ℓ_j users, then we assume that the *response time*, or *latency*, to each user i connected to this host is $\lambda_i = f(\ell_j)/\sigma_j$, where $f()$ is an increasing function of the load ℓ_j. In general, the response time has two components, one host dependent and one network dependent. In our simplified model, we consider just the host-related latency, and treat network latency to be a constant. Under the *linear model*, the latency is simply ℓ_j/σ_j. (More generally, we consider latency functions of the form ℓ_j^{p-1}/σ_j, for $p \geq 1$.) The cost of a peer matching M is the total latency of all the users:

$$\text{cost}(M) = \sum_{i=1}^{n} \lambda_i.$$

A matching is a *Nash equilibrium* if no user can improve its latency by unilaterally switching to another host. Let M_{nash} be a Nash solution, and let M_{opt} be the (coordinated) social optimum. The *price of anarchy* $\rho(G)$ is the worst-case bound on the ratio between the costs of M_{nash} and M_{opt} for the problem instance G.

In this short paper, we give a brief overview of some of our key results, so that we can formulate the research problems we wish to propose; readers interested in technical details of these results should consult our papers [5, 13]. Our analysis shows that if all hosts have equal speed, then the price of anarchy is

$$\rho(G) \leq (1 + 2/\sqrt{3}) \approx 2.15$$

A more revealing way to express this result is the following: the price of anarchy is $\rho(G) \leq 1 + \frac{2m}{n}$, which tends to one as the ratio between the number of users to hosts grows. We also exhibit an instance G for which $\rho(G) \geq 2.001$, even with linear latency and equal speed for all the servers.

In general, when the hosts can have arbitrary speeds, the price of anarchy has an upper bound of 2.5. If the latency function grows as the L_p norm of the host loads, we show that the price of anarchy is bounded by $\frac{p}{\log p}(1 + o(1))$. The matching cost turns out to be related to the *sum of the squares of the server loads*. Thus, our techniques also lead to improved bounds for a natural online greedy scheme for load balancing [1].

[3] To avoid trivialities, we assume that for each client, there is at least one server that can provide the data.

3 Related Work

Several algorithms have been proposed recently for peer selection, such as “controlled update propagation” [12],“ adaptive peer selection” [2], among others. While these papers do acknowledge the fact that users selfishly want to choose peers with the best response time, they fail to model the game theoretic interaction among the users.

Our uncoordinated load balancing game belongs to the general class of *congestion games* introduced by Rosenthal [9] in game theory. In computer science, perhaps the best known congestion game is the network routing, studied by Koutsoupias and Papadimitriou [6], Czumaj and Vöcking [3], Roughgarden [10], and Roughgarden and Tardos [11], among others.

The load balancing game differs from these routing games in one important and fundamental respect: load balancing is *atomic*, while the routing games are *non-atomic*. In the latter, a user’s task (flow) can be split arbitrarily (across multiple paths). By contrast, in atomic games, one user is wholly assigned to a single host.[4] In routing games, each user is assumed to put only a *negligible* amount of traffic load on each network link. We do not require such an assumption—a user can own an arbitrarily large fraction of a host.

The more traditional *coordinated or centralized* load balancing has a long history in distributed systems. Our peer matching problem has connections with the load balancing variant in which we wish to minimize the L_2 norm of the server loads. The best known result on this problem is by Awerbuch et al. [1], who show that the online greedy algorithm achieves a constant factor *competitive ratio*. Our new analysis in [13] leads to an improved upper bound for the greedy’s performance.

4 Bounding the Price of Anarchy

It is well-known that Nash equilibria do not always optimize the social welfare, with the Prisoner’s Dilemma being one of the best-known examples. Our peer matching problem is no exception: an equilibrium matching does not necessarily have minimum cost. See Figure 1 for examples.

Fortunately, it can be shown that the price of anarchy (the worst-case ratio between a Nash solution and the social optimum) for the peer matching game is quite modest. We begin with some preliminary facts relating the optimal and the Nash matchings.

Upper Bounds

We have n clients, labeled $1, 2, \ldots, n$, and m servers, labeled $1, 2, \ldots, m$. The term *server load* denotes the *number* of clients assigned to that server in a given matching. We first observe that the cost of a matching (sum of clients’ latency) is related to the sum of the *squared* server loads. That is, if M is a matching, in which client i has latency λ_i and server j has load ℓ_j and speed σ_j, then

[4] Indeed, for the non-atomic version of the uncoordinated load balancing, we can show the price of anarchy is one; that is, Nash is always optimal.

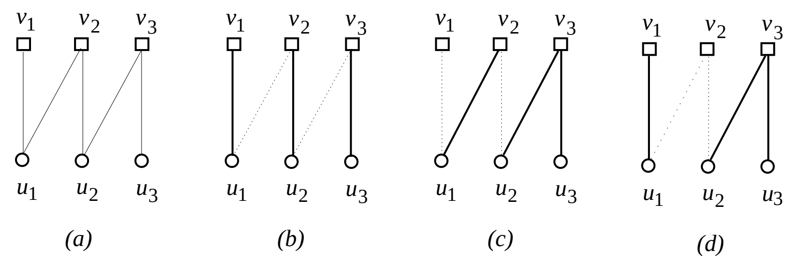

Fig. 1. (a) A client-server graph. (b) An optimal matching of cost 3. (c) A Nash but sub-optimal matching of cost 5. (d) A non-equilibrium matching.

$$\mathrm{cost}(M) \;=\; \sum_{i=1}^{n} \lambda_i \;=\; \sum_{j=1}^{m} \frac{\ell_j^2}{\sigma_j}. \tag{1}$$

This follows because each of the ℓ_j clients matched to server j suffers latency ℓ_j/σ_j. Given a problem instance, let M_{opt} denote an optimal peer matching, and let M_{nash} denote a Nash matching. Let O_j and N_j denote the set of clients assigned to server j in M_{opt} and M_{nash}, respectively. We use the shorthand notation $o_j = |O_j|$ and $n_j = |N_j|$ for the cardinalities of these sets. If a client is assigned to server j in the Nash but to server k in the optimal, then the following **Nash Condition** must hold, where σ_j and σ_k denote the speeds of these servers:

$$\frac{n_j}{\sigma_j} \;\le\; \frac{n_k+1}{\sigma_k}, \quad \text{if} \quad N_j \cap O_k \neq \emptyset.$$

This condition basically expresses the fact the client could not have improved its latency by switching unilaterally. This innocent looking condition is powerful enough to give us the following important **Nash Inequality**: *If M_{opt} is an optimal matching and M_{nash} is a Nash matching, then*

$$\sum_{j=1}^{m} \frac{n_j^2}{\sigma_j} \;\le\; \sum_{j=1}^{m} \frac{(n_j+1)o_j}{\sigma_j}.$$

This fundamental inequality allows us to prove the following upper bounds on the price of anarchy.

Theorem 1. *With linear latency functions and arbitrary server speeds, the price of anarchy is at most* 2.5. *If all servers have the same speed, then the price of anarchy is at most* $1 + 2/\sqrt{3} \approx 2.15$.

There are several specializations and generalizations of this main result. We mention a few. First, if all servers have an identical and linear latency function, then

$$\frac{\mathrm{cost}(M_{\mathrm{nash}})}{\mathrm{cost}(M_{\mathrm{opt}})} \le 1 + \frac{2m}{n}.$$

Thus, as the ratio (number of clients)/(number of servers) grows, *every Nash solution approaches the social optimum*. Second, if the server latency grows as the pth power of the server load, then we can measure the total latency using the L_p norm. In this case, the price of anarchy turns out to be $\frac{p}{\log p}(1 + o(1))$.

A Lower Bound

Figure 1(c) shows an example where the Nash solution is $5/3$ times the optimal. A more involved construction shows that the price of anarchy is at least 2.001, even with identical and linear latency functions.

5 Greedy Matching: A Myopic Strategy

Nash equilibrium is a compelling solution concept for distributed systems with self-interested players. Unfortunately, the concept is descriptive, not prescriptive: it does not suggest algorithms for computing an equilibrium and finding (distributed) algorithms for Nash equilibria remains a topic of current research.

We are therefore motivated to investigate the following simple *greedy* strategy: clients arrive in the system online in an arbitrary order; upon arrival, each client selects a permissible server with the smallest current latency, and *this selection is irrevocable*. The greedy is a myopic strategy—each client makes the best choice available to it at the moment, although future choices by *other clients* may make it regret its selection. While greedy does not generally lead to Nash solutions, it may well be a strategy most commonly used in practice. Thus, a natural question to ask is: *how bad is the greedy matching in the worst case?*

Surprisingly, even greedy performs well: with linear latency and arbitrary server speeds, the worst-case ratio $\mathrm{cost}(M_{\mathrm{greedy}})/\mathrm{cost}(M_{\mathrm{opt}})$ is less than $17/3 \approx 5.67$. If all servers have equal speed, then the ratio improves to $2 + \sqrt{5} \approx 4.24$. These bounds on the competitive ratio of the (coordinated) greedy solution are better than those known before [1].

6 Optimal, Nash, and Greedy: Some Structural Results

The price of anarchy focuses on the worst-case equilibrium solution. A more optimistic analysis could consider the *best case* Nash solution and ask how close to the social optimum can one get? In this section, we consider some questions like this and provide a few partial answers.

Theorem 2. *Assuming that every server has the same latency function $\lambda(x)$ and $x \cdot \lambda(x)$ is convex, then every optimal matching is also a Nash matching.*

Thus, for instance, if all servers have linear and identical latency functions, then the best-case Nash achieves social optimum. The following theorem, on the other hand, gives general conditions under which *no Nash solution is optimal*.

Theorem 3. *Assuming non-identical but linear latency functions, there are instances of the peer matching game where no Nash equilibrium matching is optimal. Assuming that every server has the same latency function $\lambda(x)$ and $x \cdot \lambda(x)$ is non-convex, there are instances where no Nash equilibrium matching is optimal.*

On the other hand, the following general result shows that under a broad class of latency functions, every Nash solution can be generated by the greedy strategy (with an appropriate order of client arrival).

Theorem 4. *If servers have monotone increasing (though not necessarily identical) latency functions, then every Nash matching can be generated by the greedy scheme.*

Finally, if the clients' job is allowed to be split across multiple servers, then it turns out that *every Nash solution is optimal*. That is, the price of anarchy equals 1. A non-atomic model of service is actually used in practice by the KaZaa system.

7 Research Directions

Our model and results suggest several natural and intriguing research problems, which should be of interest to distributed systems in general and peer to peer networks in particular. We mention some of the most promising such directions.

7.1 Computing Nash Matchings

The Nash equilibrium is a celebrated result in game theory. Unfortunately, there is no polynomial time algorithm known for computing a Nash solution in *general games*. The peer matching game, however, is not a general game. In fact, with identical and linear latency (or more generally, the latency functions with convex $x \cdot \lambda(x)$), even the *best case* Nash can be computed in polynomial time, by using graph matching ideas [5, 4]. However, these algorithms require that the entire client-server graph be known to all the clients. It will be interesting to explore algorithms that are distributed and require only *local information* (i.e. each client knows only about its permissible servers).

7.2 From Greedy to Nash

The matching determined by the online greedy scheme is within a constant factor of the social optimum, but it may not be stable—some of the users may want to switch to a different server. It may be interesting to investigate how to transform a greedy matching into a Nash. Suppose that users get one chance to switch their server in each round. How many rounds are needed before a stable solution is found?

7.3 The Cost of Switching Servers

In the greedy scheme, a client is not allowed to switch after its initial selection—in effect, the cost for switching servers is *infinite*. On the other hand, Nash solutions assume that a user can switch servers at zero cost. Thus, Nash and greedy can be viewed as two extremes in this cost spectrum. A natural question to ask is this: suppose a user incurs a cost α whenever he switches its server; what is the price of anarchy?

The server cost model may also have interesting algorithmic implications. With the infinite switching cost, we have a simple, distributed, online algorithm for computing the equilibrium matching (i.e. the greedy). With zero switching cost, no such (distributed) algorithm is known. Is it possible that adding switching cost improves the algorithmic complexity of finding a stable solution?

7.4 Coping with a Dynamic Client Set

We have assumed a static client group. In practice, new clients constantly arrive and old ones leave. Little is know about the loss of efficiency in such a *dynamic* setting—it involves *both the lack of information about future arrivals as well as the lack of coordination*. One basic problem is to investigate the price of anarchy where we compare centralized optimum to a solution in which clients are always at Nash—that is, whenever a new client arrives or an old one leaves, the remaining set recomputes a Nash matching. Next, it would be more realistic to consider this problem with a fixed switching cost. Still more realistic would be the model where the cost of switching depends on the *state of the client*—e.g., the cost may monotonically increase with time for each client, reflecting its unwillingness to switch as its job nears completion.

7.5 Effect of Server Speeds

Our upper bound for the price of anarchy is worse when servers can have arbitrary speeds (i.e. the ratio bound is 2.5 vs. 2.15 for equal speed servers). Intriguingly, we know of no lower bound that shows that price of anarchy should be worse with heterogeneous servers. It would be an interesting result to show that the price of anarchy with arbitrary speeds servers is never worse than with equal speed servers.

7.6 General Latency Functions

We have considered mostly linear or monomial latency functions. It would be worth investigating more general latency functions. In Section 6, we mentioned a few basic, isolated results in this direction, but much remains unknown. One very useful class to consider is the *piecewise linear* latency functions. We do not know of any non-trivial results for this class.

7.7 The Optimistic Nash

We have primarily considered the worst-case (pessimistic) Nash, except for the singular result of Theorem 2. The best-case Nash solutions are also compelling objects of study. With linear latency functions, perhaps the price of anarchy is modest enough to not worry about the best case. However, with higher order latency functions, the worst-case Nash may be too unattractive a solution, and it would be interesting to obtain bounds for the best-case Nash. It seems like a challenging problem.

7.8 Advertising Server Loads

The greedy matching, as perhaps any reasonable algorithm, requires the knowledge of current server loads. How should the load information be kept up-to-date and propagated to the clients? One simple idea is for each client to *probe* its permissible servers; the schemes in [2, 12] assume that the server either announces its load, or the client infers it through a test download. When there are many potential servers, this can be quite expensive. A possible direction to explore is to maintain either historical or current server load information in the system.

7.9 Truthfulness and Mechanism Design

Because P2P participants are assumed to be selfish, and there is no central authority, how can one ensure that players are truthful? In our context, what incentive does a server have to truthfully declare its load? It can lie both ways—if he benefits from serving many clients (e.g. through goodwill or ranking in the system), he may under-report, hoping to attract more clients; if he does not benefit, then he can over-report. Nisan and Ronen [7] have advocated using the VCG (Vickrey-Clarke-Grove) mechanism to compensate servers in a scheduling application. Ours is another natural setting where further research of this kind is needed.

References

1. A. Awerbuch, A. Azar, E. F. Grove, P. Krishnan, M. Y. Kao, and J. S. Vitter. Load balancing in the L_p norm. *Proc. 36th FOCS*, 1995, pp. 383–391.
2. D. Bernstein, Z. Feng, B. Levine, and S. Zilberstein. Adaptive peer selection. *Proc. 2nd IPTPS*, 2003, pp. 237–246.
3. A. Czumaj and B. Vöcking. Tight bounds for worst-case equilibria. *Proc. 13th SODA*, 2002, pp. 413–420.
4. N. Harvey, R. Ladner, L. Lovász, and T. Tamir. Semi-matchings for bipartite graphs and load balancing. *Proc. 8th WADS*, 2003, pp. 294–306.
5. A. Kothari, S. Suri, C. D. Tóth, and Y. Zhou. On a server selection game among selfish clients. Technical Report, Computer Science, UC Santa Barbara, 2004.
6. E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Proc. 16th STACS*, 1999, pp. 404–413.
7. N. Nisan and A. Ronen. Computationally feasible VCG mechanisms. *Proc. 2nd Conf. on EC*, 2000, pp. 242–252.
8. C. Papadimitriou. Algorithms, games, and the Internet. *Proc. 33rd STOC*, 2001, pp. 749–753.
9. R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *Int. J. of Game Theory* **2**:65–67, 1973.
10. T. Roughgarden. *Selfish Routing*. PhD thesis, Cornell University, 2002.
11. T. Roughgarden and E. Tardos. How Bad is Selfish Routing? *Journal of ACM* **49** (2):235–259, 2002.
12. M. Roussopoulos and M. Baker. CUP: Controlled Update Propagation in Peer-to-Peer networks. *USENIX Annual Technical Conference*, 2003, pp. 167–180.
13. S. Suri, C. D. Tóth, and Y. Zhou. Selfish load balancing and atomic congestion games. *Proc. 16th SPAA*, 2004, to appear.