

Games and Queues

Li Zhang Fang Wu* Bernardo A. Huberman

Information Dynamics Laboratory
HP Labs
Palo Alto, CA 94301, USA

Abstract

We consider scheduling in distributed systems from a game theoretic point view while taking into account queuing theory methodologies. In this approach no one knows the global state of the system while users try to maximize their own utility. Since the performance of such a blind scheduler is worse than optimal, it induces users to employ strategies that improve their own utilization of the system. One such strategy is that of restarting a request if it is not satisfied in a given time. Since we assume users as non-cooperative and selfish, the problem is that of studying the characteristics of the Nash equilibrium in a large distributed system with no omniscient controls. We study this problem through computer experiments and analytical approaches. We obtain exact solutions in situations delimited by two extremes: one in which users never restart an initial request, and another one in which the user's requests are restarted infinitely often. Users can switch between these two behaviors. When the system load is below a certain threshold, it is always better off to be impatient, and when the system load is higher than some other threshold, it is always better to be patient. Between these two thresholds there exists a homogeneous Nash equilibrium with non-trivial properties.

*Department of Applied Physics, Stanford University

1 Introduction

Scheduling is probably one of the most well-studied topics in Computer Science. As such it has been traditionally approached within the setting of an omniscient scheduler and passive users, with the scheduler having full knowledge of the state of the system and users placing requests and waiting passively for them to be processed. Unfortunately, both of those assumptions are often invalid in distributed computer systems. The Internet is probably the most prominent counter-example to these assumptions since its scheduling (or routing) is accomplished as a collective effort among numerous routers which only know their own state, and with users that are far from passive. Actually, most users play various strategies in order to maximize their own utility, such as restarting their browsers.

In this paper, we consider scheduling in distributed systems from a game theoretic point of view while taking into account queuing theory methodologies. Within this approach no one knows the global state of the system while users try to maximize their utility, which is measured in terms of getting their needs served as fast as possible. The system thus acts very much like a “blind” scheduler, i.e., one that has virtually no knowledge about the system’s state. Since one would expect the performance of such a blind scheduler to be worse than the optimal, it would induce users to employ strategies to improve their own utilization of the system. One common such strategy is that of restarting a request if it is not satisfied in a satisfactory interval, whose value which depends on the needs of the user.

While restart strategies are commonly used in practice, they were first studied systematically a few years ago [9] in the context of reducing the average and variance of the waiting time in Internet transactions. In that study the only tractable scenarios corresponded to situations where congestion is light, allowing for the design of restart strategies that minimize the response times and their variance. That methodology cannot easily be extended to more congested scenarios, thus motivating an approach that can yield predictions at all levels of congestion. In this paper we consider users that resort to a restart strategy in order to gain access to a particular resource. Since we assume users as non-cooperative and selfish, we cast the problem as that of studying the characteristic of the Nash equilibrium in a large distributed system with no omniscient controls within a game-theoretic sense.

In our model we consider a system with resources (machines) that satisfy user requests according to a first-in-first-out policy, and the requests are assigned to machines by a blind scheduler, i.e. one that assigns each request randomly to a machine. In such a model restarting amounts to cancelling and re-submitting a waiting request. The users can then decide how long they are willing to wait before restarting so as to minimize their total waiting time.

This mixed game theoretical queuing theory problem is first studied through computer experiments to reveal the possible regimes to be expected. We then proceed to obtain analytical solutions in situations delimited by two extremes: one in which *patient* users never restart an initial request, and another one in which the *impatient* user’s requests are restarted infinitely often. Users can switch between these two behaviors. Intuitively one might expect that when the system load is below a certain threshold, it will always be better off to be impatient, and when the system load is higher than some threshold, it will always be better off to be patient. As we show in this paper, those thresholds can be derived analytically, and have the property that between them there exists a homogeneous Nash equilibrium with non-trivial properties. Thus, in spite of its simplicity a restart strategy turns out to be very powerful when

dealing with a distributed system lacking an omniscient scheduler. This has particular appeal when designing a distributed system with a blind scheduler. As is presently the case in the Ethernet, there is no scheduler that determines who may use the publicly shared media. Instead, each user uses an exponential back-off protocol to decide when to send a message. Other examples include routing on the Internet and scheduling in highly distributed server farms.

The approach of treating users in a distributed system as players in a non-cooperative game has been used in various contexts, such as in resource allocation in distributed systems [3, 15] and in network pricing [14], and that approach has already produced a number of interesting results both analytically and experimentally [5, 11, 7, 13, 12, 8, 4] [2]. Most of the previous analysis is within the framework of congestion games [10]. In that framework, the waiting time on a machine only depends on the number of users on that machine and ignores the underlying scheduling mechanism, independently of how long a user has already waited. In our work we use a queueing model for the lower level of scheduling, which is more appropriate for practical applications such as network routing. By incorporating a queueing model, the analysis becomes much harder as it is often difficult to obtain analytical form of queueing delay. Nevertheless we are able to show both exact results in some limits and to complement them with computer simulations.

2 Modelling restart within blind scheduling

Consider a typical distributed system consisting of a set of users and machines, with users generating requests and machines processing them. When a request is assigned to a machine, it is placed in an internal first-in-first-out (FIFO) queue within the machine. An idle machine always takes the first request from its queue whenever the queue is not empty and processes that request. The time needed to process a request is S/V where S denotes the size of the request and V the speed of the machine. Each machine also prescribes a maximum queue size so that it only accepts requests when its queue size is under that limit. The response time of a request is defined to be the time between when a request is generated and when it has been processed. The performance of a system is evaluated primarily by the mean and the variance of the response time [9]. If an omniscient scheduler is available, i.e. if the scheduler knows the speed and the queue length of each machine, a simple greedy strategy works well.¹ In this paper, we are interested in systems where such information is unavailable. Conceptually such a system can be modelled as having a blind scheduler: a scheduler that has no knowledge about the machine speed and queue length, and thus assigns jobs in a random manner. We should emphasize that the blind scheduler notion is only conceptual and captures the behavior of a system when information about the global state or coordination is unavailable, such as routing on the Internet. We assume that the users may have at most one pending request at any time, and they may restart a pending request as described by the pseudo-code in Figure 1. A restart request is ignored if the request is being processed. Otherwise, the request is taken out from the queue it is in and re-assigned to a randomly chosen machine (and therefore placed at the end of the queue in that machine). We should note that in the above model there is no cost to restarts. This assumption might not be true in practice as it takes scheduler and machine's effort to locate, cancel, and reschedule a request. To avoid having the system overloaded by

¹Computing the optimum is NP-hard even for an omniscient scheduler.

```

while(true) {
    generate(R);
    submit(R);
    restart_times = 0;
    while(!processed(R)) {
        if((waiting_time > T)&&(restart_times < k)) {
            restart(R);
            request_times++;
            reset(waiting_time);
        }
    }
}

```

Figure 1: The restart algorithm. T is the restart time, and k the maximum allowed restarts for each request.

restart requests, it may discourage restart requests by adding delays to each request. On the other hand, if by using restarts the system can have better load balance, it may encourage the use of such restart strategy.

In our experiments we assume that each restart is free but that there is a limit on the number of restarts per request. This is in agreement to the intuition that while one can allow a small number of restarts, infinitely many are costly to both the user and the system. For our experiments, we assume that the users use a fixed restart time which they cannot adaptively or probabilistically change.

Whether or not to use a restart strategy largely depends on the system's load. Intuitively, when the system is less congested one should use a shorter restart time because the chance of reallocation to a lightly loaded machine is higher. But when the system load is high, using a restart strategy is probably not beneficial because users can lose their position in the queue and thus have to line up in the tail of some other queue which is likely to be full. In Section 4, we will make these points clearer. For our experiments, we focused on the interesting case when the load of the system is fair. We note that the analysis in [9] is related to the situation when the load of the network is light. In what follows, we first show that in the case of cooperative users the use of a restart strategy can achieve nearly optimal response time with a blind scheduler. Further, we study the more interesting case when the users are selfish and only care about the response time of their own requests.

Central to the theory of games with selfish players is the notion of a Nash equilibrium, the combination of strategies in which no user has incentive to change his own strategy unilaterally. We study the characteristics of the Nash equilibria in such a system with a focus on the homogeneous Nash equilibrium, in which all the users employ the same strategy. Since we forbid users from probabilistically choosing the restart time, we can view this as a game in which only pure strategies are adopted.

3 Experimental results

It is difficult to calculate analytically the mean response time as a function of the restart time. Several studies of similar models have had limited success with even

simpler models [1]. We study instead the problem first through computer experiments and then provide analytical analysis for a simplified model in the next section.

3.1 Setup

In our simulation we assume that each user generates requests according to a Poisson process. The size of requests is from a log-normal distribution. In both processes, we use the same parameters for all the users. We fix the total number (2000) of users and the maximum number ($k = 10$) of times a request may be restarted. The mean of the request time is 1500. The mean of the request size is 300 and the standard deviation is 100. We fix the number (50) of machines. The maximum queue size of each machine is 50. The base speed of each machine is fixed and uniformly distributed in the interval $[2.5, 9.5]$. To simulate different congestion condition, we can apply a multiplicative rate r to the machine speed, i.e. the real speed of a machine is r times its base speed. As explained before, we will focus on the cases when the load of the system is fair (when $r = 1.0$). With the above fixed parameters, we vary two parameters: the number N_r of users that use restart strategies and the restart time T for those users who use restart strategies. All of the simulations run fairly long with millions requests generated in total. For comparison, we run an omniscient greedy scheduling algorithm and obtain the mean response time of 614.28 and standard deviation of response time of 39.38. In the omniscient greedy algorithm, we calculate the response time of a request on each machine by the assumption that we know all the requests in each machine and the size of each request. We then assign the request to the machine with the minimum response time. While the greedy algorithm may not find the optimum value, its solution should be fairly close to the optimum. It also corresponds to the equilibrium when all the users are omniscient and can decide to which machine to assign their requests.

3.2 Cooperative restarts

In our first set of experiments we assumed that each user uses the same preset restart time. A special case is when none of the users uses restart, or in other words, when their restart time is infinity. Figure 2 shows the mean and the standard deviation of the response time as a function of the restart time. When no users uses restart strategy, the mean response time is about 1200, about twice as much as the mean response time of the omniscient greedy algorithm. The corresponding standard deviation is 1800, which is significantly higher. On the other hand, when users use the restart strategy with appropriate restart time, both the mean and the standard deviation of the response time become significantly lower. In particular, the mean response time is very close to that produced by the omniscient greedy algorithm. The performance of the system is fairly insensitive to the restart time — the minimal mean response time is achieved over a large interval $([20, 400])$ of restart time. The standard deviation of response times has a sharp drop for small restart time and reaches a minimum when $T \approx 80$ and remains fairly stable at 400 until T reaches 500. Therefore, we can conclude that when all users employ a restart strategy with the same restart time, it is easy to achieve the nearly optimal mean response time with relatively small standard deviation.

3.3 Non-Cooperative homogeneous restarts

As shown in the previous section, when all users employ a restart strategy with the same restart time, the best restart time is located in the interval $[80, 400]$ where both

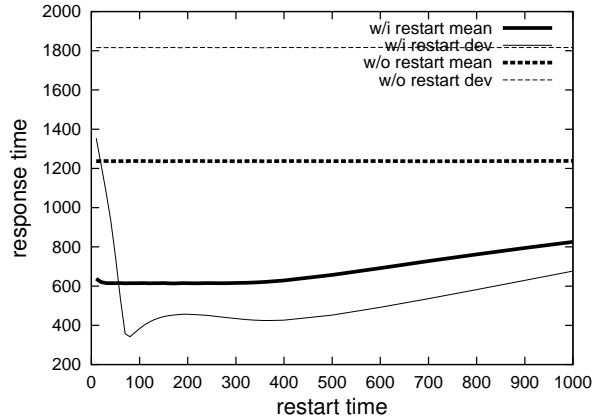


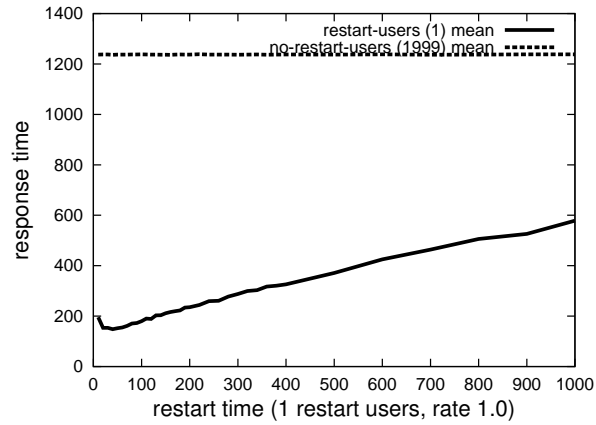
Figure 2: The mean and standard deviation of the response time for homogeneous cooperative users.

the mean and the standard deviation of the response time are low. But what if the users are non-cooperative, i.e. a user may defect from the restart time that everyone else uses? To answer that question, we first experimented with the extreme cases: first, we let one user not to restart and all the other users use the same restart time; and second, we let only one user restart and all others are not allowed to restart. The results are shown in Figure 3. In Figure 3.(1), when there is only one restart user, he has a much lower response time than those who do not restart, even significantly better than the omniscient greedy algorithm. On the other hand, when all the users restart, the non-restart user has a smaller mean response time unless the restart users use really small restart times (Figure 3.(2)).

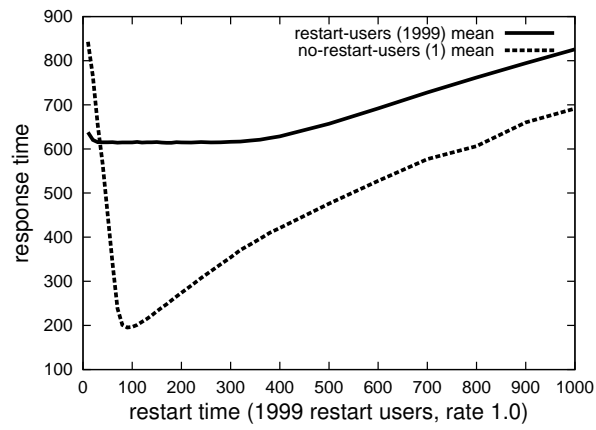
We further experimented with cases when the restart users use small restart times. We then varied one user's restart time and find that it is still beneficial for the user to defect. Figure 3.(3) shows the result. In Figure 3.(3), we plot the response time when all the users are restart users, and one user uses different restart time from the other users. The x -axis in Figure 3.(3) represents the restart time of the defect user. The solid curve corresponds to the mean response time of the homogeneous group of users. Since the group restart time is chosen from an interval such that the mean response time is small, the mean response time remains the same. The dashed curves correspond to the response time of the defect user while the group uses a different restart time.

From the figure, we can see that when all the users use the same restart time, one is always better off to be a little bit more patient, i.e. using a slightly longer restart time. However, as we showed earlier, when all the users are patient, it is always better off to restart. Therefore, there exists a critical value beyond which it is no longer true that being a bit more patient is better. In such cases, it is always better to restart really sooner than the others.

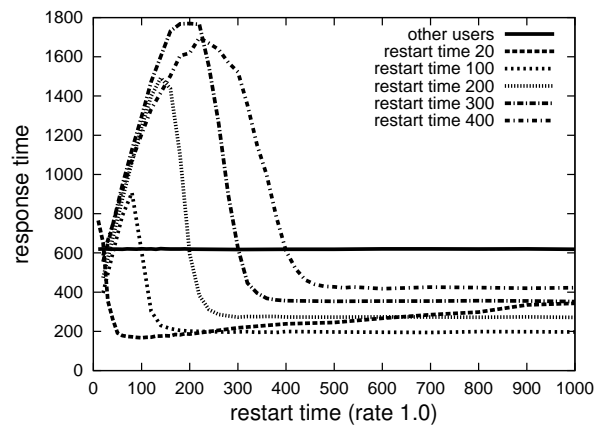
From these experiments, we conclude that it is always better to use a different restart time when all the other uses use the same restart time. In game theory terminology, this means that there does not exist a pure strategy, or homogeneous Nash equilibrium.



(1)



(2)



(3)

Figure 3: The mean and variance of the response time when there is one defect user.

3.4 Non-cooperative non-homogeneous restarts

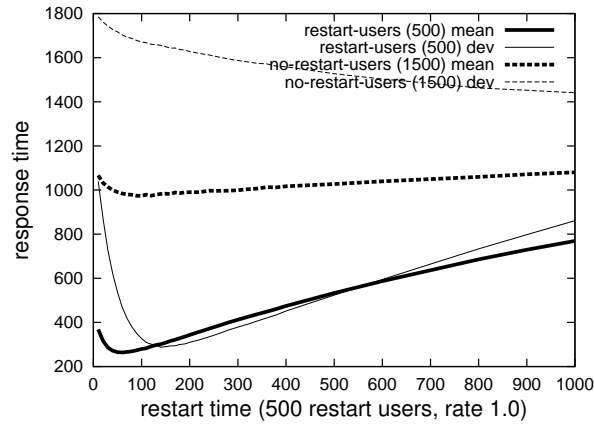
In the previous section we experimentally showed that there does not exist a pure strategy, homogeneous Nash equilibrium. This raises the question of whether there exists a pure strategy Nash equilibrium at all. We now experiment with the situation where there are two groups of homogeneous users and one group does not use restart strategy. Figure 4 shows three different settings, where the number of restart users is 500, 1000, and 1800, respectively. In all these cases, the interval $[100, 400]$ seems the most appropriate restart time for the restart users, as both the mean and the variance are low. From the figure, we can see that when N_r , the number of restart users, is 500, it is favorable for a non-restart user to switch to using restart. When $N_r = 1800$, on the other hand, it is beneficial for a restart user to switch to non-restart. When $N_r = 1000$, the gap between the restart users and non-restart users diminishes, which indicates the possibility of the existence of a Nash equilibrium in the neighborhood of this value.

4 Analysis

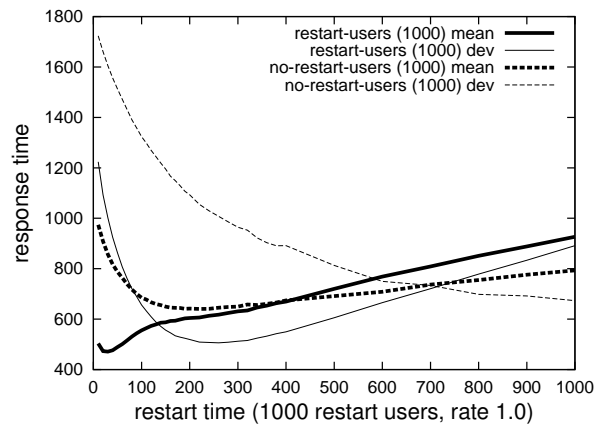
It is difficult to obtain analytically the mean response time corresponding to our experiment setup. We thus analyze a simpler model that has nontrivial solutions and show some characteristics of the restart strategy that emerge from its analysis.

4.1 A simplified model

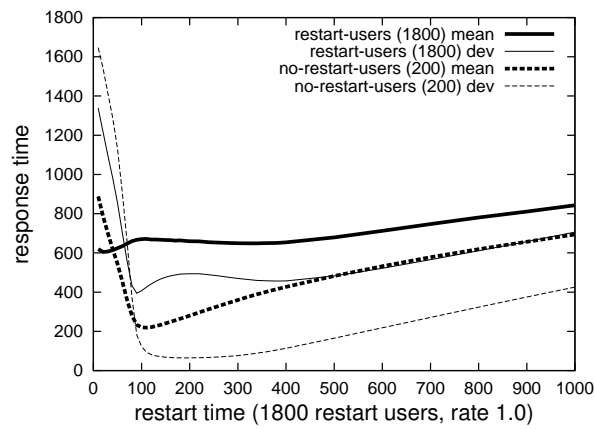
In this simplified model we assume that a restart request always restarts with 0 waiting time, i.e. it keeps restarting until the server it is randomly assigned to happens to be available. We call such requests impatient. An alternative way to view impatient requests is to create an additional queue, the “impatient” queue, to store such requests. An impatient request is served as long as one of the server is available. Another simplification is that we no longer require that each user generates only one request. Instead, we assume that users generate requests steadily at a constant rate. This simplification removes the feedback loop which adjusts the overall system input rate and makes the analysis much simpler. Formally, suppose that there are m identical $M/M/1$ queues, each having a service rate of $\mu = 1$, and there are n clients (players), each generating requests at the rate of λ/n . There are two kinds of requests, patient or impatient. A patient request is assigned randomly to a server and waits in the queue until being served. There is an additional “impatient” queue for impatient requests. An impatient request would wait in that queue and get served when one of the servers becomes available. This captures the fact that an impatient request has 0 waiting time and tries to catch any available server. Since we are only interested in the expected waiting time, the queuing principle of the impatient queue does not matter. We assume it is an FIFO queue. The immediate observation is that if all the users are impatient, then there is only a single impatient queue in the system. So the system behaves exactly as the $M/M/m$ queue, which is the most efficient way to use the system, or in the other words, it achieves the social optimum. However, what is interesting to us is the Nash equilibrium of the game where a user may decide each of its requests being patient or not. In our analysis, we consider mixed strategy, i.e. each user can decide according to a probability whether a request it generates should be patient or impatient. Let p_i be the probability that the requests generated by the client i are patient. Now, the question we would like to answer is whether there is a Nash equilibrium and, if there



(1) $N_r = 500$



(2) $N_r = 1000$



(3) $N_r = 1800$

Figure 4: Response time with two groups of homogeneous users.

is, what is the performance at the Nash equilibrium, compared to the social optimum. Let $\rho = \frac{\lambda}{m\mu} = \frac{\lambda}{m}$, and $p = \sum_{i=1}^n p_i/n$. Since all the users generate requests at the same rate, and a request is decided to be a patient request with probability p_i , we can view the system with total incoming rate λ , of which a fraction of p are patient requests, and $1-p$ are impatient requests. Let $T_1(p, \rho)$ and $T_2(p, \rho)$ denote, respectively, the expected waiting time for the patient and impatient requests of such a system. Note that T_1, T_2 are the mean waiting time but not the mean response time, which is the sum of the waiting time and the processing time. It suffices to consider waiting time because the expected processing time is the same for all the requests and all the machines in our model. In the following, we will first attempt to computing T_1 and T_2 and then analyze the property of the game.

4.2 Expected waiting time of patient requests

We are able to obtain a fairly simple formula for $T_1(p, \rho)$.

Lemma 4.1

$$T_1(p, \rho) = \frac{\rho}{1 - p\rho}.$$

Proof. Consider how a patient request can be delayed by an impatient request. An impatient request r can only grab a server when the server is free, and the patient requests that are going to be delayed due to the processing of r is exactly those requests that arrive in the busy period starting from the time when r is served. The mean of the total number of requests appearing in that busy period is $\frac{1}{1-p\rho}$ according to the $M/M/1$ queue result. Therefore, the number of patient requests that are delayed by r is $\frac{1}{1-p\rho} - 1 = \frac{p\rho}{1-p\rho}$ on average. Now consider what happens over a long time interval $[0, T]$. Since we assume $\lambda < 1$, each request experiences a delay with finite mean. Therefore, we can assume that all the requests are processed within time $[0, T + \delta T]$ where $\delta T/T \rightarrow 0$ when $T \rightarrow \infty$. By symmetry, we know that the impatient requests are evenly divided among the servers. Therefore, each server processes $\frac{(1-p)\lambda T}{m}$ impatient requests, and each causes a delay of $\frac{1}{\mu}$ for $\frac{p\rho}{1-p\rho}$ patient requests. Therefore, the total additional delay caused by impatient requests is:

$$\frac{(1-p)\lambda T}{m} \cdot \frac{p\rho}{1-p\rho} \cdot \frac{1}{\mu} = \frac{(1-p)p\rho^2 T}{1-p\rho}.$$

There are $p\lambda T/m$ patient requests arriving at any server, therefore the mean delay of each patient request is:

$$\begin{aligned} T_1(p, \rho) &= \frac{p\rho}{(1-p\rho)\mu} + \frac{(1-p)p\rho^2 T}{(1-p\rho)} \cdot \frac{m}{p\lambda T} \\ &= \frac{p\rho}{(1-p\rho)\mu} + \frac{(1-p)\rho}{(1-p\rho)\mu} \\ &= \frac{\rho}{(1-p\rho)\mu} = \frac{\rho}{1-p\rho}. \end{aligned} \tag{1}$$

□

From the above formula, we can see that $T_1(p, \rho)$ is increasing and convex in terms of both p and ρ .

4.3 Expected waiting time of impatient requests

Computing $T_2(p, \rho)$ is difficult for general p . We consider the extreme cases when $p = 0, 1$. When $p = 0$, i.e. when there are no patient requests, the system is exactly an $M/M/m$ queue. The following is a well known fact about $M/M/m$ queues [6].

Lemma 4.2

$$T_2(0, \rho) = \frac{(m\rho)^m}{m!m(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!}(1-\rho)^2 + \frac{(m\rho)^m}{m!}(1-\rho))}.$$

To calculate $T_2(1, \rho) = \lim_{p \rightarrow 1} T_2(p, \rho)$, we consider the situation when all but one requests are patient requests. When there is only one server, the impatient request has to wait only when the server is busy upon its arrival, and its waiting time is the time since its arrival to the time when the server becomes free. This is exactly the length of the busy period left of a randomly arriving request. When there are multiple servers, then the waiting time is the minimum busy period of those servers. When all the requests are patient, the queue state on each server is independent of the other servers. Therefore, we can apply the process for computing the expected value of the minimum of multiple random variables. Let $b(t)$ denote the p.d.f. for the busy period distribution of the $M/M/1$ queue with arrival rate ρ and service rate 1. It is known that

$$b(t) = \frac{1}{t\sqrt{\rho}} e^{-(1+\rho)t} I_1(2t\sqrt{\rho}),$$

where $I_1(x)$ is the Bessel function of the first kind,

$$I_1(x) = \sum_{k=0}^{\infty} \frac{(x/2)^{2k+1}}{k!(k+1)!}.$$

Now, we calculate the probability that the waiting time of an impatient request does not exceed t . Let X, X_1 denote the random variable of the waiting time of the impatient request for m and 1 servers, respectively. Then,

$$\text{Prob}[X \leq T] = 1 - \text{Prob}[X_1 > T]^m.$$

Let B be the mean of the busy period of an $M/M/1$ queue. Since $B = \frac{1}{1-\rho}$, we have that:

$$\text{Prob}[X_1 > T] = \frac{\rho}{B} \int_T^{\infty} (t-T)b(t)dt = \rho(1-\rho) \int_T^{\infty} (t-T)b(t)dt.$$

If we let $f_1(\cdot), f(\cdot)$ denote the p.d.f. of X_1 and X respectively, then

$$f_1(T) = \rho(1-\rho) \int_T^{\infty} b(t)dt,$$

and

$$\begin{aligned} f(T) &= m \text{Prob}[X_1 > t]^{m-1} \cdot f_1(t) \\ &= m\rho^m(1-\rho)^m \int_T^{\infty} b(t)dt \left(\int_T^{\infty} (t-T)b(t)dt \right)^{m-1} \end{aligned}$$

Therefore, $T_2(1, \rho)$ is given by the following formula.

$$T_2(1, \rho) = \int_0^{\infty} t \cdot f(t)dt.$$

Since $b(t)$, the p.d.f. of the busy period distribution of an $M/M/1$ queue, has a quite complicated form so the numerical solution is susceptible to errors. We show that for two servers, we can use the tool of Laplace transform to obtain a simpler form, which is an interesting result in its own right. We first calculate the distribution of X_1 . Suppose the impatient request sees N requests waiting in queue 1 when it enters the system, where N is a random variable. From [6] we know that N is geometrically distributed

$$\text{Prob}[N = n] = (1 - \rho)\rho^n, \quad n = 0, 1, \dots, \quad (2)$$

and has the generating function

$$f_N(x) = \frac{1 - \rho}{1 - \rho x}. \quad (3)$$

The time the request has to wait for queue 1 to become empty can thus be represented as a sum of busy period:

$$X_1 = \sum_{i=1}^N X_{Bi}. \quad (4)$$

X_{Bi} is the time it takes the server to decrease the queue size by one for the i 'th time, which has the same distribution as the busy period. The Laplace transform of X_1 is

$$\hat{f}_1(s) = E[e^{sX_1}] = E[E[e^{sX_1}|N]] = E[E[e^{sX_B}]^N] = E[\hat{f}_B(s)^N], \quad (5)$$

where $\hat{f}_B(s)$ is the Laplace transform of $f_B(t)$, the density of the busy period X_B . It can be deduced that [6]

$$\hat{f}_B(s) = \frac{1}{2\rho}(\rho + 1 + s - \sqrt{(\rho + 1 + s)^2 - 4\rho}). \quad (6)$$

Notice that the last step of (5) is nothing but the moment generating function of N calculated at $x = \hat{f}_B(s)$, so

$$\begin{aligned} \hat{f}_1(s) &= \frac{1 - \rho}{1 - \rho\hat{f}_B(s)} \\ &= \frac{1 - \rho}{1 - (\rho + 1 + s - \sqrt{(\rho + 1 + s)^2 - 4\rho})/2} \\ &= \frac{2(1 - \rho)}{1 - \rho - s + \sqrt{(\rho + 1 + s)^2 - 4\rho}} \\ &= \frac{1 - \rho}{2s}[-1 + \rho + s + \sqrt{(\rho + 1 + s)^2 - 4\rho}]. \end{aligned} \quad (7)$$

When there are two queues, the impatient request can be served when either queue becomes empty, so its waiting time is

$$X = \min(X_1, X_2), \quad (8)$$

where the subscripts 1 and 2 stand for queue 1 and queue 2. Because the two queues are independent and identical, X 's CDF

$$F(t) = \text{Prob}[X \leq t] = 1 - (1 - F_1(t))^2. \quad (9)$$

Then $T_2(1, \rho) = EX$, which is

$$\begin{aligned} EX &= \int_0^\infty t dF(t) = - \int_0^\infty t d(1 - F(t)) \\ &= \int_0^\infty (1 - F(t)) dt = \int_0^\infty (1 - F_1(t))^2 dt. \end{aligned} \quad (10)$$

If we let

$$g(t) \equiv 1 - F_1(t), \quad (11)$$

then (10) can be written as

$$EX = \int_0^\infty g^2(t) dt = \int_0^\infty e^{-st} g^2(t) dt \Big|_{s=0} = \hat{g}^2(0), \quad (12)$$

where $\hat{g}^2(s)$ is the Laplace transform of $g^2(t)$. The problem now boils down to calculating $\hat{g}^2(0)$. We first study some of the properties of $\hat{g}(s)$. Because $F_1(t)$ is the integral of $f_1(t)$, we have from (11) that

$$\hat{g}(s) = \frac{1 - \hat{f}_1(s)}{s}. \quad (13)$$

We claim that $s = 0$ is not a pole of $\hat{g}(s)$. This is clear from the expansion of $\hat{f}_1(s)$ near $s = 0$:

$$\hat{f}_1(s) = 1 - \frac{\rho}{(1 - \rho)^2} s + O(s^2), \quad (14)$$

which implies

$$\lim_{s \rightarrow 0} \hat{g}(s) = \frac{\rho}{(1 - \rho)^2}. \quad (15)$$

Hence $\hat{g}(s)$ is analytical at $s = 0$. However, because there is a square root in $\hat{f}_1(s)$ (see (7)), $\hat{g}(s)$ is not analytical on the entire complex plane. Define

$$\rho_1 = -(1 + \sqrt{\rho})^2, \quad \rho_2 = -(1 - \sqrt{\rho})^2. \quad (16)$$

Then the square root in (7) can be written as

$$\sqrt{(\rho + 1 + s)^2 - 4\rho} = \sqrt{(s - \rho_1)(s - \rho_2)}. \quad (17)$$

If we make a branch cut along the line segment $[\rho_1, \rho_2]$, then both $\hat{f}_1(s)$ and $\hat{g}(s)$ are analytical on the plane with the branch cut removed (Fig. 5). The Laplace transform of the product $g^2(t)$ can be represented as a convolution product of $\hat{g}(s)$ and itself:

$$\begin{aligned} \hat{g}^2(s) &= \int_0^\infty g^2(t) e^{-st} dt \\ &= \int_0^\infty \left(\frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \hat{g}(s') e^{s't} ds' \right) g(t) e^{-st} dt \end{aligned} \quad (18)$$

$$\begin{aligned} &= \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \hat{g}(s') ds' \int_0^\infty g(t) e^{-(s-s')t} dt \\ &= \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} \hat{g}(s') \hat{g}(s - s') ds', \end{aligned} \quad (19)$$

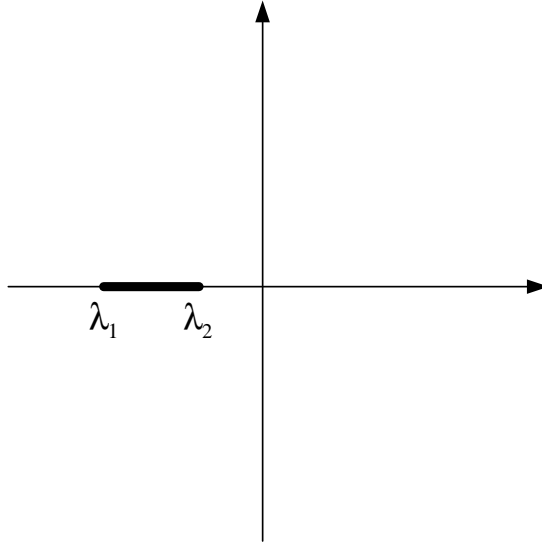


Figure 5: Branch cut of $\hat{f}_1(s)$ and $\hat{g}(s)$.

where (18) is the formula of inverse Laplace transform. c can be any real number larger than the convergence abscissa of $\hat{g}(s)$, which is ρ_2 in our case. In particular, we can take $c = 0$ because $\rho_2 < 0$. Thus

$$\hat{g}^2(0) = \frac{1}{2\pi i} \int_{-i\infty}^{i\infty} \hat{g}(s)\hat{g}(-s)ds. \quad (20)$$

The integral of (20) can be calculated with the aid of the contour shown in Fig. 6. Because $\hat{g}(s)\hat{g}(-s)$ is analytical on the entire complex plane with the two branch cuts removed, we have

$$\left(\int_C + \int_{C_\infty} + \int_{C_1} + \int_{C_2} \right) \hat{g}(s)\hat{g}(-s)ds = 0. \quad (21)$$

As $s \rightarrow \infty$, $\hat{f}_1(s) \rightarrow 1 - \rho$ uniformly. As a result $\hat{g}(s)$ is of the order s^{-1} when s is large, and $\hat{g}(s)\hat{g}(-s)$ is of the order s^{-2} . The length of the half circle is of the order s , so the integral on the path C_∞ is of the order s^{-1} which goes to zero when $s \rightarrow \infty$. Therefore

$$\int_{-i\infty}^{i\infty} \hat{g}(s)\hat{g}(-s)ds = - \left(\int_{C_1} + \int_{C_2} \right) \hat{g}(s)\hat{g}(-s)ds. \quad (22)$$

Because $\hat{g}(s)$ has the same real part on the upper and lower edges of the branch cut, the real parts of the two integrals along C_1 and C_2 with opposite directions cancel out. Only the image part of $\hat{g}(s)$ counts for the integral. The image part of $\hat{g}(s)$ on the upper edge and the lower edge differ only in the sign, so

$$\begin{aligned} & \int_{-i\infty}^{i\infty} \hat{g}(s)\hat{g}(-s)ds \\ &= -2i \int_{C_1} \text{Im}[\hat{g}(s)]\hat{g}(-s)ds \end{aligned}$$

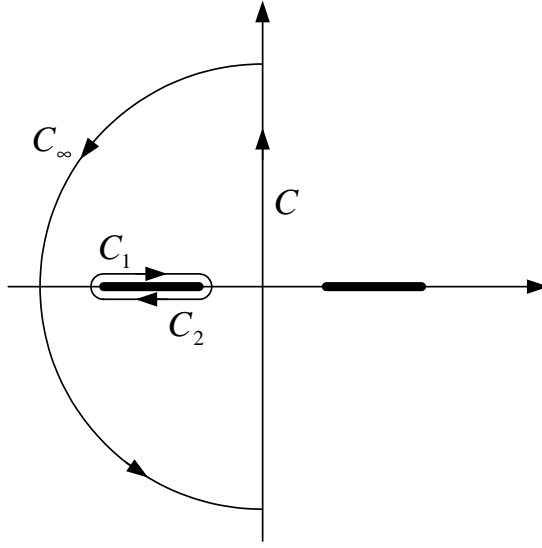


Figure 6: Contour of integral.

$$\begin{aligned}
&= 2i \int_{C_1} \frac{1}{s} \text{Im}[\hat{f}_1(s)] \hat{g}(-s) ds \\
&= i \int_{C_1} \frac{1-\rho}{s^2} \sqrt{s-\rho_1} \sqrt{\rho_2-s} \hat{g}(-s) ds
\end{aligned} \tag{23}$$

Finally we have

$$T_2(1, \rho) = EX(\rho) = \hat{g}^2(0) = \int_{\rho_1}^{\rho_2} \frac{1-\rho}{2\pi s^2} \sqrt{s-\rho_1} \sqrt{\rho_2-s} \hat{g}(-s) ds. \tag{24}$$

A numerical computation of the integral is shown in Fig. 7. To summarize, we have that

Lemma 4.3

$$T_2(1, \rho) = \int_0^\infty t \cdot f(t) dt.$$

In particular, when $m = 2$,

$$T_2(1, \rho) = \hat{g}^2(0) = \int_{\rho_1}^{\rho_2} \frac{1-\rho}{2\pi s^2} \sqrt{s-\rho_1} \sqrt{\rho_2-s} \hat{g}(-s) ds.$$

4.4 Nash equilibrium of the restart game

By the above analysis, we can compute the critical values ρ_1 and ρ_2 where $T_1(0, \rho_1) = T_2(0, \rho_1)$ and $T_1(1, \rho_2) = T_2(1, \rho_2)$. Those values are of importance because of the following statement.

Lemma 4.4 1. When $\rho \leq \rho_1$, for any $0 \leq p \leq 1$, $T_1(p, \rho) \geq T_2(p, \rho)$. Or when $\lambda \leq m\mu\rho_1$, it is always better off to be impatient, i.e. $p_i = 0$, for $i = 1, \dots, n$, is the Nash equilibrium in this case.

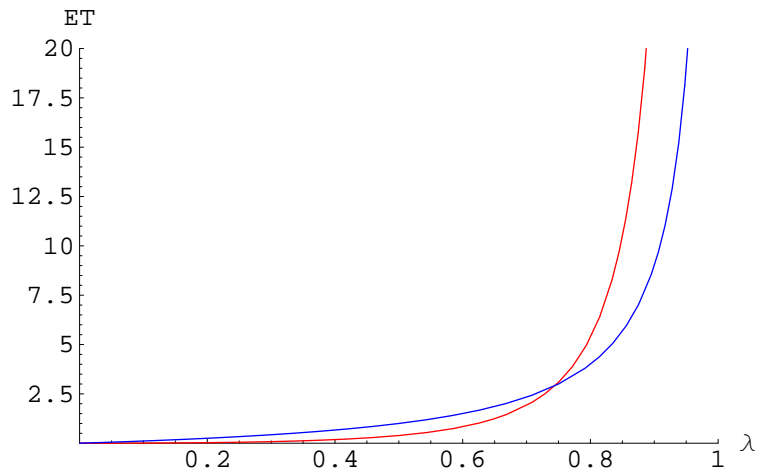


Figure 7: Expected waiting time versus incoming rate. The red line is for the impatient request and the blue line is for patient requests. The two lines cross at $\rho = 0.744$.

m	2	3	4	5	6	7	8	9	10	100
ρ_1	0.618	0.748	0.810	0.847	0.871	0.889	0.902	0.912	0.921	0.991
ρ_2	0.744	0.875	0.925	0.950	0.964	0.973	0.979	0.983	0.987	

Table 1: The critical value ρ_1 and ρ_2

2. When $\rho \geq \rho_2$, for any $0 \leq p \leq 1$, $T_1(p, \rho) \leq T_2(p, \rho)$. Or when $\lambda \geq m\mu\rho_2$, it is always better off to be patient, i.e. $p_i = 1$, for $i = 1, \dots, n$, is the Nash equilibrium.

By Lemma 4.1 and 4.2, we can solve, numerically, for ρ_1 . It is the root of the following equation:

$$\frac{(m\rho)^m}{m!m(\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} (1-\rho)^2 + \frac{(m\rho)^m}{m!} (1-\rho))} = \frac{\rho}{1-p\rho}.$$

Table 1 shows the numerical solution to the above equation for various m 's. The value of ρ_1 approaches 1 when $m \rightarrow \infty$, which is consistent with the intuition that when the number of servers increases, it is more likely that there exists a free server. So when $\rho \leq \rho_1$, the Nash equilibrium is achieved when all the p_i 's are 0. In this case, the Nash equilibrium is also the social optimum. Similarly, we calculate ρ_2 numerically, and the solution is again given in Table 1. In Figure 8, we plot the critical values ρ_1, ρ_2 as a function of the number of servers. When $m \rightarrow \infty$, both ρ_1 and ρ_2 approach to 1. In the regime of $\rho \geq \rho_2$, the Nash equilibrium is achieved when $p_i = 1$ for all the i 's. The mean waiting time at the Nash equilibrium is $\frac{m\rho}{1-m\rho}$, the mean waiting time given by an $M/M/1$ queue, while the social optimum happens when all the requests are impatient. There is a loss of efficiency at the Nash equilibrium. When $m = 2$, $T_2(0, \rho) = \frac{\rho^2}{1-\rho^2}$. Therefore, for two servers, the efficiency of the Nash equilibrium is $T_2(0, \rho)/T_1(1, \rho) = \frac{\rho}{1+\rho}$. When the number of servers increases, the efficiency decreases. In the above, we have characterized the Nash equilibrium when the system is lightly or heavily loaded. In both cases, there exist a homogeneous pure strategy Nash equilibrium. What about

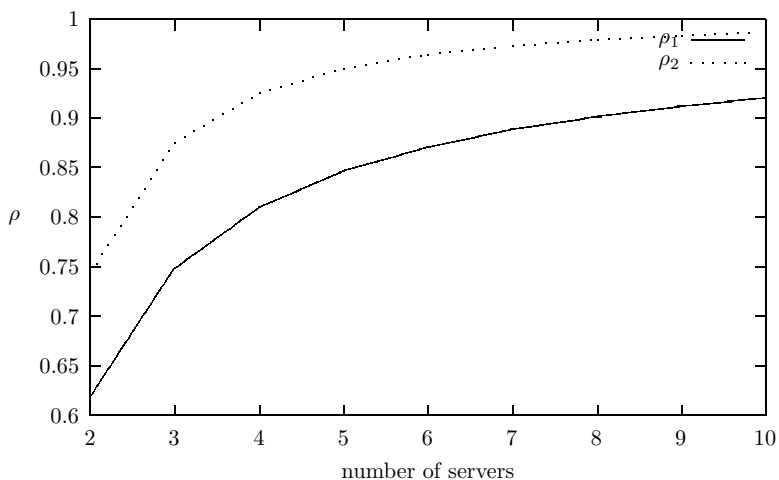


Figure 8: The plot of critical values ρ_1, ρ_2 .

the case when $\rho_1 < \rho < \rho_2$? When $\rho_1 < \rho < \rho_2$, there exists $0 < p_\rho < 1$ such that $T_1(p, \rho) = T_2(p, \rho)$. It is tempting to conclude that if all the users decide with probability p that a request is patient, then it is a Nash equilibrium because it does not seem to make difference when a request switches from patient to impatient and vice versa. This is indeed the case when there are infinitely many users and each user generates one request. The reason is that an individual user's decision will have no effect on the other requests. However, when there are a finite number of users, it is not the case by the following analysis. We will have to need an assumption which we have observed consistently from simulation results but are, unfortunately, unable to prove:

Conjecture. For any fixed $\rho \in (\rho_1, \rho_2)$, T_1, T_2 are continuous, monotonically increasing, and convex with respect to p . That is,

$$T_1'(p), T_2'(p), T_1''(p), T_2''(p) \geq 0.$$

Further, $T_1(p) - T_2(p)$ is monotonically increasing. The above hypothesis implies that there exists a unique $0 < p_\rho < 1$ such that $T_1(p_\rho) = T_2(p_\rho)$. This follows from that $T_1(0) < T_2(0)$, $T_1(1) > T_2(1)$, and $T_1 - T_2$ is continuous and monotonically increasing. Let $p = \frac{\sum_{i=1}^n p_i}{n}$. The expected waiting time $T^{(i)}$ of the i -th player is

$$T^{(i)} = p_i T_1(p) + (1 - p_i) T_2(p).$$

For a combination of strategies p_1, \dots, p_n , it is a Nash equilibrium if for any $1 \leq i \leq n$ and $0 \leq x \leq 1$:

$$T^{(i)} \leq x T_1 \left(p + \frac{x - p_i}{n} \right) + (1 - x) T_2 \left(p + \frac{x - p_i}{n} \right).$$

In particular, when the users are homogeneous, i.e. $p_i = p$, it is:

$$p T_1(p) + (1 - p) T_2(p) \leq x T_1 \left(\frac{n-1}{n} p + \frac{1}{n} x \right) + (1 - x) T_2 \left(\frac{n-1}{n} p + \frac{1}{n} x \right).$$

Let $q = (1 - \frac{1}{n})p$. Then a homogeneous strategy where $p_i \equiv p$ is a Nash equilibrium if the function $F_p(x)$

$$F_p(x) = xT_1\left(q + \frac{1}{n}x\right) + (1-x)T_2\left(q + \frac{1}{n}x\right),$$

achieves the minimum at $x = p$. Assuming the hypothesis, we can show that for any $0 \leq p \leq 1$, $F_p(x)$ is strictly convex.

$$\begin{aligned} F_p'(x) &= T_1\left(q + \frac{1}{n}x\right) + \frac{x}{n}T_1'\left(q + \frac{1}{n}x\right) \\ &\quad - T_2\left(q + \frac{1}{n}x\right) + \frac{1-x}{n}T_2'\left(q + \frac{1}{n}x\right). \end{aligned} \quad (25)$$

$$\begin{aligned} F_p''(x) &= \frac{2}{n}\left[T_1'\left(q + \frac{x}{n}\right) - T_2'\left(q + \frac{x}{n}\right)\right] \\ &\quad + \frac{x}{n^2}T_1''\left(q + \frac{x}{n}\right) + \frac{1-x}{n^2}T_2''\left(q + \frac{x}{n}\right). \end{aligned} \quad (26)$$

Since $T_1 - T_2$ is increasing and $T_1'', T_2'' > 0$, we have that $F_p''(x) > 0$, i.e. F_p is a strictly convex function for any $p > 0$. Now, consider $F_0'(0) = T_1(0) - T_2(0) + \frac{1}{n}T_2'(0)$. Since F_0 is strictly convex, $F_0(0)$ is the minimum of $F_0(x)$ if and only if $F_0'(0) \geq 0$. Therefore, when $F_0'(0) \geq 0$, the strategy $p_i \equiv 0$ is a Nash equilibrium. When $F_0'(0) < 0$, we consider $F_{p_\rho}'(p_\rho)$.

$$\begin{aligned} F_{p_\rho}'(p_\rho) &= T_1(p_\rho) + \frac{p_\rho}{n}T_1'(p_\rho) - T_2(p_\rho) + \frac{1-p_\rho}{n}T_2'(p_\rho) \\ &= \frac{p_\rho}{n}T_1'(p_\rho) + \frac{1-p_\rho}{n}T_2'(p_\rho) > 0. \end{aligned} \quad (27)$$

Therefore, there exists $p \in (0, p_\rho)$ such that $F_p'(p) = 0$. Since $F_p(\cdot)$ is strictly convex, $F_p(x)$ achieves the minimum at $x = p$. Thus $p_i \equiv p$ is a Nash equilibrium. According to Equation (27), $F_{p_\rho}'(p_\rho) \rightarrow 0$ when $n \rightarrow \infty$. Therefore, $p_i \approx p_\rho$ is a Nash equilibrium when there are a large number of users. To summarize, when $\rho_1 < \rho < \rho_2$, there exists a unique homogeneous Nash equilibrium at $p < p_\rho$. At the Nash equilibrium, the impatient requests have longer waiting time than the patient requests.

5 Conclusion

In this paper, we studied the restart strategy of a group of users trying to access multiple servers with many users by modeling it as a queueing problem with game theoretic considerations. We first showed that if the users are cooperative, then by using appropriately chosen restart time, the performance of the system can be very close to a system with an omniscient scheduler, which is often expensive to construct in a large distributed system. In addition, we studied the situation when the users are competitive. In this case we showed the characteristics of the Nash equilibrium for such a game through the use of both computer experiments and analytical analysis of a simplified model. In particular, our experimental results suggest that there does not exist a pure strategy symmetric Nash equilibrium. By analyzing a simplified model, we obtain analytical congestion threshold value that help decide when a user should always restart or always not restart.

There are many open questions raised by this paper, such as finding the Nash equilibrium of the game described in our computer experiments and rigorously proving the conjecture for the simplified model. Another interesting question is to study the restart strategy in a more sophisticated model when there are dependencies among the queues.

Acknowledgment

We thank Alex Fabrikant for his initial work that led to the analysis of the simple model.

References

- [1] G. I. Falin and J. G. C. Templeton. *Retrial Queues*. Chapman & Hall, 1997.
- [2] D. Friedman and B. A. Huberman. Internet congestion: a laboratory experiment. In *Experimental Business Research*, volume II. Kluwer, 2004.
- [3] B. A. Huberman and T. Hogg. Distributed computation as an economic system. *Journal of Economic Perspectives*, pages 141–152, 1995.
- [4] R. Johari and J. N. Tsitsiklis. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research*, 2004.
- [5] F. P. Kelly. Charging and rate control for the elastic traffic. *European Transactions on Telecommunications*, 8:33–37, 1997.
- [6] L. Kleinrock. *Queueing Systems*, volume I and II. John Wiley & Sons, 1976.
- [7] E. Koutsoupias and C. Papadimitriou. Worst-case equilibria. *Lecture Notes in Computer Science*, 1563:404–413, 1999.
- [8] R. T. Maheswaran and T. Basar. Nash equilibrium and decentralized negotiation in acutioning divisible resources. *Group Decision and Negotiation*, 12:361–395, 2003.
- [9] S. M. Maurer and B. A. Huberman. Restart strategies and Internet congestion. *Journal of Economic Dynamics and Control*, 25:641–654, 2001.
- [10] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13:111–124, 1996.
- [11] N. Nisan and A. Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.
- [12] C. H. Papadimitriou. Algorithms, games, and the Internet. In *ACM Symposium on Theory of Computing*, pages 749–753, 2001.
- [13] T. Roughgarden and E. Tardos. How bad is selfish routing? In *IEEE Symposium on Foundations of Computer Science*, pages 93–102, 2000.
- [14] S. Shenker, D. Clark, D. Estrin, and S. Herzog. Pricing in computer networks: reshaping the research agenda. *ACM Computer Communication Review*, 26:19–43, 1996.
- [15] I. Stoica, H. Abdel-Wahab, K. Jeffay, S. Baruah, J. Gehrke, and C. G. Plaxton. A proportional share resource allocation algorithm for real-time, time-shared systems. In *IEEE Real-Time Systems Symposium*, pages 288–299, 1996.