Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 959782, 11 pages
http://dx.doi.org/10.1155/2014/959782

Research Article

Load Balancing for Future Internet: An Approach Based on Game Theory

Shaoyi Song, Tingjie Lv, and Xia Chen

School of Economics and Management, Beijing University of Posts and Telecommunications, Beijing 100876, China

Correspondence should be addressed to Shaoyi Song; songshaoyi.sem@gmail.com

Received 9 September 2013; Accepted 11 December 2013; Published 10 February 2014

Academic Editor: Ching-Jong Liao

In recent years, countries all over the world consider the future internet as the country's strategic development directions, so projects about future internet have been launched by these countries. Load balancing algorithms and job allocations are main research problems in areas of resource management of future internet. In this paper, we introduce a load balancing model for future internet. We formulate the static load balancing problem in the model proposed above as noncooperative game among users and cooperative game among processors. Based on this model, we derive a load balancing algorithm for computing center. Finally, we execute the algorithm presented in this paper with another three algorithms for comparison purpose. The advantages of our algorithm are better scalability to the model, improving system performance, and low cost on maintaining system information.

1. Introduction

Due to rapid development, Internet has become one of the most important infrastructures in information society. All countries in the world have considered Internet's sustainable development as an important method to occupy information technology and strategic demands for enhancing international competitiveness. Therefore, all countries have launched the research programs for future Internet. FIND under US National Natural Science Foundation mainly develops the future Internet architecture and answers "what Internet will be in the next 15 years, and how it will run." GENI project mainly studies the future Internet from the aspect of security, mobility, and sensor network and will build a large open experimental platform which could truly verify the future Internet's architectural design. European 4WARD and FIRE as well as Japanese AKARI have also carried out the study on the future Internet [1–5]. China also launched the National Basic Research Program "Service-oriented Future Internet Structure and Mechanism Research." As one part of this program, this paper focuses on load balancing for computing resources of future internet.

The concept of load balancing is a little different between different researchers in different research areas. There is no description for load balancing in future internet. During our research on future internet, we find that users in future internet may only have a screen used to get access to internet, while all the calculations, the storages, the application services, and some other services provided by our PC nowadays are all offered by internet in future. So in this paper, we define the load balancing of future internet as a mechanism aiming to spread the whole internet's computing load, traffic load, and other items depending on networks' resources self-adaptively and self-organizationally to each resource center equally, to spread the whole work load to each working node in each resource center, to minimize the average task response time to users, to maximize the utilization of whole internet, and to establish a green future internet. In this paper, we focus on the computing load in computing center of future internet.

1.1. Load Balancing Algorithm. In traditional research, load balancing algorithms can be classified as centralized (e.g., [6, 7]) and decentralized (e.g., [8, 9]). In the centralized approach, there is only one node making load balancing decisions, and all the information have to go through this node. All the jobs in the system are allocated by this node to the other nodes to be processed. So there may be the single point of failure. In decentralized approach, all nodes involved in the load balancing decisions. Though, it is more robust than the centralized one, it is costly for many nodes to maintain load balancing information of whole system in decentralized approach. Most decentralized approaches have each node obtaining and maintaining only partial information locally to make suboptimal decisions [10].

According to the stage that the load balancing algorithm implements on, load balancing algorithm can be divided into static (e.g., [6, 11, 12]) and dynamic (e.g., [13–15]). In static load balancing algorithm, all the information about the system is known in advance, and the load balancing strategy has been made by load balancing algorithm at compile time. This load balancing strategy will be kept constantly during runtime of the system. In contrast, dynamic algorithm is implemented at running time, and the load balancing strategies change according to the real statement of the system. Though, the dynamic algorithm has better adaptability, it is sensitive to the accuracy of the load information or statement of system. It may cause terrible mistakes for dynamic approach if the accuracy of information is slightly less than 100%, and in real system 100% accuracy is impossible to achieve.

In recent years, the so-called hybrid scheduling has been receiving some attention [16, 17]. It combines the advantages of static and dynamic algorithms and minimizes their relative inherent disadvantages.

1.2. Related Work. For load balancing problem, plenty of work has been done. For example, Bryhni et al. summarized load balancing algorithms in the area of scalable web service for comparison purpose [18], and Lu et al. proposed a novel load balancing algorithm for dynamic scalable web services [19]. Soror et al. considered a common resource consolidation scenario, in which several database management system instances, each running in a virtual machine, are sharing a common pool of physical computing resources [20].

Most of researchers resolved load balancing problem by using game theory under the environment of distributed computing [11], grid computing [6, 12, 21, 22], and cluster computing [23, 24]. Viscolani characterized the pure-strategy Nash equilibria in a game with two competing profit-maximizing manufacturers who have access to a set of several advertising media [25]. Nathani et al. proposed dynamic planning based scheduling algorithm to maximize resource utilization [26]. Ye and Chen investigated noncooperative games on two resource allocation problems: the server load balancing and the VM placement problems, having proved the existence of a Nash equilibrium for both games [27]. Wu et al. modeled a cooperative behavior control game where the individual utility function is derived from the energy efficiency in terms of the global max-min fairness with the outage performance constraint [28]. Kong et al. investigated the problem of virtual resource allocation in noncooperative cloud environment, where computing resources are provided dynamically in pay-as-you-go manners and virtual machines can selfishly request resource to maximize its own benefit [29].

Recently, with the cloud technology development, many scientists shift their attentions to cloud and data center environment [30–34]. Khiyaita et al. gave an overview of load balancing in the cloud computing such as DNS, ZXTM LB, and AMAZON LOAD BALANCING by exposing the most important research challenges [35]. And we find that scalability, energy efficiency, and green computing are another three objects for load balancing research in now and future, according to [28, 36–39]. As future internet is a new research area, there is few work for load balancing using game theory approach.

In this paper, we present a load balancing model for future internet. Then we propose a semidecentralized solution to the load balancing problem of future internet. This solution is also a hybrid approach that combines a noncooperative game among users and a cooperative game among processors (NOCOG). In this model, all the nodes do not need to maintain as much information as in traditional method. So the advantages of our algorithm are better scalability to the model, improving system performance, better fairness between processor nodes, and low-cost on maintaining system information.

In Section 2, we discuss a load balancing model for future internet firstly. Then we formulate the static load balancing problem in the model presented above as noncooperative game among users and cooperative game among processors. And we derive a load balancing algorithm for future internet, described in Sections 3 and 4. Finally, we compare the algorithm proposed in this paper with the other three existing algorithms, described in Section 5.

2. System Model for Load Balancing in Future Internet

During our research on future internet from the viewpoint of management, we believe that the future internet is easily accessed, hierarchical, virtualized, perceptional, and personalized. As shown in Figure 1, future internet is virtualized into three management layers. Level 1 is a national layer of Resource control and processing center; level 2 is a region layer of Resource control and processing center; level 3 is networks routing nodes that can offer storage and calculating services. Users can get services from internet by accessing to the internet through their Phones, Pads, or some other devices. We assume that all the jobs sent by users can be considered as the requests to network resources. When users send requests to level 3, and if level 3 cannot deal with these requests, these requests may be sent to the upper level. So no matter what layer sending requests is, it acts as a role of user and the upper layer acts as the resource center. Based on this, we present a load balancing model for future internet, as shown in Figure 2.

The load balancing model for future internet proposed in this paper consists of p users, n load managers, and $n * m$ processors. All users generate the jobs/requests and send them to the resource center. The jobs arrive at the resource center and are allocated to processors to be processed by load

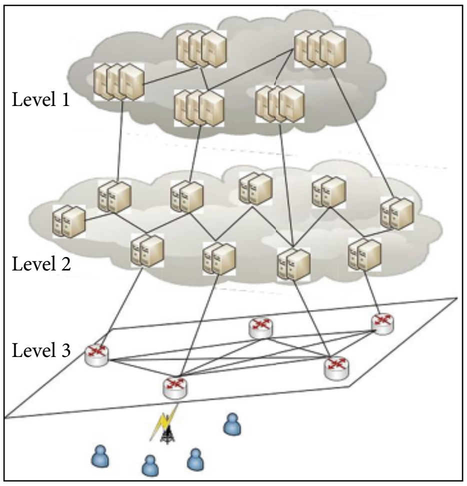

Figure 1: Architecture of future internet.

managers. In this paper, we assume the resource center as computing center offering calculation services.

In this model, all users send jobs to computing center, the jobs are received by load managers randomly, a load manager can get jobs from many users, and jobs from a user can be received by more than one load manager; a load manager dispatches the jobs to the processors which is managed as soon as it receives them; a job may be executed by a allocated processor and wouldn't be dispatched again to another processor; each processor maintains a queue that holds jobs to be executed; each job is processed on a first-come-first-serve (FCFS) basis and then sends the results back to users.

We call processors managed by a load manager and this load manager a cluster, and information and job allocation of a cluster are charged by the load manager. In this model, user i generates jobs and sends them to computing center with an average job generation rate G_i; load manager j with average job/request arrival rate R^j is in charge of dispatching jobs/requests it receives to a processor k it manages with an job sending rate S_k^j; a processor k managed by load manager j is characterized by average processing rate P_k^j and send the result back to users after execution. The vector $S^j = [S_1^j, S_2^j, S_3^j, \dots, S_m^j]^T$ is called the load balancing strategy of cluster j managed by load manager j. The vector $S = [S^1, S^2, S^3, \dots, S^n]^T$ is called the load balancing strategy profile of the whole cloud center. The vector $g^i = [g_i^1, g_i^2, g_i^3, \dots, g_i^n]^T$ is called the job allocation strategy of user i, and $g = [g^1, g^2, g^3, \dots, g^p]^T$ is called the load balancing strategy profile of the whole game.

Most of the previous work on static load balancing formulates the problem as cooperative game among processors, whose main objective is to minimize overall expected response time. Though, fairness index of processors can be reached, fairness between users is difficult to achieve. In this mechanism, some users may still wait for the response of their jobs, while other users' jobs have been processed and sent the results back to users, as this mechanism ignores the selfish character of the users. In the real world, users make their jobs allocation strategies to get the minimal response time. So the noncooperative approach has been proposed. By simulation, we find that it has bad fairness among processors in noncooperative approach. According to these above, we formulate the load balancing problem in future internet as noncooperative game among users. Then we formulate the load balancing problem in a cluster as cooperative game among processors. Our algorithm improves the fairness among processors and the expected system time.

3. Load Balancing Algorithm as Noncooperative Game among Users

A load manager dispatches the jobs to the processors it manage as soon as it receives them; there is no waiting queue at the load manager. So we consider a cluster managed by a load manager as a super processor with an average job processing rate, σ^j:

$$\sigma^j = \sum_{k=1}^{m} P_k^j, \tag{1}$$

where m is the number of processors managed by load manager j. And each cluster is modeled as an M/M/1 (poisson arrivals and exponentially distributed processing times) queuing system [40, 41]. In order to ensure that the model is effective, the average job arrival rate R^j must be less than the total average processing rate of cluster j:

$$R^j < \sigma^j. \tag{2}$$

g_i^j presents the fraction of workload that user i sends to load manager j, where g_i^j is

$$\sum_{j=1}^{n} g_i^j = 1, \tag{3}$$

$$g_i^j \geq 0. \tag{4}$$

As is shown in Figure 2, the problem is to decide how to distribute jobs received from a user to clusters, and once $g_i = [g_i^1, g_i^2, \dots, g_i^n]^T$ is determined by using our algorithm proposed in this paper, cluster j receives the jobs from user i at a rate given by

$$R^j = \sum_{i=1}^{p} g_i^j * G_i. \tag{5}$$

Completion time of a job in such a queuing system involves transfer time and residence time in computing center which consists of executing time of the job and waiting time at the queue. As each cluster is modeled as an M/M/1 queueing system at the point of load manager, so the expected response time of a job at cluster j is given by

$$T^j(g_i) = \frac{1}{\sigma^j - R^j} = \frac{1}{\sigma^j - \sum_{i=1}^{p} g_i^j * G_i}. \tag{6}$$

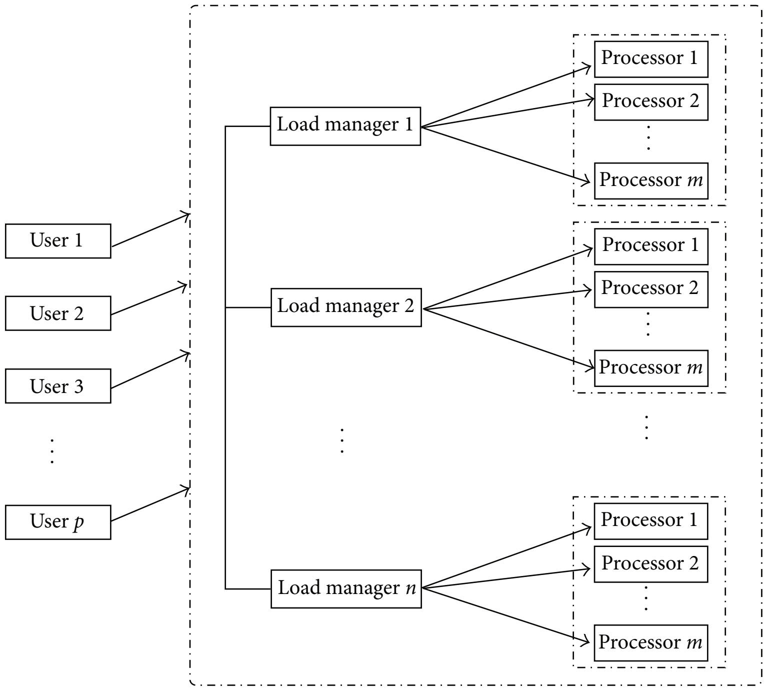

FIGURE 2: Load balancing model for future internet.

We introduce a new variable t_{ij} that defines the transferring time from user i to cluster j, and it may be relevant to the average size of a job, the distance between user and computing center, and the bandwidth available for users and some other factors. Thus, the overall expected response time of user i with its job allocation decision $g_i = [g_i^1, g_i^2, \dots, g_i^n]^T$ is

$$\begin{aligned} T_i(g_i) &= \sum_{j=1}^{n} g_i^j * \left[T^j(g_i) + t_{ij}\right] \\ &= \sum_{j=1}^{n} g_i^j * \left(\frac{1}{\sigma^j - \sum_{i=1}^{p} g_i^j * G_i} + t_{ij}\right). \end{aligned} \tag{7}$$

The goal of a user is to find a load balancing strategy g_i such that $T_i(g_i)$ is minimized, and the decision of a user depends on the load balancing strategy of the other users. For noncooperative game, a Nash equilibrium can be achieved that no user can decrease its average expected response time by unilaterally changing its strategy [42, 43].

So the problem can be translated to the following optimization problem [44]: minimizing the expected response time of a user i given by (7) with the constrains given by (2), (3), and (4). From (7) it can be easily shown that $\partial T_i(g)/\partial g_i^j \geq 0$ and $\partial^2 T_i(g)/\partial g_i^{j^2} \geq 0$; this means that $T_i(g)$ is a convex function in g_i^j and all the constrains (2), (3), and (4) are linear. The first order Kuhn-Tucker (KT) conditions are necessary and sufficient for optimality. The Lagrangian is

$$\text{Lagrange} = \sum_{j=1}^{n}\left(\frac{g_i^j}{\sigma^j - \sum_{i=1}^{p} g_i^j * G_i} + g_i^j t_{ij}\right) - \alpha\left(\sum_{j=1}^{n} g_i^j - 1\right). \tag{8}$$

The KT conditions imply that $g^i = [g_i^1, g_i^2, g_i^3, \dots, g_i^n]^T$ is the optimal solution to minimize (7) if and only if $\alpha \geq 0$ exists. The necessary conditions are

$$\begin{aligned} \frac{\partial \text{Lagrange}}{\partial g_i^j} &= 0, \\ \frac{\partial \text{Lagrange}}{\partial \alpha} &= 0. \end{aligned} \tag{9}$$

Here we introduce a variable σ_i^j, representing the available processing rate at cluster j as seen by user i. σ_i^j is given by (10). Attention, the higher average processing rate a cluster has, the higher fraction of jobs are assigned to it, and the available processing rate of a cluster is

$$\sigma_i^j = \sigma^j - \sum_{q=1,1\neq i}^{p} g_q^j * G_q. \tag{10}$$

According to (10), we can get

$$\sum_{i=1}^{p} g_i^j * G_i = \sigma^j - \sigma_i^j + g_i^j * G_i. \tag{11}$$

Solving (9), we get the following:

$$\alpha = \frac{\sigma_i^j}{\left(\sigma_i^j - g_i^j * G_i\right)^2} + t_{ij}, \tag{12}$$

$$g_i^j = \frac{\sigma_i^j - \sqrt{\sigma_i^j / \left(\alpha - t_{ij}\right)}}{G_i}. \tag{13}$$

Due to constrain (4), we have $g_i^j = 0$, and set this equation to (12):

$$\alpha = \frac{1}{\sigma_i^j} + t_{ij}. \tag{14}$$

If we consider all the n clusters in (12) and (13), the equation may show us how negative g_i^j occurs. These are due to the cluster with low processing rate. So some clusters should be excluded, and the fraction of jobs assigned to these clusters should be set to 0. We sort the cluster according to (15), and $s^1 < s^2 < s^3 < \cdots < s^n$, where s^j is given by

$$s^j = \frac{1}{\sigma_i^j} + t_{ij}. \tag{15}$$

This means that there exists an index r ($1 \leq r \leq n$), making $g_i^r = 0$ ($p \leq r \leq n$). So we get the minimum index r, which satisfies the inequality below (16), according to (3):

$$\min_{1\leq r\leq n} r : \frac{\sum_{j=1}^{r} \sigma_i^j}{\sum_{j=1}^{r} \sigma_i^j - G_i} \leq \sqrt{s^r - t_{ij}}. \tag{16}$$

We get α, where α satisfies (17):

$$\sqrt{\alpha - t_{ij}} = \frac{\sum_{j=1}^{r} \sqrt{\sigma_i^j}}{\sum_{j=1}^{r} \sigma_i^j - G_i}. \tag{17}$$

Finally g_i^j is given by (18):

$$g_i^j = \begin{cases} \dfrac{\sigma_i^j - \sqrt{\sigma_i^j / \left(\alpha - t_{ij}\right)}}{G_i} & 1 \leq j < r \\ 0 & \text{others}. \end{cases} \tag{18}$$

Algorithm for noncooperative game among users: (see Algorithm 1).

4. Cooperative Game among Processors of a Cluster

A Cluster j with average job arrival rate R^j should dispatch jobs to the processors he manages immediately when he receives these jobs from users. Then the processors execute these jobs and send the result back to users. As mentioned in Section 4, each processor maintains a waiting queue and executes a job at a time with FCFS. We model each processor an M/M/1 queuing system. Here, we ignore the transfer time of a job from a load manager to a processor, as they are connected with an inner communication network in computing center. So the response time of a job in a cluster consists of its processing time (T'_{proc}) and waiting time (T'_{wait}) in the queue:

$$T' = T'_{\text{proc}} + T'_{\text{wait}}. \tag{19}$$

We get the average response time of a job at a processor k in a cluster j, T_k^j:

$$T_k^j = \frac{1}{P_k^j - S_k^j}, \tag{20}$$

where P_k^j presents the average processing rate of processor k in cluster j and S_k^j is the sending rate of load manager j to processor k it manages. As a job will not be transferred to another processor after being dispatched to a processor k, S_k^j is also the job arrival rate of processor k of cluster j. And all the jobs dispatched by load managers should be executed; that is

$$R^j = \sum_{k=1}^{m} S_k^j. \tag{21}$$

Now we formulate this problem as a cooperative game among processors of a cluster, where (22) is the cost function of a processor. All the processors work cooperatively to finish all the jobs as fast as possible. So the Nash bargaining solution is determined by solving the following optimization problem:

$$\min_{S_k^j} \quad T^j = \sum_{k=1}^{m} \ln\left(\frac{1}{P_k^j - S_k^j}\right), \tag{22}$$

$$\text{S.t.} \quad S_k^j \geq 0, \tag{23}$$

$$R^j = \sum_{k=1}^{m} S_k^j. \tag{24}$$

From (22) it can be easily shown that $\partial T^j / \partial S_k^j \geq 0$ and $\partial^2 T^j / \partial {S_k^j}^2 \geq 0$; this means that T^j is a convex function in S_k^j, and all the constrains (23) and (24) are linear. The first-order Kuhn-Tucker conditions are necessary and sufficient for optimality. The Lagrangian is

$$\text{Lagrange} = \sum_{k=1}^{m} \ln\left(\frac{1}{P_k^j - S_k^j}\right) - \beta\left(R^j - \sum_{k=1}^{m} S_k^j\right). \tag{25}$$

The Kuhn-Tucker conditions imply that R^j, $j = 1, 2, \ldots, n$, is the optimal solution to (22) if and only if $\beta \geq 0$, such that

$$\frac{\partial \text{Lagrange}}{\partial S_k^j} = 0, \qquad \frac{\partial \text{Lagrange}}{\partial \beta} = 0. \tag{26}$$

```
Input:
Users' generating rate: $[G^1, G^2, G^3, \ldots, G^p]$;
Processors' processing rate: $\begin{bmatrix} P_1^1 & \cdots & P_1^n \\ \vdots & \ddots & \vdots \\ P_m^1 & \cdots & P_m^n \end{bmatrix}$;
The number of users, load managers, processors for each load manager: $k$, $n$, $m$;
Transforming time: $t_{ij}$
Fault tolerant: $\varepsilon$
Output:
    Fraction of job assignment for each user: $\begin{bmatrix} g_1^1 & \cdots & g_1^n \\ \vdots & \ddots & \vdots \\ g_k^1 & \cdots & g_k^n \end{bmatrix}$
While (1) do
$\sigma_i^j \leftarrow \sigma_i^j - \sum_{q=1,q!=i}^{p} g_q^j * G_q$
For $i = 1, 2, 3, \ldots, k$
    $P \leftarrow 1$; $\alpha \leftarrow 0$;
While (1) do
    For $j = 1, 2, 3, \ldots, r$ do    $S = \sum_{j=1}^{r} \frac{\sigma_j^j - \sqrt{\sigma_i^j/(\alpha - t_{ij})}}{G_i}$
    If $\left( s \le \sqrt{\frac{1}{\sigma_i^p}} \right)$ $p \leftarrow p + 1$;   Else BREAK;
$\alpha \leftarrow (s)^2 + t_{ij}$;
For $j = 1, 2, 3, \ldots, n$
    if $(j < P)$ $g_i^j = \frac{\sigma_i^j - \sqrt{\sigma_i^j/(\alpha - t_{ij})}}{G_i}$;   else $g_i^j \leftarrow 0$;
        $T_i \leftarrow T_i + \frac{g_i^j}{\sigma_i^j - g_i^j G_i} + g_i^j t_{ij}$;
For $i = 1, 2, 3, \ldots, k$ do
If $(T_i - T_i' > \varepsilon)$ $T_i' \leftarrow T_i$;
            Else BREAK;
```

Algorithm 1

From (26), we may get the following:

$$\frac{1}{P_k^j - S_k^j} - \beta = 0. \tag{27}$$

Solving (27) we get S_k^j the following:

$$S_k^j = P_k^j - \frac{1}{\beta}. \tag{28}$$

Considering (24) and (28), we get (29), which determines the Lagrange multiplayer β:

$$R^j = \sum_{k=1}^{m} S_k^j = \sum_{k=1}^{m} \left(P_k^j - \frac{1}{\beta} \right). \tag{29}$$

Due to the constrain $S_k^j \ge 0$, we set $S_k^j = 0$ into (28), getting the following:

$$\beta = \frac{1}{P_k^j}. \tag{30}$$

If we consider all the m processors in (29), (28) may show us how negative S_k^j occurs. These are due to the processors with low processing rate. So some processors should be excluded, and the arrival rate of these processors should be set 0. We sort the processors by their value of t_k^j ($t_1^j < t_2^j < t_3^j < \cdots < t_m^j$), where t_k^j satisfies the following (31):

$$t_k^j = \frac{1}{P_k^j}. \tag{31}$$

This means that there exists an index f ($1 \le f \le m$), making $P_k^j = 0$ ($f \le k \le m$). So we get the minimum index q, which satisfies the inequality below (32):

$$\min_{1 \le f \le m} f : R^j \le \sum_{k=1}^{f} \left(P_k^j - \frac{1}{t_q^j} \right). \tag{32}$$

We get β that satisfies (33) with index f got in (32):

$$R^j = \sum_{k=1}^{f} \left(P_k^j - \frac{1}{\beta} \right). \tag{33}$$

Finally S_k^j is given by (34) as follows:

$$S_k^j = \begin{cases} P_\beta^j - \dfrac{1}{\beta} & 1 \le k < f, \\ 0 & \text{others.} \end{cases} \tag{34}$$

Above all, $S^j = [S_1^j, S_2^j, S_3^j, \ldots, S_m^j]^T$ is the vector of load allocation of cluster j, and $S = [S^1, S^2, S^3, \ldots, S^n]^T$ is the load balancing profile of the whole cloud computing center.

Algorithm for cooperative game among processors in a cluster is shown in Algorithm 2.

5. Results and Discussion

We perform simulations to study the fairness between processors and between users at a certain utilization of computing center, expected response time for users, and effect of system load. In this simulation environment, there are 15 processors managed by 4 load managers as shown in Table 1. And there are 7 users generating jobs and sending them to computing center; the relative job generation rates are shown in Table 2.

Processors numbered 1 to 4 are managed by load manager 1, processors numbered 5 to 8 are managed by load manager 2, processors numbered 9 to 11 are managed by load manager 3, and processors numbered 12 to 15 are managed by load manager 4. Each relative processing rate is processing rate divided by the lowest processing rate in the center. A user can send his jobs to any load manager according to its job allocation policy, which is made by gaming with the other users. Each relative job generation rate is a user's job generation rate divided by total job generation rate of all users. A Load manager allocates the jobs to processors it manages immediately when it receives them from users. A processor maintains a waiting queue modeled as M/M/1 queuing system, processes jobs receiving from its load manager, and sends the results back to users.

As we assume that the total job generation rate cannot be faster than the total processing rate of center, the relative job generation rate presents the proportion of a user's job generation rate to the total job generation rate of all the users. Here we introduce a variable ρ, which presents the utilization of computing center, and the real job generation rate of a user is given by (35) as follows:

$$G_i' = \gamma_i * \rho * \sum_{j=1}^{n} \sum_{k=1}^{m} P_k^j, \tag{35}$$

where n is the number of load manager, m is processor number of a cluster, and γ_i presents the relative job generation rate. During whole simulation, we assume that the average communication delay time is 0.01 second.

During simulation, we implement another three algorithms presented in [9, 10, 15] for comparision purpose. Algorithm in [9] labeled as COG is formulated as a cooperative game problem among brokers, but each broker needs to maintain the information of all providers. The algorithm in [10] labeled as NOG is formulated as a non-cooperative game problem among users; the jobs are sent to the processor directly according to the load balancing profile of users. And algorithm in [15] labeled as PS is a proportional-scheme algorithm; each user should maintain the information of all processors. According to the information maintained, user i in PS decides how to allocate its jobs to processer j at a job sending rate $\varphi_{i,k}$, where $\varphi_{i,k}$ is given below:

$$\varphi_{i,k} = G_i' * \frac{P_k^j}{\sum_{k=1}^{m} P_k^j}, \tag{36}$$

where m is the number of processors in the system. All these three algorithms are not hierarchical; jobs in these methods are sent to the processors to be processed directly, while the algorithm proposed in this paper is hierarchical, and the computing center simulates a noncooperative game among users then dispatches job to the load managers based on load balancing profile got from our algorithm; and load managers allocate them to processors.

5.1. Fairness. Fairness is an important measure of quality to a load balancing algorithm. For users, fairness indicates that each user has the same response time, and the scenario that a user is still waiting for its response while the other users have finished their jobs cannot exist. For processors, fairness indicates that each processor has the same average job completion time. If a load balancing algorithm is 100% fair, the fairness index (FI) would be 1.0. The fairness index FI is given by (37):

$$\text{FI} = \frac{\left(\sum_{j=1}^{n} T_k^j \right)^2}{n * \sum_{j=1}^{n} T_k^j}, \tag{37}$$

where T_k^j is the average completion time of processor k managed by load manager j discussed in [45]. In this part of our simulation, the utilization of computing center is varied from 10% to 90% increasing by 10%. In Figure 3, we present the fairness index for different values of computing center utilization ranging from 10% to 90% with 10% increase. It can be observed that, for users, algorithms NOCOG and PS maintain a fairness index of 1 over the whole range of computing center utilization. The NOG method has a fairness index close to 1 at the low utilization of the computing center and has a fairness index of 1 at the high utilization. Conversely, the fairness index of COG scheme is lower than that of the other three schemes. For processors, with increasing the utilization, the fairness index of NOCOG method is growing up to 0.95 from 0.58, which is better than the PS and NOG algorithms, while the COG scheme has a fairness index of 1 over the whole simulation. Above all, the algorithm proposed in this paper improves the fairness index of processors in the premise of guaranteeing the fairness of users, compared with the traditional noncooperative method such as NOG. And at

```
INPUT: Load Managers' load vector/arriving rate: $[R^1, R^2, R^3, \ldots, R^n]$;
        Processors' processed rate: $\begin{bmatrix} P_1^1 & \cdots & P_1^n \\ \vdots & \ddots & \vdots \\ P_m^1 & \cdots & P_m^n \end{bmatrix}$;
OUTPUT: Processors' load vector/arriving rate: $\begin{bmatrix} S_1^1 & \cdots & S_1^n \\ \vdots & \ddots & \vdots \\ S_m^1 & \cdots & S_m^n \end{bmatrix}$
Initialization:
  $\beta \leftarrow 0; q \leftarrow 1$;
  For $j = 1, 2, 3, \ldots, n$ do
    If $(R^j > 0)$
For $k = 1, 2, 3, \ldots, m$ do // Sort the processors in increasing order of the equation below
such that $t_1^j < t_2^j < t_3^j < \cdots < t_m^j$
            $t_k^j \leftarrow \frac{1}{P_k^j}$ ;
RIGHT $\leftarrow$ 0; // Find the minimum $f$ to satisfy the inequality
        $R^j \le \sum_{k=1}^{f}(P_k^j - \frac{1}{t_f^j})$
While $(f \le m)$ do
For $k = 1, 2, \ldots, f$ do
            RIGHT $\leftarrow$ RIGHT $+ P_k^j - \frac{1}{t_f^j}$;
If (RIGHT $\ge R^j$)
    BREAK;
Else
            $f \leftarrow f + 1$
            $a \leftarrow 0; b \leftarrow 1$; SUM $\leftarrow$ 0; // Calculate $\alpha$ using a binary search; sort $\sigma^i$ in
            increase order
While (1) do
        $\beta \leftarrow \frac{a+b}{2}$;
        For $k = 1, 2, 3, \ldots, q$ do
            SUM $\leftarrow$ SUM $+ P_k^j - \frac{1}{\beta}$;
        If $(|\mathrm{SUM} - R^j| < \varepsilon)$ BREAK
If (SUM $> R^j$)    $b \leftarrow \beta$;
        Else        $a \leftarrow \beta$;
      For $i = 1, 2, 3, \ldots, m$ do
If $(i \le q)$
        $S_k^j = P_k^j - \frac{1}{\beta}$;
```

Algorithm 2

the high utilization (from 70% to 90%), the processes fairness index of NOCOG algorithm is very closed to that of COG algorithm.

5.2. Response Time. In this simulation, we use the same system configuration as before and set the utilization of the cloud to 70%. According to the fairness definition, we devise a job allocation method for each of the users in COG algorithm:

$$G_i^j = R^j * \frac{G_i}{\sum_{i=1}^{p} G_i}. \tag{38}$$

And we devise a job allocation method for users in PS algorithm:

$$G_i^j = G_i * \frac{R^j}{\sum_{j=1}^{n} R^j}. \tag{39}$$

And the expeted job response time for each user is shown in Figure 4, when the algorithm is at equilibrium. The average job completion consists of transferring time to computing center waiting time at the queue, and executing time of job itself. Figure 4 shows that the NOCOG and PS algorithm guarantee equal expected response time for all users but with the disadvantage of a higher expected response time,

Table 1: Configure information of computing center.

Load manager	1				2				3			4			
Processor	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Processing rate	10	20	50	100	10	20	20	100	10	20	100	50	50	100	100
Relative processing rate	1	2	5	10	1	2	2	10	1	2	10	5	5	10	10

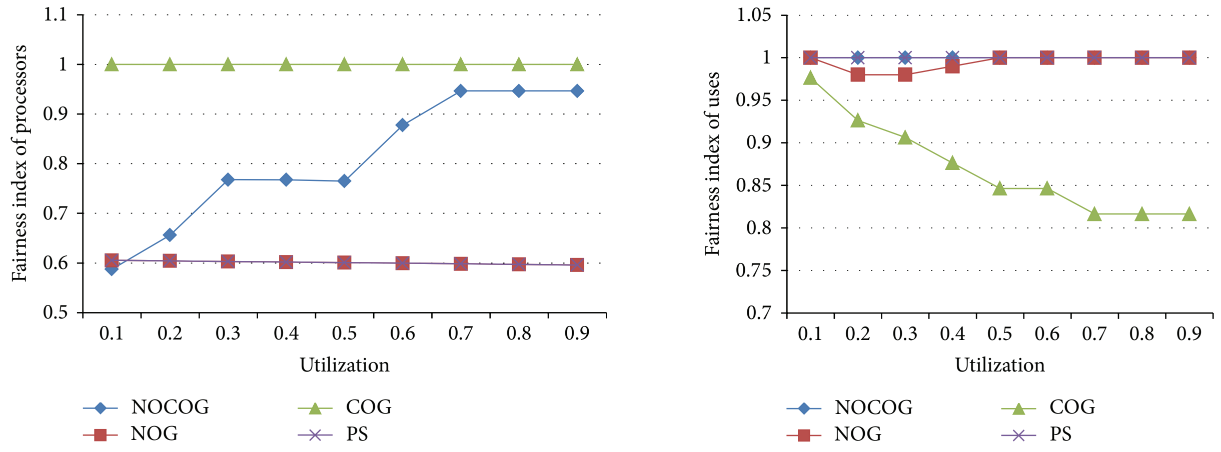

Figure 3: Fairness index versus utilization of computing center.

Table 2: Relative job generation rate of users.

User	1	2-3	4–6	7
Relative job generation rate	0.25	0.2	0.1	0.05

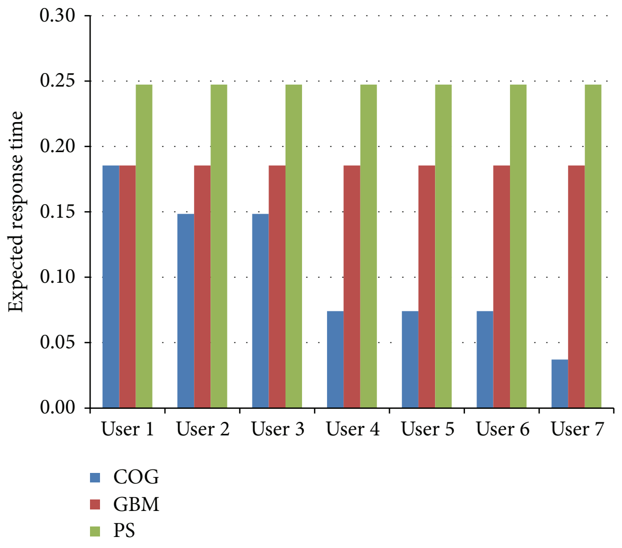

Figure 4: Expected response time for each user.

while the COG algorithm has a better expected response time for each user. We can also get that the NOCOG and PS algorithms have a better fairness index.

5.3. Effect of System Load. In this part, we also vary the computing center utilization from 10% to 90% with 10% increase. All the simulation configurations are the same with the two simulations above. We get the real job generation rate of users via (35) then we calculate the load balancing profile of cloud center and the average job completion time of each processor. Then we get the average job completion time (avg) of cloud center at the utilization of computing center varied from 10% to 90% via (40):

$$\text{avg} = \frac{\sum_{j=1}^{n} \sum_{k=1}^{m} S_k^j * T_k^j}{\sum_{j=1}^{n} \sum_{k=1}^{m} S_k^j}. \qquad (40)$$

Figure 5 shows the average system time of each algorithm versus the utilization of computing center ranging from 10% to 90% with 10% increase. The average system time of PS algorithm is always higher than that of the other three algorithms, and the average system of COG algorithm is always lower than that of the other three algorithms. At the low utilization (from 10% to 40%) of computing center, our NOCOG algorithm is better than NOG. At the middle utilization (from 50% to 70%), the average system time of two algorithms is nearly the same. And at the high utilization of computing center, the NOCOG algorithm is better than NOG again.

In Figure 6, it is shown how many processors in the center are participating in job processing for each algorithm. For NOG and PS algorithms, all the processors participate in job processing no matter what system load is. For NOCOG and COG algorithms, the same number of processors is participating in job processing at 10%, 20%, 40%, 80%, and 90% load. And at another system load, there are less processors participating in job processing of NOCOG algorithm than

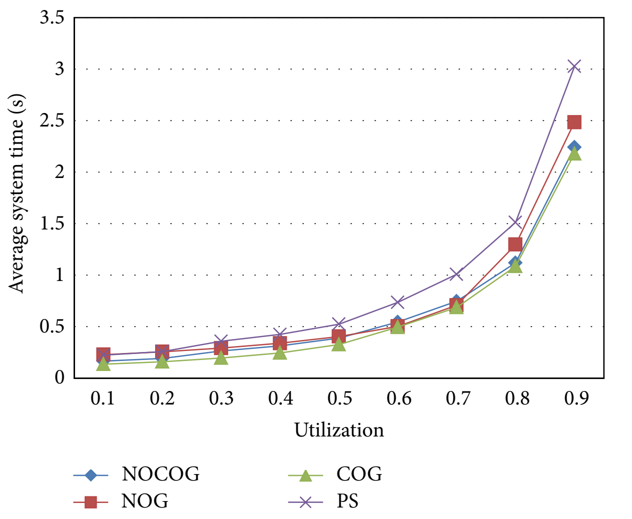

FIGURE 5: Average system time versus computing center utilization.

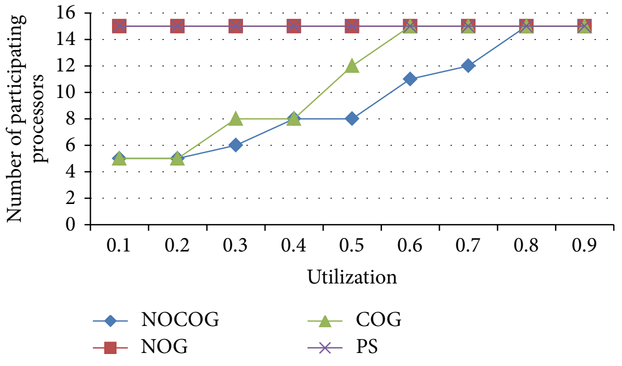

FIGURE 6: Number of participating processors versus system load.

those of COG algorithm. So our algorithm can achieve the goal of energy conservation and emission reduction.

6. Conclusion

In this paper, we define the concept of load balancing in future internet, discuss the probable architecture of future internet, and present a new framework to solve problem for computing center in future internet. As a cooperative game among processors approach considers the minimal system executing time as its goal; the fairness index of users is ignored. Conversely, a noncooperative game among users approach considers the minimal job response time of users as its goal; the fairness index of processors is also ignored. At the same time, we believe that the future internet is service-oriented, so we solve the load balancing problem in future internet from the perspective of users. According to the model we establish, we formulate the problem as noncooperative game among users and cooperative game among processors. From the simulation results, we can draw conclusions that the algorithm proposed in this paper improves the fairness index of processors in the premise of guaranteeing the fairness of users, improves the scalability and efficiency of system compared with other noncooperative game methods among users, and achieves the goal of energy conservation and emission reduction.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This project was supported by National Basic Research Program of China (973 Program) under Grant no. 2012CB315805, National Natural Science Foundation of China under Grant no. 71172135 and no. 71231002, and the Fundamental Research Funds for the Central Universities under Grant no. 2013RC0603.

References

[1] Future internet design (FIND) program, http://www.nets-find.net/.

[2] L. Peterson, T. Anderson, D. Blumenthal et al., "GENI design principles," *IEEE Computer*, vol. 39, no. 9, pp. 102–105, 2006.

[3] N. Niebert, S. Baucke, I. El-Khayat et al., "The way 4 WARD to the creation of a future internet," in *Proceedings of the IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '08)*, pp. 1–5, September 2008.

[4] G. Anastasius, K. Arto, F. Serge, M. Martin, and P. Martin, "Future internet research and experimentation: the FIRE initiative," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 89–92, 2007.

[5] H. Harai, "AKARI architecture design for new generation network," in *Proceedings of the IEEE/LEOS Summer Topical Meeting (LEOSST '09)*, pp. 155–156, July 2009.

[6] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "A cooperative game framework for QoS guided job allocation schemes in grids," *IEEE Transactions on Computers*, vol. 57, no. 10, pp. 1413–1422, 2008.

[7] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *Journal of the ACM (JACM)*, vol. 32, no. 2, pp. 445–465, 1985.

[8] L. F. Bittencourt, F. K. Miyazawa, and A. L. Vignatti, "Distributed load balancing algorithms for heterogeneous players in asynchronous networks," *Journal of Universal Computer Science*, vol. 18, no. 20, pp. 2771–2797, 2012.

[9] D. Grosu, A. T. Chronopoulos, and M. Y. Leung, "Load balancing in distributed systems: An approach using cooperative games," in *Proceedings of the IEEE Abstracts and CD-ROM Parallel and Distributed Processing Symposium (IPDPS '02)*, pp. 52–61, 2002.

[10] D. Grosu and A. T. Chronopoulos, "Noncooperative load balancing in distributed systems," *Journal of Parallel and Distributed Computing*, vol. 65, no. 9, pp. 1022–1034, 2005.

[11] D. Grow and A. T. Chronopoulos, "A game-theoretic model and algorithm for load balancing in distributed systems," in *Proceeding of the 16th International Parallel and Distributed Processing Symposium (IPDPS '02)*, Fort Lauderdale, Fla, USA, 2002.

[12] G. Murugaboopathi, V. Sujathabai, and T. K. S. R. Babu, "A QoS based grid job allocation scheme using game theoretic approach," *International Journal*, vol. 2, no. 8, 2012.

[13] S. Penmatsa and A. T. Chronopoulos, "Dynamic multi-user load balancing in distributed systems," in *Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07)*, pp. 1–10, March 2007.

[14] R. Schlagenhaft, M. Ruhwandl, H. Bauer, and C. Sporrer, "Dynamic load balancing of a multi-cluster simulator on a network of workstations," in *Proceedings of the 9th Workshop on Parallel and Distributed Simulation (PADS '95)*, pp. 175–180, Lake Placid, NY, USA, June 1995.

[15] Y. C. Chow and W. H. Kohler, "Models for dynamic load balancing in a heterogeneous multiple processor system," *IEEE Transactions on Computers*, vol. 100, no. 5, pp. 354–361, 1979.

[16] M. Avvenuti, L. Rizzo, and L. Vicisano, "A hybrid approach to adaptive load sharing and its performance," *Journal of Systems Architecture*, vol. 42, no. 9-10, pp. 679–696, 1997.

[17] Y. Li, Y. Yang, M. Ma, and L. Zhou, "A hybrid load balancing strategy of sequential tasks for grid computing environments," *Future Generation Computer Systems*, vol. 25, no. 8, pp. 819–828, 2009.

[18] H. Bryhni, E. Klovning, and Ø. Kure, "A Comparison of load balancing techniques for scalable Web servers," *IEEE Network*, vol. 14, no. 4, pp. 58–64, 2000.

[19] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg, "Join-Idle-Queue: a novel load balancing algorithm for dynamically scalable web services," *Performance Evaluation*, vol. 68, no. 11, pp. 1056–1071, 2011.

[20] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, "Automatic virtual machine configuration for database workloads," *ACM Transactions on Database Systems*, vol. 35, no. 1, article 7, 2010.

[21] R. Subrata, A. Y. Zomaya, and B. Landfeldt, "Game-theoretic approach for load balancing in computational grids," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 1, pp. 66–76, 2008.

[22] S. K. Goyal and M. Singh, "Adaptive and dynamic load balancing in grid using ant colony optimization," *International Journal of Engineering and Technology*, vol. 4, no. 9, p. 167, 2012.

[23] V. Ungureanu, B. Melamed, and M. Katehakis, "Effective load balancing for cluster-based servers employing job preemption," *Performance Evaluation*, vol. 65, no. 8, pp. 606–622, 2008.

[24] Q. Zhang, A. Riska, W. Sun, E. Smirni, and G. Ciardo, "Workload-aware load balancing for clustered web servers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 3, pp. 219–233, 2005.

[25] B. Viscolani, "Pure-strategy Nash equilibria in an advertising game with interference," *European Journal of Operational Research*, vol. 216, no. 3, pp. 605–612, 2012.

[26] A. Nathani, S. Chaudhary, and G. Somani, "Policy based resource allocation in IaaS cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 94–103, 2012.

[27] D. Ye and J. Chen, "Non-cooperative games on multi-dimensional resource allocation," *Future Generation Computer Systems*, no. 6, pp. 1345–1352, 2013.

[28] D. Wu, Y. Cai, L. Zhou et al., "Cooperative strategies for energy-aware ad hoc networks: a correlated equilibrium game-theoretic approach," in *Proceedings of the IEEE Transactions on Vehicular Technology*, vol. 62, no. 5, pp. 2303–2314, 2013.

[29] Z. Kong, C. Z. Xu, and M. Guo, "Mechanism design for stochastic virtual resource allocation in non-cooperative cloud systems," in *Proceedings of the IEEE 4th International Conference on Cloud Computing (CLOUD '11)*, pp. 614–621, Washington, DC, USA, July 2011.

[30] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 34–39, 2013.

[31] S. Begum and C. S. R. Prashanth, "Review of Load Balancing in Cloud Computing," *International Journal of Computer Science Issues*, vol. 10, no. 1, p. 343, 2013.

[32] M. Randles, D. Lamb, and A. Taleb-Bendiab, "A comparative study into distributed load balancing algorithms for cloud computing," in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA '10)*, pp. 551–556, April 2010.

[33] S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 1, pp. 108–113, July 2010.

[34] P. Membrey, D. Hows, and E. Plugge, "Load Balancing in the Cloud," in *Practical Load Balancing*, pp. 211–224, Apress, 2012.

[35] A. Khiyaita, M. Zbakh, and H. El Bakkali, "Load balancing cloud computing: state of art," in *Proceedings of the National Days of IEEE Network Security and Systems (JNS2 '12)*, pp. 106–109, Marrakech, Morocco, 2012.

[36] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '10)*, pp. 1–9, March 2010.

[37] D. Wu and Y. Cai, "A cooperative communication scheme based on dynamic coalition formation game in clustered wireless sensor networks," in *Proceedings of the 54th Annual IEEE Global Telecommunications Conference: "Energizing Global Communications" (GLOBECOM '11)*, pp. 1–6, December 2011.

[38] X. León and L. Navarro, "A Stackelberg game to derive the limits of energy savings for the allocation of data center resources," *Future Generation Computer Systems*, vol. 29, no. 1, pp. 74–83, 2013.

[39] J. Baliga, R. W. A. Ayre, K. Hinton, and R. S. Tucker, "Green cloud computing: balancing energy in processing, storage and transport," *Proceedings of the IEEE*, vol. 99, no. 1, pp. 149–167, 2010.

[40] N. Chee-Hock and S. Boon-He, *Queueing Modelling Fundamentals*, John Wiley & Sons, Chichester, UK, 2002.

[41] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, Chichester, UK, 1991.

[42] T. Basar, G. J. Olsder, and G. J. Clsder, *Dynamic Noncooperative Game Theory*, Academic Press, London, UK, 1995.

[43] R. Gibbons, *A Primer in Game Theory*, Prentice Hall, 1992.

[44] J. C. Harsanyi and R. Selten, *A General Theory of Equilibrium Selection in Games*, vol. 1, MIT Press Books, 1988.

[45] R. Jain, D. M. Chiu, and W. R. Hawe, *A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System*, Eastern Research Laboratory, Digital Equipment Corporation, 1984.