Dynamic Task Assignment and Resource Management in Cloud Services Using Bargaining Solution

KwangSup Shin[1], Myung-Ju Park[2] and Jae-Yoon Jung[2†]

[1] Graduate School of Logistics, Incheon National University

12-1 Songdo-Dong, Yeonsu-Gu, Incheon, 406-840, Korea

ksshin@incheon.ac.kr

[2] Department of Industrial and Management Systems Engineering, Kyung Hee University

1 Seocheon-Dong, Giheung-Gu, Yongin-Si, Gyeonggi-Do, 446-701, Korea

{pmj0684, jyjung}@khu.ac.kr

[†] Corresponding author

Abstract

Cloud computing provides infrastructure, platform and software as services to customers. For the purpose of providing reliable and truthful service, a fair and elastic resource allocation strategy is essential from the standpoint of service customers. In this paper, we propose a game-theoretic mechanism for dynamic cloud service management, including task assignment and resource allocation to provide reliable and truthful cloud services. A user utility function is first devised considering the dynamic characteristics of cloud computing. The elementary stepwise system is then applied to efficiently assign tasks to cloud servers. A resource allocation mechanism based on bargaining game solution is also adopted for fair resource allocation in terms of quality of service of requested tasks. Through numerical experiments, it is shown that the proposed mechanism guarantees better system performance than several existing methods. The experimental results show that the mechanism completes the requested tasks earlier with relatively higher utility while providing a significant level of fairness compared to existing ones. The proposed mechanism is expected to support cloud service providers in elastically managing their limited resources in a cloud computing environment in terms of quality of service.

Keywords: *Bargaining Solution, Cloud Computing, Quality of Service, Resource Allocation, Task Assignment*

Article first published online in Concurrency and Computation: Practice and Experience on 29 August, 2013.

1. Introduction

1.1 Research background

Gartner recently identified cloud computing as one of the top strategic technologies for the next ten years [1]. Cloud computing is designed to power the next-generation data centers as the enabling platform for dynamic and flexible application provisioning. This is facilitated by exposing data center's capabilities as a network of virtual services so that users are able to access and deploy applications from anywhere in the Internet driven by the demand and Quality of Service (QoS) requirements [2]. The fundamental concept of cloud computing is to share resources among cloud service consumers, cloud partners and cloud vendors in a value chain [3]. The essential characteristics of Cloud computing can be represented with scalability, flexibility, quality of service (QoS) guaranteed on-demand service provisioning and resource pooling [4]. By exploring these advantageous functions of Cloud computing, it is possible to delegate computing and data on desktops and portable devices to large data centers, called cloud services [5].

There are several key challenges associated with provisioning of services on clouds: service discovery, monitoring, deployment of Virtual Machines (VMs) and applications, and load-balancing among others [6]. In particular, the resource requests for cloud services change over time and the computing resources can be allocated dynamically upon the requirements and preferences of consumers. Therefore it is not easy to dynamically and efficiently allocate the cloud resources because consumers may access applications and data in the "Cloud" from anywhere at any time and different users have conflicting preferences [7]. Additionally, although Cloud has been increasingly seen as the platform that can support elastic applications, it faces certain limitation pertaining to core issues such as ownership, scale, and locality. For instance, a Cloud can only offer a limited number of hosting capability to application services at a given instances of time: hence, scaling the application's capacity beyond a certain extent becomes complicated [6]. Therefore, traditional system-centric resource management architecture cannot process the resource assignment task and dynamically allocated the available resources in a cloud computing environment [7].

For these reasons, effective and efficient resource allocation is a core and challenging issue in Cloud computing, and a next-generation technology requires flexible resource management such as monitoring current service requests and adjusting schedules, prices, and amount of allocated resources for service requests [2].

1.2 Challenges and motivation

Recently, resource allocation mechanisms are developed or analyzed for each service types of Cloud computing, Software-as-a-Service (SaaS) [8], Infrastructure-as-a-Service (IaaS) [9], and Platform-as-a-Service (PaaS) [10]. These researches cannot represent the conflicting relationship among users or the requested Cloud services from users. In order to consider the conflicting relationship of Cloud computing, it seems that a game theoretical approach is needed for developing efficient resource allocation mechanism.

Great deals of researches have proposed game theoretic resource management mechanisms for Grid computing [11-20], and a few researches for Cloud computing [4, 21-23]. However, there are great differences between the Grid computing and Cloud computing in terms of basic objectives, characteristics and the way of assigning tasks and allocating resources [24]. Hence, it is difficult to directly apply the prior mechanisms which are developed for the traditional grid computing to cloud computing. In addition, most of prior game theoretic approaches in both Grid computing and Cloud computing have adopted the efficient and effective scheduling or discriminative mechanisms like auctions. Thus, it seems to be difficult to guarantee the vital characteristics of Cloud computing, on-demand service provisioning, and also to correspond to the dynamic change of users' requests. Consequently, for the efficient and effective Cloud service providing, a new approach for task assignment and resource allocation should be developed to guarantee the on-demand service provisioning as well as the high level of system performance, which can be evaluated with general quantitative metrics such as average lead time, network bandwidth, completion time (make-span), task costs, reliability and fairness [7, 25]. These requirement of performance evaluation can be found in the increased importance of Application Performance Management (APM) of Cloud computing. As more applications are pushed on the cloud, monitoring performance and measuring the SLAs for application performance in cloud [26]. The software packages such as AppDynamics, Koscom, Ovum, and CA APM provide the function of performance evaluation in the view point of users as well as monitoring over the whole resources including infrastructure, network and application for Cloud services.

As included in the pool of metrics of performance evaluation and Ergu, *et al.* and Chandak, *et al.* have pointed [7, 25], the fairness should be included in the pool of metrics for performance evaluation. Because users pay costs according to the amount of resource usage in Cloud computing, the effectiveness and efficiency of resource usage become the primary metrics in cloud users' position.

In addition, resource allocation and task assignment mechanisms in the prior researches have been developed in order to make service provider's utility or profit higher, not considered the users. Thus, in order to keep the better continuity of business, a Cloud service provider should be able to guarantee not only higher efficiency (or effectiveness) but also relatively higher fairness compared with other users or past service using experiences.

1.3 Contribution

The ultimate goal of this paper is to propose an efficient resource allocation mechanism in Cloud computing environment, which can enhance the fairness of resource utilization while considering the conflicting relationship among the requested Cloud services.

In order to accomplish the ultimate goal, we adopt a bargaining solution, Nash bargaining solution (NBS), which is well known to guarantee the fairness based on the utility among players in competition, for task assignment and resource allocation in cloud computing services. With the proposed mechanism, it seems to be possible to effectively and efficiently utilize limited computing resources while guaranteeing the on-demand cloud service provisioning.

We also present and analyze numerical experiments to show that the proposed mechanism guarantees better performance than existing methods such as proportional share (PS), Auction and the equal rate allocation system (ERAS). From the result of analysis, it has been shown that the proposed mechanism using NBS guarantees the shortest average lead time and make-span than others, and provides a relatively higher fairness of utility-cost ratio.

The proposed mechanism, NBS, can be applied to general cloud services such as SaaS, IaaS and PaaS, and hybrid services which are combined with general services by modeling cloud resources such as CPU, memory, physical disk space and bandwidth as general cloud resources. It also supports cloud service providers in flexibly managing their limited resources in a cloud computing environment in terms of quality of service.

The remainder of this paper is organized as follows: We introduce related work in Section 2. The overall procedure of the proposed mechanism is explained in Section 3, and the detailed algorithms for task assignment and resource allocation are then described in Section 4. We illustrate the performance of the proposed mechanism with numerical experiments in Section 5. Finally, we conclude the paper in Section 6.

2. Related Work

2.1 Game theoretic resource allocation mechanisms

Resource allocation has been a research topic and key factor in distributed computing and grid computing [23, 27-30]. Various game theoretic solutions have been adopted in settings of both Cloud and Grid computing [22]. Generally, the game theory concerns the necessity of decision making in a certain situation where by two or more rational opponents are being occupied under conditions of competition along with conflicting interests in expectation of definite outcomes in excess period of time [31]. Game-theoretic resource allocation mechanisms are categorized based on the types of models such as general equilibrium, bargaining and negotiation, bid based proportional sharing, and auction. Especially, Buya et al., have introduced various economic approaches such as commodity market model [32], posted price model, bargaining model, tendering/contract-net model, auction model, bid-based, proportional resource share model [14]. In this section, the game theoretic task assignment and resource allocation mechanisms for Grid computing and cloud computing are summarized at first. And then, task scheduling models and other issues related resource management are discussed.

General equilibrium: The resource allocation approaches applying economics in Grid computing have been covered in many researches [13, 14, 33-37]. The majority of them investigated the economy in general equilibrium, which can be converted to the Nash equilibrium (NE) [31]. Yazir et al. [38] proposed a new approach for dynamic autonomous resource allocation in computing clouds through distributed multiple criteria decision analysis. Lee and et al. (2010) suggested an evolutionary game theoretic mechanism, which is called Nuage, allows applications to adapt their locations and resource allocation to the environmental conditions in a cloud [39]. Bayesian Nash Equilibrium Allocation algorithm to solve resource management problem in cloud computing has been developed by Teng and Magoulès. They have been shown that cloud users may receive Nash equilibrium allocation solutions according to the gambling stages, and the resource will converge to the optimal price [40]. Wei and et al. (2010) developed a stepwise mechanism for a QoS constrained resource allocation problem in cloud computing. Their game theoretic solution is obtained by two steps. First, each participant solves its optimal problem independently, and then evolutionary mechanism which takes both optimization and fairness into account is designed [23].

Auction: Auction is the most popular resource allocation mechanism when users or tasks compete for limited computing resources in Cloud computing as well as Grid computing because it can describe the negotiation process well and is easy to understand and implement. Lei and et al. (2008) proposed an auction based resource allocation mechanism using prediction of others' bid based on the mean value of historical bids [16]. Auction model using SimGrid framework [15], combinatorial auction based resource allocation protocol [11], and the Continuous Double Auction [18] have been adopted to resolve some ineffectiveness of grid resource allocation. Different from these mechanisms using only a singular price and match, there exists a research which adopted more sophisticated auction protocol [19]. Also, decentralized auction mechanism has been adopted in resource accounting systems such as POPCORN [17], Spawn [20], and CPM [12]. Sun et al. (2012) has developed a resource allocation model, ESPSA, based on the second price auction for both grid and cloud computing environments. This model introduced a bidder number restriction method to assure the victorious probability of each users and resource brokers [22]. Also, they developed a heuristic resource allocation algorithm based on a continuous double auction and M/M/1 queuing model to improve both performance-QoS and economic-QoS simultaneously [4].

Bargaining and negotiation model: In the bargaining model, resource brokers bargain with Grid Service Providers (GSPs) for lower access prices and higher usage durations. Both brokers and GSPs have their own objective functions and they negotiate with each other as long as their objectives are met. This negotiation is guided by user requirements and brokers can take risks and negotiate for cheaper prices as much as possible, and they can discard expensive machines. However, this might lead to lower resource utilization, so GSPs generally employ this model when market supply and demand and service prices are not clearly established [14]. An et al. devised a negotiable contract model with commitment price and de-commitment cost between the cloud service provider and customers, which the agents negotiate to generate contracts for resource leases for a fixed time interval [21]. Wei et al. proposed a two-stage optimization algorithm using game theory [23].

Bid based proportional sharing: Resource allocation models using bid-based proportional resource sharing are quite popular in cooperative problem solving environments such as clusters (in a single administrative domain). In this model, the percentage of resource share allocated to the user application is proportional to the bid value in comparison to other’s bids [14]. Examples systems such as Rexec/Anemone, Xenosevers and D’Agents CPU market employ a proportional resource sharing model in managing resource allocations [13].

Task scheduling: Efficient task scheduling mechanism can meet users’ requirements and improve the resource utilization [41]. An evolutionary mechanism and game theoretic method have been developed to find the optimal schedule of dependent cloud computing services based on the simulation approach [42]. By applying game theoretic approaches, most of resource allocation scheduling model have considered the relationship between users and grid resources, not the interaction among users [43]. Especially, Ergu, et al.

suggested a task oriented scheduling mechanism using Analytic Hierarchy Process (AHP) in Cloud computing [7]. They set the rank on the cloud tasks based on the results of pairwise comparison among dynamically requested tasks and user preferences for evaluating performance of resource allocation, which rank values are utilized as the criteria for task scheduling and resource allocation.

Other issues: In addition, the issue of determining how many hosts or resources are adequate for maintaining stable services while satisfying obligations [44], and autonomic detecting and dynamic resolving models (scaling up or down) for bottlenecks in a multi-tier Web application hosted on a cloud based on response time [45] have been introduced.

There are some different features in the process of assigning tasks and allocating resources between the Cloud computing and other paradigms. First of all, for grid computing, the resources are highly unpredictable, heterogeneous, and their capabilities are typically unknown and changing over time, which may connect and disconnect from the grid at any time. Therefore, the same task is sent to more than one computer in Grid computing, and the user receives the output of the computer that completes the task first [24]. Dynamic allocation of tasks to computers is complicated in the grid computing environment due to the complicated process of assigning multiple copies of the same task to different computers [7]. However, Cloud computing enables users to access to a cloud computing environment to visit their data and obtain the computation at anytime and anywhere [46]. In addition, Cloud computing is attempting to provide cheap and easy access to measurable and billable computational resources comparing with other paradigms. Hence, various task assignment and resource allocation mechanisms cannot be directly applied to Cloud computing without modification.

2.2 Limitations

Despite a variety of studies, some limitations to develop efficient and effective resource management mechanisms for Cloud computing still remain. First of all, resource management should take into consideration the essential characteristic of Cloud computing, QoS-guaranteed on-demand services, which distinguish from other computing paradigms [47]. It means that users can access cloud services just at the time when they request while providing a certain level of service quality. Most of prior researches have still adopted discriminative approaches such as auctions or market based models which guarantee the more bargaining power to certain users to increase profit of the service provider. However, task scheduling or discriminative assignment mechanisms such auction assume that the requested task can be delayed for the efficiency of the resource utilization.

It may also cause the problem that requested tasks are aborted or discarded. Note that, before a task is aborted or discarded, it consumes system resources including network bandwidth, storage space, and processing power, and thus can directly or indirectly affect the system performance [48].

Thus, it may be said that the ultimate goal of Cloud computing cannot be accomplished. In other words, the performance of the resource utilization may be evaluated based on the delay of completion time (make-span), but the starting time of task execution should not be delayed after the requested time. In addition, both the efficiency and effectiveness can be enhanced simultaneously if it is possible to prevent the resource utilization get lower while applying the bargaining model. Therefore, a dynamic and flexible resource management is required to guarantee the on-demand service provisioning.

In the most game-theoretic mechanisms adopted to solve optimization problems of competitive resource allocation and task assignment, each task is considered as an independent decision maker that competes for cloud resources [23]. These independent players can participate in the resource allocation game only when the fair mechanism is assured. The fairness of the market means that each resource owner has an equal opportunity to offer its resource and it can obtain a fair profit according to its capability [49]. Although the fairness of resource allocation should be considered as the criteria for the performance evaluation like response time, compete time and reliability, only two models, commodity market model [32] and incentive based scheduling [49] in Grid computing have tried to minimize the fairness deviation among resources. It is hard to find researches which consider the fairness in Cloud computing.

Motivated by these considerations, we have devised a task assignment and resource allocation mechanism using the bargaining solution, Nash Bargaining Solution (NBS) to guarantee the on-demand service provisioning as well as the fairness of resource allocation which is evaluated with utility-cost ratio.

3. Proposed Mechanism

In this section, we describe the overall procedure of the proposed task assignment and resource allocation mechanism using NBS. The system architecture of the proposed mechanism focuses on the procedure of Cloud service execution, as shown in Figure 1. The system to provide cloud service is generally composed with four different layers: user interface layer, service management layer, service execution layer and resource virtualization layer.

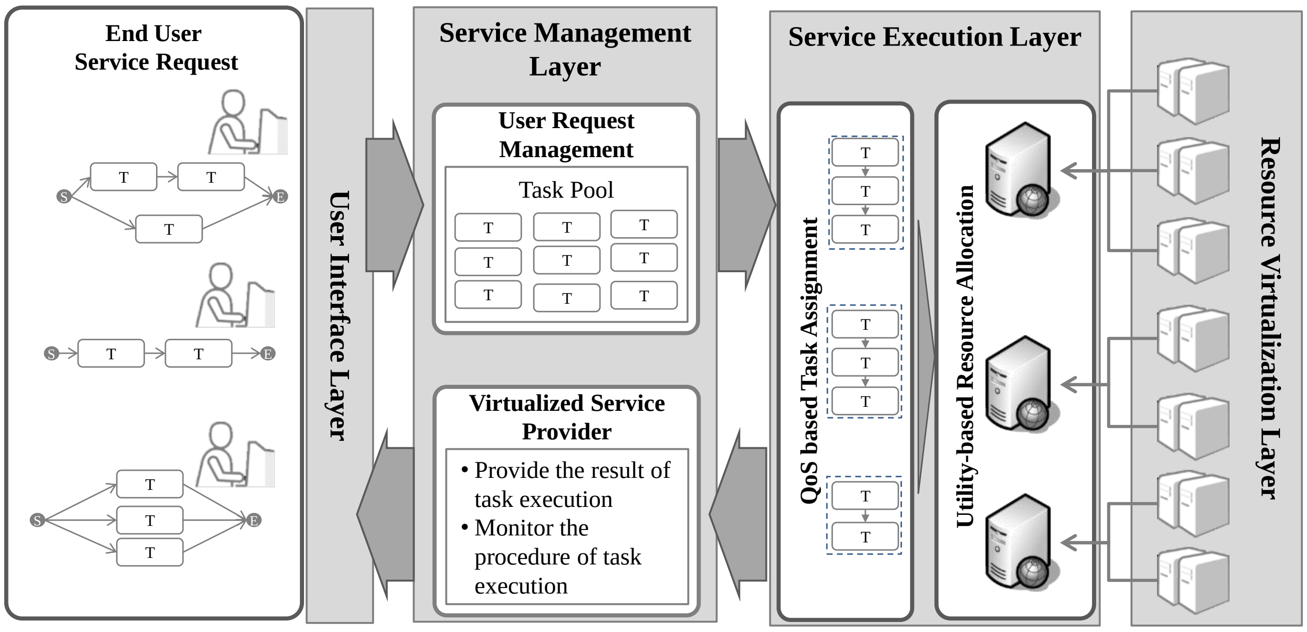

End User
Service Request
S
T
T
E
T
User Interface Layer
Service Management Layer
User Request Management
Task Pool
Virtualized Service Provider
• Provide the result of task execution
• Monitor the procedure of task execution
Service Execution Layer
QoS based Task Assignment
Utility-based Resource Allocation
Resource Virtualization Layer

Figure 1. Architecture of a cloud computing system

First of all, the Cloud service provider like Google, Amazon, and Salesforce.com designs and deploys its own services in the service management layer. This cloud service may be composed with one or more unit services. Each unit service can be characterized with the minimum requirement of resource and unit price. Like conventional Web-based services, users request the service provider to execute Cloud services through the user interface layer. After user requested Cloud services, a new instance of requested unit Cloud service is created, called a task. The properties of a task follow those of the requested unit service. For example, let's assume that a Cloud service for the quotation is composed with three different unit services such as create customer, create quotation table and compare with other quotation, and it has been deployed in the Cloud system. If two different sales representatives request a quotation service, respectively, two different set composed with three predefined tasks are initiated and requested according to the flow of tasks. Then, Cloud system should execute the requested tasks and provide the results, respectively.

In the proposed system structure, these requested tasks are gathered in a task pool in the service management layer and then handed over to the service execution layer. In the service execution layer, upon the arrival of tasks, a QoS-based task management module chooses the proper distributed servers in the resource virtualization layer which can guarantee the higher level of utility (QoS). After executing the tasks, the result is provided to the users. In this paper, we focus on task assignment and resource allocation in the service execution layer, which is composed of distributed physical servers.

To simplify the task assignment and resource allocation problem without loss of generality, we assume that the cloud computing environment is characterized as follows: (1) *independent and indivisible cloud task*: a task is a minimum independent and indivisible service unit, (2) *single type of resources*: all physical service are composed with a single type of resource and all unit services require the single type of resource and (3) *Agent based automatic performance evaluation*: the device users own has an agent to communicate with the cloud system to submit requests and receive the computational result, thus users can quantitatively evaluate the QoS based on the lead-time to complete the requested tasks. The notations used in the proposed mechanism are summarized in Table 1.

Table 1 Notations and descriptions

Notation	Description
S	set of services provided in cloud computing
c_s	unit price of a resource while using service $s \in S$
R_s^0	minimum amount of resources required for executing service $s \in S$
M	set of cloud servers
Q_j	resource capacity of the cloud server $j \in M$
T	set of tasks in services requested by users, $T_i \in T$ and $T_i \in S$
c_i	unit price of a resource while executing task i, $T_i \in T$
b_i	available budget for executing task i that a user has set
R_s^0	minimum amount of resources required for executing task i, $T_i \in T$
R_i^{total}	total amount of resources requested by users, $R_i^{total}=b_i/c_s$
R_i^t	amount of resources allocated to task i at time t
X_i^t	utility of task i, a function of the amount of allocated resources at time t, $X_i^t = \pi_i(R_i^t)$

There are $|S|$ types of unit Cloud service, and the characteristics of each service are represented by the unit cost per resource (c_s) and the minimum resource amount (R_s^0). Since the task T_i is an instance of cloud service, all properties of a task follow those of the corresponding cloud service. Therefore, if a task, T_i is an instance of a service type s, the minimum requirement and unit cost of the task become equal to those of service s, ($R_i^0 = R_s^0$, $c_i=c_s$), respectively. On requesting task T_i, users set the budget (b_i) for executing the task. Here, the total resource requirement (R_i^{total}) is determined by b_i/c_i, and there are m distributed physical servers with maximum resource capacity (Q_j).

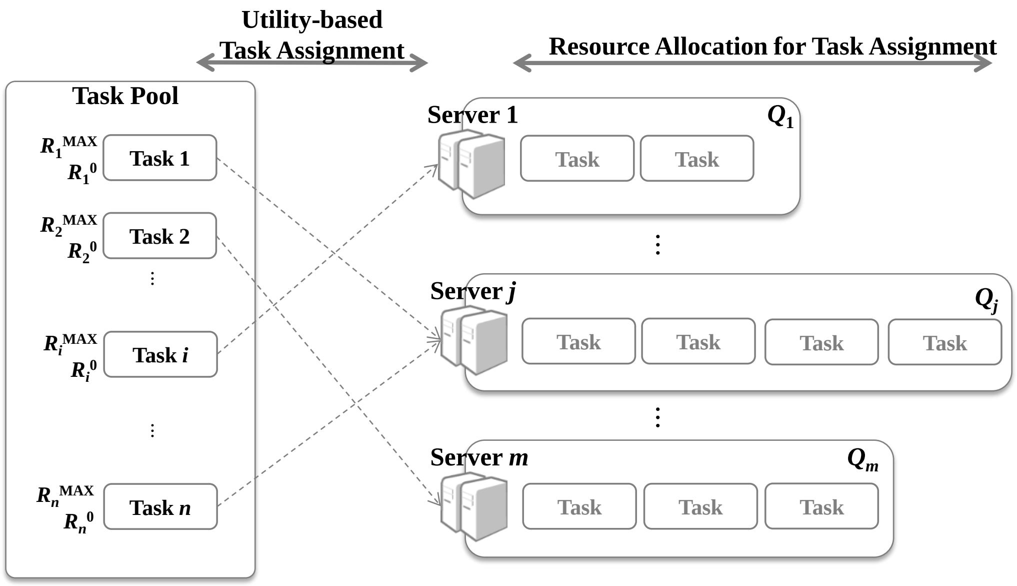


Figure 2. Description of the problem: Task assignment and resource allocation

The proposed mechanism performs task assignment and resource allocation using a bargaining solution. The overall structure of the problem is depicted in Figure 2. It is assumed that there are *m* distributed physical servers with maximum resource capacity (Q_j). The first problem pertains to which distributed server the requested cloud task is assigned to. Let x_{ij} be a binary decision variable indicating whether task *i* is assigned to cloud machine *j*. The equation, $\sum_{j \in M} x_{ij}=1$, can be established because a task is a minimum indivisible service unit. Since each server has a different resource capacity Q_j, the amount of resources allocated to the tasks is dependent on the maximum capacity of the distributed server and the requirements of other tasks that are assigned to the same server. The second problem is determining how to allocate resources to tasks assigned to the same server. Because the amount of allocated resources determines the utility of each task (QoS), all tasks compete with other tasks to secure as many resources as possible. These two problems should be integrated because the result of one problem has an impact on the other.

Here, the integrated problem is modeled as a game of tasks that compete for the limited resources of servers. Thus, the task assignment and resource allocation mechanism should guarantee that all tasks have the best utility. Bargaining solutions such NBS and KSBS are utilized for the resource allocation mechanism because they are known to guarantee fairness. In this paper it will be shown that the problem can have at least Nash equilibrium by proving that the problem can be transformed into a congestion game. The Elementary Stepwise System (ESS) is also proposed to guarantee that the Nash equilibrium can be reached in polynomial time.

Algorithm 1. Task assignment and resource allocation

```
Input:
    a task set, $T$ ($\ni T_i$) and the total requirement of each task ( $R_i^{total}$)
    a cloud servers set, $M$ ($\ni M_j$) and maximum amount of resources required for each server ($Q_j$)
    task sets requested at time $t$, $N^t$ ($\subseteq T$), $0 \le t \le t^{MAX}$
    Set $t$=0
Repeat
    Set the status of $T_i \in N^t$ to 'on'
    Randomly assign tasks whose status are 'on' ($\sum_t R_i^t < R_i^{total}$ ) to cloud servers
    Update the maximum requirement of tasks, $R_i^{MAX} = R_i^{total} - \sum_t R_i^t$

    Repeat at most ($n$-1) times {
        1) Allocate resources ($Q_j$) to tasks that are assigned to server $j$ using Resource-Allocation ($R^{MAX}$, $c$, $Q_j$) for
           all $j$ ($\in M$)
        2) Calculate utilities ($X_i$) for the allocated resources
        3) Create an ordered queue of tasks SEQ in descending order of the utility gaps, $X_i^{MAX} - X_i$
        4) Update the task assignment using ESS (described in Algorithm 2) with SEQ and Resource-Allocation
           ($R^{MAX}$, $c$, $Q_j$)
    }
    set the status of tasks to which all required resources are allocated ($\sum_t R_i^t = R_i^{total}$) as 'off'
    $t \leftarrow t+1$
Until $t \le t^{MAX}$

Resource-Allocation ($R^{MAX}$, $c$, $Q_j$){
    $R_i$ =0;
    while ($\sum_i R_i = Q_j$ or $R_i = R_i^{MAX}$ , for all $i$) {
        $R_i = R_i$+ amount of resources allocated to task $i$ using one of the resource allocation methods;
        $Q_j = Q_j - \sum_i R_i$;
        $R^{MAX} = R^{MAX} - R_i$;
    }
    return $R$;
}
```

The ultimate goal of this research is to develop a task assignment and resource allocation mechanism to enhance the utility of all requested tasks. This goal can be achieved by *Algorithm 1*, which describes the overall procedure of assigning tasks to distributed servers that can guarantee the better utility. The utility of task i, $X_i = \pi_i(R_i)$, is calculated based on the amount of allocated resources. In this algorithm, once tasks are assigned to a distributed server, resources are allocated using resource allocation mechanism until all amounts of resources in a server are consumed or the maximum requirements are met of all assigned tasks. And, the sequence of server change is then determined based on the gap between the maximum and current utilities (X_i^{MAX} - X_i). Each task changes its server choices if another server guarantees greater utility than the current one. This procedure is repeated until an equilibrium status is reached.

The procedures and criteria for resource allocation which is described in the function, *Resource-Allocation* (R^{MAX}, c, Q_j), vary with the different resource allocation mechanisms. Details of the resource allocation framework are given in the following sections.

4. Utility-Based Task Assignment and Resource Allocation

In this section, the utility function of a task is first devised considering the characteristics of cloud computing service. Then, the task assignment and resource allocation framework based on utility are described in detail.

4.1 Utility function of a task

A utility function is the criterion used for task assignment and resource allocation in this research. As depicted in Figure 1, Cloud services are continuously and dynamically requested from users, and distributed servers should execute the requested tasks and present the result to users as soon as possible. Meanwhile, users do not recognize which cloud server will take their tasks and how many resources are allocated to the tasks. Thus, users can evaluate the quality of services based only on how their tasks are executed. In this paper, we present a utility function inspired from the concept of response time in order to consider the fundamental characteristics and relationship between the Cloud system and users. In practice, prior researches [50-52] have shown that the response time is dependent on the amount of allocated resource and user’s utility function in Cloud computing has the inverse relationship with the response time.

Figure 3A shows response time as a function of the amount of allocated resources. As the amount of resources increases, the response time decreases and gets close to a certain point (D_i). From this point of view, the utility, which is defined as the value a user places on the cloud services according to the amount of allocated resources, can be illustrated as shown Figure 3B. As the amount of allocated resources increases, the marginal utility decreases.

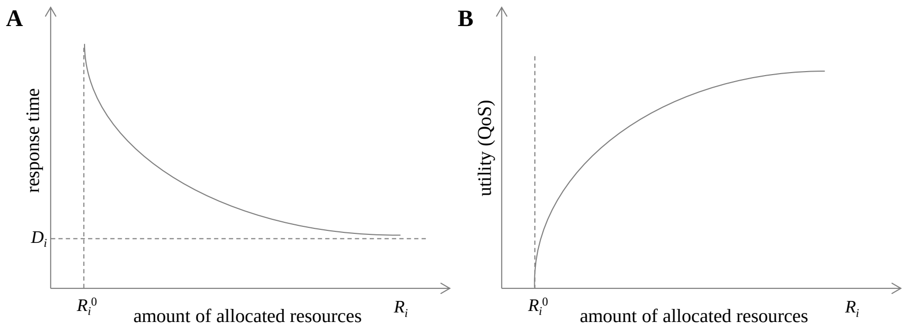


Figure 3. Response time and utility (QoS) vs. amount of allocated resources: A, response time; B, utility

By reflecting the fact that utility is inversely proportional to response time, we devised a generalized utility function as follows:

$$QoS_i\left(R_i\right)=\frac{\kappa_i\left(R_i-R_i^0\right)}{\left(D_i\left(R_i-R_i^0\right)+\omega_i\right)}, \tag{1}$$

where R_i is the amount of allocated resources, and ω, k, R_i^0 and D_i are parameters dependent on service type. Note that k_i and ω_i are positive and D_i is non-negative. Here, the amount of allocated resource (R_i) for task i should be equal to or greater than the minimum requirement (R_i^0). If R_i is less than R_i^0, the QoS becomes zero. Therefore, the minimum requirement can be interpreted as the disagreement point or the minimal requirement for guaranteeing the performance isolation.

4.2 Utility-based task assignment

In this study, task assignment is performed based on the utility derived from the amount of allocated resources. If the assignment of multiple tasks can change simultaneously, the resource allocation plan cannot be achieved [53-55]. Therefore, we assume that only one task can change the server selection at a time, and we implement the framework with ESS. It has been proven that ESS converges to Nash equilibrium [54] and that the required number of task assignments for n tasks to reach Nash equilibrium is at most n-1, which is linearly proportional to the number of tasks [56].

In the remainder of this subsection, we describe how to apply ESS to our problem. The order in which to change the server selection is determined based on the utility gaps of tasks, $\rho_i(x_{ij})=X_i^{MAX}-X_i^*$, which is defined as the difference between the maximum achievable utility and current one. Hence, the bigger utility gap a task has, the earlier the task changes the server selection. The procedure of the proposed ESS is described in *Algorithm 2*.

***Algorithm 2. ESS* for task assignment**

```
Input:
    Ordered queue of users, SEQ
    task assignment $x=(x_{i\cdot})$, $\forall i \in SEQ$
Repeat
    1) Retrieve a task with the largest utility gap in SEQ. u=SEQ.pop()
    2) Suppose task u is assigned to cloud server w (i.e., $x_{uw}=1$).
       If another cloud server can offer greater utility than w, update the task assignment of u.
       $s^* = \arg\max_{j \in M} \left( \pi_u \left( R\left( x_{uj} \right) \right) \right)$
       if $\pi_u \left( R\left( x_{uw} \right) \right) < \pi_u \left( R\left( x_{us^*} \right) \right)$,   $x_{uw} \leftarrow 0$ and $x_{us^*} \leftarrow 1$.
Until SEQ is empty
```

4.3 Resource allocation using bargaining game solutions

Here, the resource allocation framework using a bargaining solution, NBS, is explained under the assumption that all tasks are assigned to servers.

4.3.1 Bargaining power and feasible solution

The bargaining power of tasks should be defined before developing resource allocation mechanism using the NBS. Generally, since bargaining power is used to represent the discriminated priority to the resource, the bargaining power is defined based on the unit price of the cloud service that is instantiated by a task as follows:

$$\alpha_i = \frac{1}{(n \cdot \xi + 1)} \left(\xi + \frac{c_i}{\sum_{j=1}^{n} c_j} \right). \tag{2}$$

The service provider can adjust gaps in bargaining power among tasks using the parameter, ξ (≥ 0).

4.3.2 Nash bargaining solution (NBS)

Only one generalized NBS $X^*=(X_1^*, \ldots, X_n^*)$ exists that satisfies the six axioms explained in [57]. Additionally, the generalized NBS is the solution of the following optimization problem [58, 59]:

$$\begin{aligned} X^* &= \arg\max_{X\in S} G(X) = \prod_{i=1}^{n} (X_i - d_i)^{\alpha_i} \\ & s.t\,(X_1,\ldots,X_n)\in S,\ d_i \le X_i, \\ & \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \ge 0, \text{for all } i. \end{aligned} \tag{3}$$

where $(X_1, \ldots, X_n)$ is the joint utility point in the feasible solution, **S**, defined in 4.3.1 with bargaining power α. In this study, the disagreement point, $\mathbf{d} = (d_1, \ldots, d_n)$ is assumed to be the origin. For any optimization problem with differentiable objective and constraint functions for which strong duality holds, any pair of primal and dual optimal points must satisfy the Karush-Kuhn-Tucker (KKT) conditions. Moreover, if the primal problem is convex, the KKT conditions are sufficient for the points to be primal and dual optimal [60]. If it is possible to assume that $G(X)$ is a concave function and **S** is a convex set, the primal problem is convex [57]. In addition, if $G(X)$ is a non-decreasing function, the optimal solution should be in the bargaining set **B**, which is a set of all individually rational, Pareto optimal utility pairs in a feasible solution, **S** [61, 62]. Thus, the optimization problem can be expressed as follows:

$$\begin{aligned} X^* &= \arg\max_{X\in S} G(X) = \prod_{i=1}^{n} X_i^{\alpha_i} \\ & s.t, \sum_{i=1}^{n} R_i = \sum_{i=1}^{n} \frac{\omega_i X_i}{(\kappa_i - D_{0i} X_i)} = Q - \sum_{i=1}^{n} R_i^0 \\ & X_i > 0, \text{for all } i \end{aligned} \tag{4}$$

Since the objective function and constraint functions are differentiable, the following KKT conditions are established [57]:

1) *Primal constraints*: $\sum_{i=1}^{n}\left(\omega_i X_i / (\kappa_i - D_{0i} X_i)\right) = Q - \sum_{i=1}^{n} R_i^0$, $X_i>0$ for all i;

2) *Dual constraints*: $\lambda \ge 0$;

3) *Complementary slackness*: $\lambda_i X_i = 0$ for all i;

4) The *gradient of the Lagrangian with respect to X vanishes*: $(\triangledown G(X))_i - \lambda_i + \nu(\kappa_i \omega_i / (\kappa_i - D_{0i} X_i)^2) = 0$ for all i;

where $(\bigtriangledown G(X))_i$ represents the *i*th element in the vector of $\bigtriangledown G(X)$ and λ_i and ν are Lagrangian multipliers associated with the *i*th inequality constraint and equality constraint, respectively. The gradient of the objective function is as follows:

$$\nabla G(X)=\left[a_1 X_1^{\alpha_1-1}\prod\nolimits_{i=2}^{n} X_i^{\alpha_i},\ldots,a_n X_n^{\alpha_n-1}\prod\nolimits_{i=1}^{n-1} X_i^{\alpha_i}\right]^T. \tag{5}$$

The first condition is satisfied because the generalized NBS is a point in the bargaining set, **B**. The third condition (complementary slackness) is satisfied by setting $\lambda_i=0$ because $X_i>0$ for all i due to the assumption that a higher utility than the disagreement point is allocated to each task. Hence, the second condition (dual constraints) is obviously satisfied. From the last condition, the gradient of the Lagrangian can be rewritten as $(\bigtriangledown G(X))_i - \nu(\kappa_i\omega_i/(\kappa_i-D_{0i}X_i)^2)$ for all *i*. Since ν is a constant, the relationship between X_m and X_k is established by setting *m*th and *k*th rows to be equal as follows:

$$\frac{\alpha_m\left(\kappa_m-D_{0m}X_m\right)^2}{\omega_m X_m}=\frac{\alpha_k\left(\kappa_k-D_{0k}X_k\right)^2}{\omega_k X_k}. \tag{6}$$

By applying this equation, the NBS can be easily obtained if one of the points can be fixed. When assuming the utility of the first task is obtained, the utilities of the remaining tasks can be determined by using the following equation:

$$R_k=\begin{cases}\dfrac{-\alpha_1\omega_1\omega_k+\sqrt{\left(-\alpha_1\omega_1\omega_k\right)^2+4\alpha_1\alpha_k\omega_1\omega_k D_{0k}\left(R_1-R_i^0\right)\left(D_{01}\left(R_1-R_i^0\right)+\omega_1\right)}}{2\alpha_1\omega_1 D_{0k}} & \text{if } D_{0k}\neq 0\\[2ex] \dfrac{\alpha_k\left(R_1-R_i^0\right)\left(D_{01}\left(R_1-R_i^0\right)+\omega_1\right)}{\alpha_1\omega_1}+R_k^0 & \text{if } D_{0k}=0, k=2\ldots n\end{cases} \tag{7}$$

Here, the bisection method [60] is applied to the optimal solution because it can be used to obtain the solution in the *n*-th polynomial time. By adopting the upper bound ($u=Q$) and the lower bound ($l=R_i^0$) of resource allocation, the bisection method can be easily applied. The method then requires exactly $\lceil log_2((u-l)/\varepsilon)\rceil$ iterations [57].

5. Illustrative Experiments

In this section, the results of numerical experiments are presented to illustrate how the proposed task assignment and resource allocation framework works and the superior performance of the mechanism compared to other resource allocation frameworks.

5.1 Experimental design

A few assumptions for experimental design such as the type and the amount of resources are chosen from the CloudSim model [6] which characterizes and simplifies the general circumstances of Cloud computing. Other assumptions and parameters for the experiments are artificially generated to illustrate how the proposed mechanism works. Especially, the parameters for the utility functions, D_{0i}, w_i, and k_i, can adjust the characteristics of utility functions, such as the maximum level, the velocity of convergence, and sensitivity to the amount of allocated resources. Thus, the values are selected to illustrate the environment of general cloud computing by utilizing the minimum response time of a famous cloud service, Google Calendar, and the maximum tolerable waiting time. The parameters for the experiments are summarized in Table 2.

Table 2. Parameter settings for numerical experiments

Parameters for experiments		Parameters for cloud service types											
Parameters	Values	no.	c	D_0	ω	κ	λ	no.	c	D_0	ω	κ	λ
M	10	1	1.0	0.01	4.9	10.0	0.5	6	1.2	0.03	5.1	10.2	2.0
Q	4GBytes RAM	2	1.1	0.01	5.1	10.1	0.5	7	1.3	0.03	5.2	10.0	1.0
T	$\lvert T\rvert$=1,000	3	1.1	0.02	5.0	10.2	1.0	8	1.3	0.03	5.3	10.1	1.0
R_i^0	0, $\forall i \in T$	4	1.2	0.02	5.1	10.0	1.0	9	1.3	0.04	5.2	10.2	0.5
S	$\lvert S\rvert$=10	5	1.2	0.02	5.2	10.1	2.0	10	1.4	0.04	5.3	10.0	1.0

In the experiments, to make the experiments simple, it is assumed ten types of Cloud service, which each service has different parameters for utility function and unit price for single type of resource. The minimum requirement for all service type is assumed as zero, and the maximum requirement is determined based on the level of budget that users have randomly set for each service request and the unit price of requested service type ($R_i^{MAX} = b_i/c_i$). This maximum requirement is continuously updated according to the result of resource allocation as in Equation (8). In the equation, t is the index of iteration time. If the maximum requirement is satisfied at the time t, ($R_i^{MAX,t}=R_i^t$), the requested task, T_i, is terminated.

$$R_i^{MAX,t} = R_i^{total} - \sum_{\tau<t} R_i^{\tau} \tag{8}$$

In order to quantitatively evaluate the performance of task assignment and resource allocation, an artificial workload model which services are randomly requested, at first. We have developed a workload model by using random arrival process which follows a Poisson distribution with arrival rate, λ. This arrival rate of each service type has its own value. However, in order not to make characteristics of service types have impacts on the result of simulation, the total number of service requests for all service types is equally set to be one hundred. Consequently, a thousand tasks are randomly requested. The simulation will be terminated when the all tasks are terminated. Also, we assumed that there are ten distributed servers, that each server has 4GBytes of RAM memory.

By synthesizing the results of repeated simulation with these assumptions and settings, it may be able to evaluate the efficiency and effectiveness of proposed task assignment and resource allocation mechanism when cloud servers are crowded with tasks.

5.2 Other resource allocation mechanisms

The performance of the proposed framework in allocating resources is evaluated in comparison with other resource allocation frameworks, including the ERAS, PS and PA. On-demand service provisioning means that all tasks should be executed as soon as requested. In other words, the service provider should allocate resources to all requested tasks. The resource allocation mechanisms—NBS, MRPS, CBPS and ERAS—are able to guarantee the on-demand provision of service. PA guarantees on-demand service provisioning only to tasks with higher bid prices, while making the other tasks wait for their services until resources are available. Even though PA cannot guarantee on-demand service provisioning to all tasks, it has been included to compare the other metrics' agility and efficiency.

ERAS is the most simple and intuitive resource allocation framework, which is supposed to guarantee fairness by allocating exactly equal amounts of resources to all tasks remaining in the system. Here, to avoid allocating more resources than required, the amount of resources allocated to task *i* at time *t* is determined by Equation (9).

$$R_i^t = \min\left(\frac{Q}{n^t}, R_i^{MAX,t}\right) \tag{9}$$

In this equation, n^t is the number of tasks remaining in the system at time t while applying the ERAS framework.

Different from ERAS, proportional share proportionally allocates resources according to the weights of each task. Two different proportional share mechanisms, maximum requirement-based proportional share (MRPS) and cost-based proportional share (CBPS), are adopted in the experiments. In these two mechanisms, the amount of allocated resources is determined according to the maximum resource requirements and costs, as calculated in Equations (10) and (11), respectively.

$$R_i^t(x_{ij}) = \frac{R_i^{MAX,t}}{\sum_{j=1}^{n_j^t} R_j^{MAX,t}} Q_j \tag{10}$$

$$R_i^t(x_{ij}) = \min\left(\frac{c_i}{\sum_{j=1}^{n_j^t} c_j} Q_j, R_i^{MAX,t} \right) \tag{11}$$

For a PA, a task can be interpreted as the bidder at an auction, and the unit price of service type and the maximum resource requirement at each time are regarded as the bid price and quantity, respectively. In a bid profile $\Phi=(\Phi_1,\ldots,\Phi_n)$, a task bid Φ_i means that task i wants to purchase a quantity $R_i^{MAX,t}$ at the unit price, c_i. The auctioneer adopts an auction rule A to determine an allocation $A(\Phi)=(R^t(\Phi), TC(\Phi))$, where $R_i^t(\Phi)$ and $TC_i(\Phi)$ are the quantity allocated to and the total cost paid by task i, respectively. The resource allocation rule of PA proposed in [58] is adopted as follows:

$$Q(c_i, \Phi_{-i}) = \left[Q - \sum_{k \neq i: c_k > c_i} R_k(\Phi) \right]^+ \tag{12}$$

$$R_i^t(\Phi) = \min\left(R_i^{MAX,t}, \frac{R_i^{MAX,t}}{\sum_{k:c_k=c_i} R_k^{MAX,t}} Q(c_i, \Phi_{-i}) \right) \tag{13}$$

Equations (18) and (19) represent the fact that the resource Q_j is allocated to tasks with a higher bid price prior to tasks with a lower price. Term $Q(c_i, \Phi_{-i})$ denotes the quantity remaining after complete allocation to all tasks with higher prices than task i. $R_i^t(\Phi)$ is the revised allocation rule proposed in [63]. If the remaining quantity $Q(c_i, \Phi_{-i})$ is sufficient, $R_i^{MAX,t}$ can be allocated to task i; otherwise, the remaining resource is divided amongst task i and the tasks with the same bid price as task i. Then, the remaining resource $Q(c_i, \Phi_{-i})$ is shared proportionally to meet the maximum resource requirements of tasks, $R_k^{MAX,t}$. In the event that the remaining quantity is sufficient, the remainder will be again allocated to the tasks that have lower bid prices than task i.

5.3 Performance evaluation

For the performance evaluation, two different criteria, agility and efficiency, have been utilized. There is no doubt that agility and efficiency are the basic performance indicators for generic applications.

5.3.1 *Agility*: Average lead time and makespan

Generally, agility represents how long it takes the system to execute the requested cloud tasks. In this paper, the average lead time and makespan are utilized as the metrics for agility. The first metric, average lead time can be measured when each task is terminated because the requested time of each task is already known. *Makespan* is the measure throughput of completion time. It can be calculated as maximum of completion time [25].

Figure 4 shows the number of tasks remaining in the cloud computing system. All resource allocation frameworks show a different number of remaining tasks as time elapses. If there are more tasks remaining, it can be said that relatively more time is required to complete those tasks.

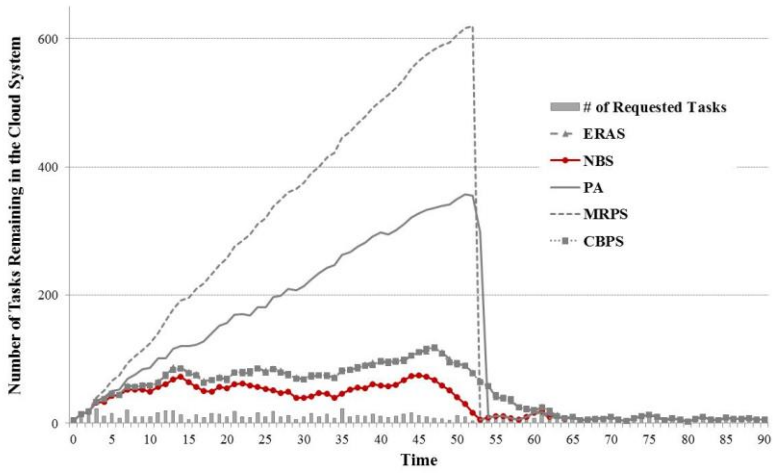
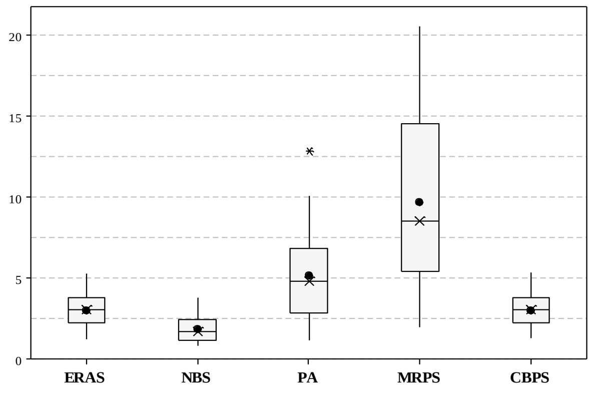
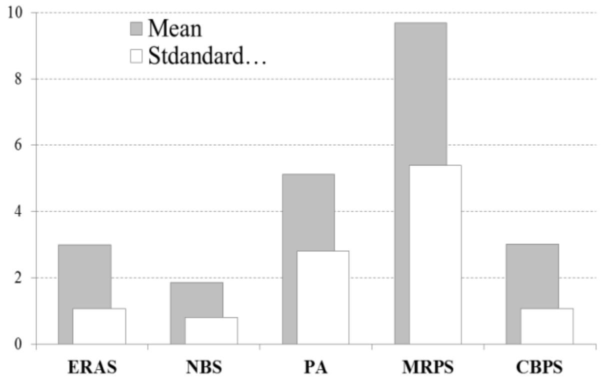


Figure 4. The number of tasks remaining in the cloud computing system

After all requested tasks are finished, the average lead time can be determined. To minimize the effect of randomness, the experiments are repeated 30 times.

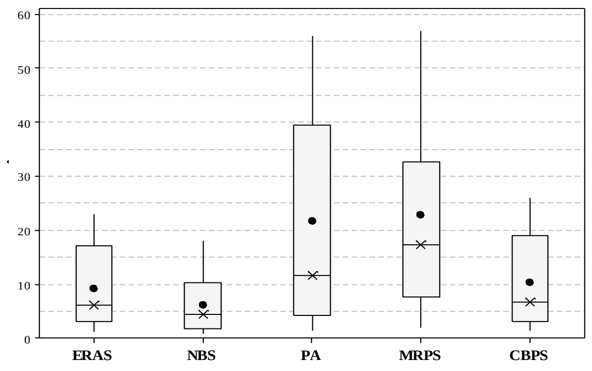
The distribution, mean and standard deviations of average lead times are summarized in Figure 5. And same information of makespan is depicted in Figure 6.

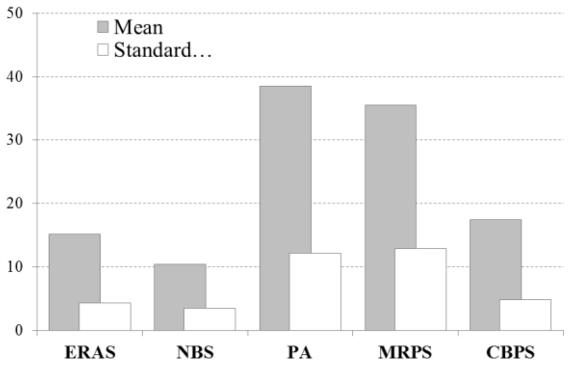




A. distribution of average lead time B. mean and standard deviation of average lead time

Figure 5. Distribution, mean and standard deviation of average lead time of the repeated experiments





A. distribution of makespan B. mean and standard deviation of makespan

Figure 6B. Distribution, mean and standard deviation of makespan of the repeated experiments

The random workload model and service type assigning make the requested time and total requirement of all tasks be randomly determined in each experiment. Thus, it is insufficient to compare agility with the simple average and standard deviation of average lead time ($\overline{LT}$) and makespan ($\overline{MS}$). For a more exact comparison, a paired *t*-test is applied because it can compare two population means of two samples by pairing observations in one sample with those in the other. With the results of paired *t*-tests, the following hypotheses can be tested.

H_0: $\overline{LT}_{NBS} < \overline{LT}_{other}$, $MS_{NBS} < MS_{other}$

H_1: $\overline{LT}_{NBS} \geq \overline{LT}_{other}$, $MS_{NBS} \geq MS_{other}$

The results of the paired *t*-tests of average lead time and makespan are summarized in Table 3 and Table 4, respectively

Table 3. Results of paired *t*-test comparisons of average lead time

Results	**NBS vs. other mechanisms**			
	ERAS	**Progressive Auction**	**MRPS**	**CBPS**
$\mu_{NBS\text{-}Other}$	-1.1399	-3.279	-7.840	-1.1660
$\sigma_{NBS\text{-}Other}$	0.3362	2.155	4.679	0.3374
SE	0.0614	0.393	0.854	0.0616
95% upper bound for mean difference	-1.0356	-2.610	-6.389	-1.0614
t-value	-18.57	-8.34	-9.18	-18.93
p-value	0.000	0.000	0.000	0.000

Table 4. Results of paired *t*-test comparisons of makespan

Results	**NBS vs. other mechanisms**			
	ERAS	**Progressive Auction**	**MRPS**	**CBPS**
$\mu_{NBS\text{-}Other}$	-3.042	-15.43	-16.63	-4.117
$\sigma_{NBS\text{-}Other}$	2.230	14.74	12.69	3.353
$SE_{NBS\text{-}Other}$	0.407	2.69	2.32	0.612
95% upper bound for mean difference	-2.351	-10.86	-12.69	-3.076
t-value	-7.47	-5.74	-7.18	-6.72
p-value	0.000	0.000	0.000	0.000

The paired *t*-test results indicate that the null hypothesis cannot be rejected. Therefore, it can be said that the resource allocation mechanism using NBS guarantees the best performance in both terms of average lead time and makespan.

In addition to bargaining solutions, each resource allocation mechanism (PA, MRPS and CBPS) shows a different result. MRPS sets the weight of each task based on the maximum requirement and allocates a certain amount of resources to all tasks. Thus, MRPS terminates most of the tasks at the same time and shows worse performance than CBPS.

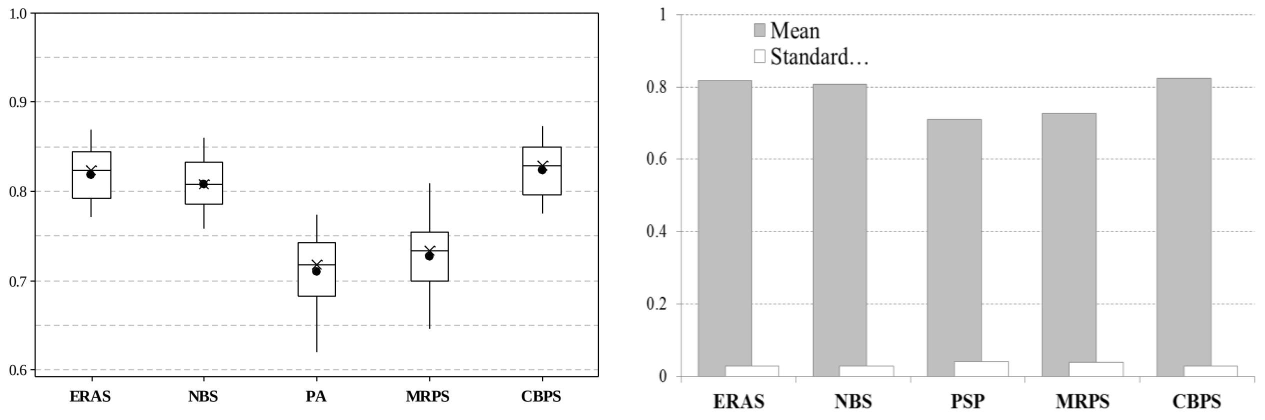
5.3.2 *Efficiency*: Fairness index

The performance of task assignment and resource allocation in Cloud computing should be evaluated based on cost and utility [64]. Thus, we have designed the metric, efficiency, to consider both cost and utility, simultaneously. QoS differentiation and adaptation is beyond the capability of systems in a traditional "best-effort" service architecture. The best-effort service model offers excellent price-performance ratios in the multiplexing of shared computing and communication resources. However, it does not allow for the simultaneous provision of excellent services and sustained high usage of shared resources [65]. Thus, the efficiency of resource allocation based on QoS is evaluated with the utility-cost ratio ($r_i^t = X_i^t / c_i R_i^t$) in this paper.

As described in the first section, the fairness of the market means that each resource owner has an equal opportunity to offer its resource and it can obtain a fair profit according to its capability [49]. Only a few previous researches [32, 66] have tried to minimize the fairness deviation for the Grid computing. It is also evaluated based on the level of fairness of the resource allocation by comparing each task with others using the fairness index, which was developed in [67] as follows:

$$FI = \left(\sum_{i=1}^{n} r_i\right)^2 / n\sum_{i=1}^{n} r_i^2 \qquad (14)$$

This fairness index can be applied to the utility-cost ratio in order to evaluate the absolute extent of user satisfaction and relative to cost. The average fairness indices of the experiments are depicted in Figure 7.



A. distribution of average fairness indices B. mean and standard deviation of average lead time

Figure 7. The mean and standard deviation of average fairness indices of repeated experiments

By referencing the Figure 7, we may intuitively expect that ERAS will show the highest *FI* value because it allocates exactly the same amount of resources to each task. On the other hand, the auction may show the lowest value due to the discriminating strategy. Instead, NBS, which guarantees the best performance in terms of agility, shows a level of fairness nearly identical to that with CBPS and ERAS.

Like the case of agility in the prior subsection, the following hypotheses are tested based on the results of paired *t*-tests.

H_0: $\overline{FI}_{other} < \overline{FI}_{NBS}$

H_1: $\overline{FI}_{other} \geq \overline{FI}_{NBS}$

Table 5. Results of paired *t*-tests comparing the fairness index

Results	**other mechanisms vs. NBS**				**ERAS vs. CBPS**
	ERAS	**Progressive Auction**	**MRPS**	**CBPS**	
μ_{X1-X2}	0.010622	-0.09768	-0.08006	0.015951	-0.005329
σ_{X1-X2}	0.002987	0.03237	0.02829	0.003353	0.000716
SE_{X1-X2}	0.000545	0.00591	0.00517	0.000612	0.000131
95% upper bound for mean difference	0.011548	-0.08764	-0.07128	0.016991	-0.005107
t-value	19.48	-16.53	-15.50	26.06	-40.77
p-value	1.000	0.000	0.000	0.000	0.000

Table 5 shows the results of the paired *t*-tests. Based on the results, we can say that NBS did not show better performance than ERAS and CBPS, but PA and MRPS. As expected, the auction shows the worst performance with regard to the fairness index. It is notable that CBPS shows the best performance in terms of the fairness index. In addition, we cannot say that ERAS which is expected to show the best performance may perform better than or even equal to CBPS. The reason of these results can be found in the characteristics of fairness index of utility-cost ratio. Since the fairness index does not compare the absolute value of the utility-cost ratio, but computes the degree of fairness with the variance, it seems to be natural that NBS cannot guarantee the highest fairness index. Although NBS always guarantees the same or better level of fairness by comparing ERAS and CBPS, it is possible to say that the performance of NBS is not far behind ERAS and CBPS when considering the mean ratio of average fairness index of NBS to ERAS (98.7%) and CBPS (98.1%).

When synthesizing the results of the performance comparison in terms of agility and fairness, we can conclude that NBS is more suitable for addressing the task assignment and resource allocation problem in a cloud computing system because it guarantees the best agility and has nearly the highest level of fairness. In other words, users will be satisfied with the rapid response time in the execution of requested tasks and will be assured that the Cloud service is fair to all users.

6. Conclusion

In this paper, we presented a game-theoretic mechanism for task assignment and resource allocation in cloud computing. While previous studies on service deployment and resource allocation generally focused on the evaluation of service quality and resource allocation based on the relationship between a cloud service provider and users or on the users' perspective, we devised a utility function based on the response time considering the characteristics of the cloud service. Then, we proposed a game-theoretic mechanism for task assignment and resource allocation using ESS and bargaining solution, NBS, based on the competitive relationship among tasks given limited cloud resources.

Through numerical experiments, we showed that the proposed mechanism can guarantee better performance than existing methods including ERAS, MRPS, CBPS and PA, in terms of agility and fairness. Applying NBS for resource allocation guarantees that the requested tasks will be terminated earlier. Also, since this method yields a higher fairness with regard to the utility-cost ratio, resource allocation using NBS is more suitable for resource management in the cloud computing system than existing methods. In other words, with the proposed mechanism, a service manager can elastically manage cloud resources in response to the dynamic changes in user requests in a cloud computing environment. Here, a limitation can be found in the characteristics of utility function. Actually, NBS can be adopted only when the convexity of utility functions can be assumed. Because the marginally decreasing utility function is the representative form as found in a great deal of prior researches, the convexity of utility function may be assumed. However, if the utility function shows different forms such as stepwise or sigmoid function, it is impossible to adopt the NBS. If the convexity cannot be assured, KSBS can substitute NBS because KSBS can find the Pareto optimal solution even though the convexity cannot be obtained while guaranteeing the similar level of performance to NBS [68, 69].

In addition, the proposed game-theoretic mechanism can be applied to any type of cloud computing services, such as IaaS, PaaS and SaaS, by the general cloud resource interpreted as CPU, memory, physical disk space and network bandwidth, and the quality of service can be considered in terms of response time. Still, some limitations remain. In the proposed approach, only a single type of resource is assumed. However, in general, two or more different types of resources, for example CPU, memory and network bandwidth, are simultaneously required in order to execute a cloud task. Thus, the response time can be devised as an aggregate function with the different resources. Therefore, the proposed mechanism can be extended to consider resource combinations and optimization.

Finally, it remains as a future work to develop better heuristic algorithms which can guarantee the benefit of applying complex optimization algorithms. Although the complexities of the proposed heuristic algorithms, bisection and ESS, were relatively low, the overhead of applying algorithms may cause the degradation of system performance while assuming the extremely large number of tasks and servers. In order to applying the proposed mechanism to practical large scale Cloud computing systems, a new alternative can be developed to secure the benefit of applying heuristic algorithms as well as the system performance.

References

1. Christy Pettey. Gartner Identifies the Top 10 Strategic Technologies for 2011. *Proceedings of Gartner Symposium/ITxpo*, October 2010. Gartner:
2. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James BrobergIvona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computing Systems* 2009; **25** (6): 599-616
3. Keqiu Li, Laurence T. YangXuemin Lin. Advanced topics in cloud computing. *J. of Net. and Com. App.* 2011; **34** (4): 1033-1034
4. Dawei Sun, Guiran Chang, Chuan Wang, Yu XiongXingwei Wang. Efficient Nash equilibrium based cloud resource allocation by using a continuous double auction. *Proceedings of 2010 International Conference on Computer Design and Application (ICCDA'10)*, June 2010. V1-94-V1-99
5. M. D. Dikaiakos, D. Katsaros, P. Mehra, G. PallisA. Vakali. Cloud Computing: Distributed Internet Computing for IT and Scientific Research. *IEEE Internet Computing* 2009; **13** (5): 10-13
6. Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De RoseRajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience* 2011; **41** (1): 23-50
7. Daji Ergu, Gang Kou, Yi Peng, Yong ShiYu Shi. The analytic hierarchy process: task scheduling and resource allocation in cloud computing environment. *The Journal of Supercomputing* 2011: 1-14
8. Fotis Aisopos, Konstantinos TserpesTheodora Varvarigou. Resource management in software as a service using the knapsack problem model. *International Journal of Production Economics* 2013; **141** (2): 465-477
9. David Villegas, Athanasios Antoniou, Seyed Masoud SadjadiAlexandru Iosup. An Analysis of Provisioning and Allocation Policies for Infrastructure-as-a-Service Clouds. IEEE Computer Society, 2012.
10. Hien Nguyen Van, Frederic Dang TranJean-Marc Menaud. Autonomic virtual resource management for service hosting platforms. IEEE Computer Society, 2009.
11. Das AnubhavDaniel Grosu. Combinatorial auction-based protocols for resource allocation in grids. *Proceedings of 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)*, 4-8 April 2005.
12. Francine BermanRichard Wolski. The AppLeS Project: A Status Report. *Proceedings of 8th NEC Research Symposium*, May 1997.
13. Rajkumar Buyya, David AbramsonJonathan Giddy. A Case for Economy Grid Architecture for Service-Oriented Grid Computing. IEEE Computer Society, 2001.
14. Rajkumar Buyya, David Abramson, Jonathan GiddyHeinz Stockinger. Economic models for resource

management and scheduling in Grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14** (13-15): 1507-1542

15. Daniel GrosuAnubhav Das. Auction-Based Resource Allocation Protocols in Grids *Proceedings of 16th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2004)*, 2004. 20-27
16. Yao Lei, Dai Guanzhong, Zhang Huixiang, Ren ShuaiNiu Yun. A Novel Algorithm for Task Scheduling in Grid Computing Based on Game Theory. *Proceedings of High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, 25-27 Sept. 2008 2008. 282-287
17. Noam Nisan, Shmulik London, Ori RegevNoam Camiel. Globally Distributed Computation over the Internet - The POPCORN Project. IEEE Computer Society, 1998.
18. Saeed Parsa, Amin ShokriSadegh Nourossana. A novel market based grid resource allocation algorithm. *Proceedings of First International Conference on Networked Digital Technologies (NDT '09)* 28-31 July 2009 2009. 146-152
19. Tuomas W. Sandholm. *Distributed Rational Decision Making*. G. Weiss (ed.) MIT Press 2000;
20. Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffrey O. KephartW. Scott Stornetta. Spawn: a distributed computational economy. *IEEE Transactions on Software Engineering* 1992; **18** (2): 103-117
21. Bo An, Victor Lesser, David IrwinMichael Zink. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. *Proceedings of Proc. 9th International Conf. on Autonomous Agents and Multiagent Systems*, 2010. International Foundation for Autonomous Agents and Multiagent Systems: 1838338, 981-988
22. Weifeng Sun, Qiufen Xia, Zichuan Xu, Mingchu LiZhenquan Qin. *A Game Theoretic Resource Allocation Model Based on Extended Second Price Sealed Auction in Grid Computing*. 2012;
23. Guiyi Wei, Athanasios Vasilakos, Yao ZhengNaixue Xiong. A game-theoretic method of fair resource allocation for cloud computing services. *Journal of Supercompting* 2010; **54** (2): 252-269
24. Ger KooleRhonda Righter. Resource allocation in grid computing. *Journal of Scheduling* 2008; **11** (3): 163-173
25. Ashish. V. Chandak, Bibhudatta SahooAshok Kumar Turuk. Heuristic Task Allocation Strategies for Computational Grid. *International Journal of Advanced Networking and Applications* 2011; **2** (5): 804 - 810
26. Satish Agrawal. APM for cloud apps: Next big challenge *InfromationWeek*. 2012
27. Yin-Fu HuangBo-Wei Chao. A priority-based resource allocation strategy in distributed computing networks. *Journal of Systems and Software* 2001; **58** (3): 221-233
28. L. Ismail, B. MillsA. Hennebelle. A Formal Model of Dynamic Resource Allocation in Grid Computing Environment. *Proceedings of Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008 (SNPD '08)*, 6-8 August 2008. 685-693
29. Ioannis MoschakisHelen Karatza. Evaluation of gang scheduling performance and cost in a cloud computing system. *The Journal of Supercomputing* 2012; **59** (2): 975-992
30. Wanneng Shu. Optimal resource allocation on grid computing using a quantum chromosomes genetic algorithm. *Proceedings of IET Conference on Wireless, Mobile and Sensor Networks, 2007 (CCWMSN07)*, 12-14 Dec. 2007 2007. 1059-1062
31. Haruna Ahmed Abba, Nordin B. ZakariaNazleeni Haron. Grid Resource Allocation: A Review. *Research Journal of Information Technology* 2012; **4** (2): 38-55
32. Gunther Stuer, Kurt VanmechelenJan Broeckhove. A commodity market algorithm for pricing substitutable Grid resources. *Future Generation Computer Systems* 2007; **23** (5): 688-701
33. Rajkumar Buyya. Economic-based Distributed Resource Management and Scheduling for Grid

Computing. Monash University, 2002.

34. Cheng Chun-TianLi Zhi-Jie. Parallel Algorithm for Grid Resource Allocation Based on Nash Equilibrium. *Proceedings of International Conference on Machine Learning and Cybernetics, 2006* 13-16 Aug. 2006 2006. 4383-4388
35. Donald F. Ferguson, Christos Nikolaou, Jakka SairameshYechiam Yemini. *Economic models for allocating resources in computer systems*. (ed.) World Scientific Publishing Co., Inc. 1996; 156-183
36. Kumaran Subramoniamt, Muthucumaru MaheswarantjMichel Toulouse. Towards a micro-economic model for resource allocation in Grid computing systems. *Proceedings of Canadian Conference on Electrical and Computer Engineering, 2002 (IEEE CCECE 2002)* 2002. 782-785
37. Chuliang Weng, Minglu Li, Xinda LuQianni Deng. An economic-based resource management framework in the grid context. *Proceedings of IEEE International Symposium on Cluster Computing and the Grid, 2005 (CCGrid 2005)* 9-12 May 2005. 542-549
38. Y. O. Yazir, C. Matthews, R. Farahbod, S. Neville, A. Guitouni, S. GantiY. Coady. Dynamic Resource Allocation in Computing Clouds Using Distributed Multiple Criteria Decision Analysis. *Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD)* 5-10 July 2010 2010. 91-98
39. Chonho Lee, Junichi Suzuki, Athanasios Vasilakos, Yuji YamamotoKatsuya Oba. An evolutionary game theoretic approach to adaptive and stable application deployment in clouds. ACM, 2010.
40. Fei TengFrédéric Magoulès. *A New Game Theoretical Resource Allocation Algorithm for Cloud Computing Advances in Grid and Pervasive Computing*. P. Bellavista, R.-S. Chang, H.-C. Chao, S.-F. Lin and P. Sloot (ed.) Springer Berlin / Heidelberg 2010; 321-330
41. Yiqiu Fang, Fei WangJunwei Ge. *A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing Web Information Systems and Mining*. F. Wang, Z. Gong, X. Luo and J. Lei (ed.) Springer Berlin/Heidelberg 2010; 271-277
42. Rajkumar BuyyaManzur Murshed. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing. *Concurrency and Computation: Practice and Experience* 2002; **14** (13-15): 1175-1220
43. David Abramson, Rajkumar BuyyaJonathan Giddy. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems* 2002; **18** (8): 1061-1074
44. Joao Nuno Silva, Luis VeigaPaulo Ferreira. Heuristic for resources allocation on utility computing infrastructures. *Proceedings of Proc. 6th international workshop on Middleware for grid computing (MGC2008)*, December 2008. ACM: 1462713, 1-6
45. Waheed Iqbal, Matthew N. Dailey, David CarreraPaul Janecek. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Future Genereration Computing Systems* 2011; **27** (6): 871-879
46. Mache Creeger. Cloud Computing: An Overview. *Queue* 2009; **7** (5): 3-4
47. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James BrobergIvona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Compter Systems* 2009; **25** (6): 599-616
48. Liu Shuo, Quan GangRen Shangping. On-Line Scheduling of Real-Time Services for Cloud Computing. *Proceedings of 2010 6th World Congress on Services (SERVICES-1)* 5-10 July 2010. 459-464
49. Xiao Lijuan, Zhu Yanmin, L. M. NiXu Zhiwei. Incentive-Based Scheduling for Market-Like Computational Grids. *Parallel and Distributed Systems, IEEE Transactions on* 2008; **19** (7): 903-913
50. Norman W. Paton, Marcelo A. T. de Aragão, Kevin Lee, Alvaro A. A. FernandesRizos Sakellariou. Optimizing Utility in Cloud Computing through Autonomic Workload Execution. *IEEE Computer Society Data Engineering Bulletin* 2009; **32** (1): 51-58
51. M. N. BennaniD. A. Menasce. Resource Allocation for Autonomic Data Centers using Analytic

Performance Models. *Proceedings of Autonomic Computing, 2005. ICAC 2005. Proceedings. Second International Conference on*, 13-16 June 2005 2005. 229-240

52. A. Menascé DanielNgo Paul. Understanding Cloud Computing: Experimentation and Capacity Planning. *Proceedings of 2009 Computer Measurement Group Conference*, 7-11 December 2009.
53. Eyal Even-Dar, Alex KesselmanYishay Mansour. Convergence Time to Nash Equilibria. *Proceedings of 30th International Colloquium on Automata, Languages and Programming (ICALP2003)*, June 2003. 502-513
54. T. Boulogne, E. AltmanO. Pourtallier. On the Convergence to Nash Equilibrium in Problems of Distributed Computing. *Annals of Operations Research* 2002; **109** (1): 279-291
55. Ariel Orda, Raphael RomNahum Shimkin. Competitive routing in multiuser communication networks. *IEEE/ACM Transaction on Networking* 1993; **1** (5): 510-521
56. Hyunggon ParkM. van der Schaar. Congestion game modeling for brokerage based multimedia resource management. *Proceedings of Packet Video 2007*, 2007. PacketVideo: 18-25
57. Hyunggon ParkM. van der Schaar. Bargaining Strategies for Networked Multimedia Resource Management. *IEEE Transaction on Siginal Processing* 2007; **55** (7): 3496-3511
58. Ken Binmore. *Fun and Games: A Text on game theory*. D.C Heath and Company: Lexington, Massachusetts, 1992;
59. Haïkel Yaïche, Ravi R. MazumdarCatherine Rosenberg. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM Trans. Netw.* 2000; **8** (5): 667-678
60. Stephen BoydLieven Vandenberghe. *Convex optimiztion*. Bambridge Univ. Press: New York, US, 2004;
61. Yiannis Andreopoulos, Adrian Munteanu, Joeri Barbarien, Mihaela Van der Schaar, Jan CornelisPeter Schelkens. In-band motion compensated temporal filtering. *Signal Processing: Image Communication* 2004; **19** (7): 653-673
62. Dov MondererLloyd S. Shapley. Potential Games. *Games and Economic Behavior* 1996; **14** (1): 124-143
63. Bruno Tuffin. Revisited Progressive Second Price Auction for Charging Telecommunication Networks. *Telecomm. Sys.* 2002; **20** (3): 255-263
64. S. M. Shatz, J. P. WangM. Goto. Task allocation for maximizing reliability of distributed computer systems. *IEEE Transactions on Computers* 1992; **41** (9): 1156-1168
65. Xiaobo Zhou, Jianbin WeiCheng-Zhong Xu. Quality-of-service differentiation on the Internet: A taxonomy. *Journal of Network and Computer Applications* 2007; **30** (1): 354-383
66. Hesam Izakian, Ajith AbrahamBehrouz Tork Ladani. An auction method for resource allocation in computational grids. *Future Generation Computer Systems* 2010; **26** (2): 228-235
67. R. Jain, D. ChiuW. Hawe. A Quantitative Measure Of Fairness And Discrimination For Resource Allocation In Shared Computer Systems. Digital Equipment Corporation, September. 1984.
68. S. Borağan Aruoba, Guillaume RocheteauChristopher Waller. Bargaining and the value of money. *Journal of Monetary Economics* 2007; **54** (8): 2636-2655
69. H. Houba, X. Tieman, R. BrinksmaTinbergen Institute. *The Nash- and Kalai-Smorodinsky bargaining solution for decision weight utility functions*. Tinbergen Institute: 1996;