International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:1, No:6, 2007

A Fitted Random Sampling Scheme for Load Distribution in Grid Networks

O. A. Rahmeh, P. Johnson, and S. Lehmann

Abstract**—**Grid networks provide the ability to perform higher throughput computing by taking advantage of many networked computer's resources to solve large-scale computation problems. As the popularity of the Grid networks has increased, there is a need to efficiently distribute the load among the resources accessible on the network. In this paper, we present a stochastic network system that gives a distributed load-balancing scheme by generating almost regular networks. This network system is self-organized and depends only on local information for load distribution and resource discovery. The in-degree of each node is refers to its free resources, and job assignment and resource discovery processes required for load balancing is accomplished by using fitted random sampling. Simulation results show that the generated network system provides an effective, scalable, and reliable load-balancing scheme for the distributed resources accessible on Grid networks.

Keywords**—**Complex Networks, Grid Networks, Load-Balancing, Random Sampling.

I. Introduction

IMPROVEMENTS in computer and networking technologies over the past decades produced dramatic increases in communication and computer capabilities. Numerous methods have been developed to maximize the use of networked computers for large-scale computing, and several protocols have been developed to efficiently utilize the distributed computing resources. Moreover, the spread of the Internet as well as the availability of powerful computers and high-speed network technologies are rapidly changing the computing landscape and society. All these technology opportunities have led to the possibility of using wide-area distributed computers for solving large-scale problems.

Large-scale computing can provide the ability to perform higher throughput computing by taking advantage of many networked computers to model a virtual computer architecture that is able to distribute process execution between the computers in the network. An example of such network system is the Grid Networks [1]. These networks use the resources of many computers connected by a network to solve large-scale computation problems.

To achieve this, there is a need for an effective load-balancing paradigm to distribute the load among the computers available on the network. When one node is overwhelmed by work, it can make use of unused computing power in the network. Therefore, implementing an integrating an efficient load distribution and resource discovery will have an essential role in implementing the self-configuring and self-optimizing characteristics of Grid networks.

The paper is organized as follows. Section II reviews related load-balancing work. Section III describes the stochastic network system. Section IV presents the mathematical analysis of the network system and provides a stationary distribution solution. Section V provides a description of network and simulation implementation. Finally, simulation results and conclusion have been discussed in section VI and section VII respectively.

Authors are with Liverpool John Moores University, Liverpool, L3 3AF, UK (Tel.: +447825415947; fax: +441512982624; e-mail: o.a.rahmeh@gmail.com).

II. Related Work

Load balancing is an active research field, and many methods and algorithms have been used to approach this problem [2-6]. The use of polling, agent-based methods, global random choice, randomized algorithms, and local diffusion methods has produced great advances in the field of load balancing [7-13]. However, most of these methods depend on central server techniques, which can be efficient in small-scale networks, or on particular properties of the load distribution in larger networks. As central servers require high computing power and large bandwidth, network systems that depend on such techniques are un-scalable [14][15]. Besides, reliability is another concern since the central server is a single point of failure.

Recently, a new specialty called Complex Networks Theory emerged, which has deep roots in statistical and non-linear physics. Complex networks theory is the field where the structural and dynamic properties of networks are analyzed. Statistical models of large systems will let systems detect or predict overall performance problems from the stream of data from individual devices.

Traditionally, complex networks have been described by *Graph Theory* [16][17]. Random graph theory was the simplest theory to describe complex network. Pál Erdős and Alfréd Rényi were the first who studied Random Graphs [16]. According to the Erdős-Rényi (ER) model, we start with N nodes and connect every pair of nodes with probability p. At the end of this process, the graph will have approximately $pN(N-1)/2$ edges distributed randomly. Therefore, the probability of having a graph with N nodes and k edges follows a Binomial distribution, and it is given by

$$P_{N,k,p}(G) = p^k (1-p)^{\frac{N(N-1)}{2}-k} \quad (1)$$

In a large random graph, there are several nodes with the same degree, and the number of nodes with a given degree can be calculated. Accordingly, in a random graph with connection probability p, the number of nodes with degree k is

$$P(k) = C_{N-1}^{k} p^k (1-p)^{N-1-k} \quad (2)$$

where C_{N-1}^{k} is the probability space in which k edges are

chosen from the total number of edges *N*-1. Thus, in ER model, the probability that an ER graph has more or less than the expected number of edges decreases exponentially. This binomial distribution implies that each node will have a degree near to the average degree, and that the number of nodes with much higher or much lower degree than average is very small. Thus, the probability that any node has the expected number of edges is the same, which gives us load balancing.

III. Stochastic Network System

For efficient usage of Grid network's resources, one would want to distribute processes as evenly as possible, so that no server is more loaded than others are. Therefore, we need to create a dynamical network system that gives balanced load distribution and efficient resource discovery.

In order to design such dynamical system, we will analyse the degree distribution of nodes in a stochastic network system with a fixed number of nodes and fixed average number of edges. A node's in-degree refers to the free resources of a node. The job assignment and resource updating processes required for load balancing are encoded in the network structure. Therefore, when a node receives a new job, it will remove one of its edges to decrease its in-degree.

Similarly, when the node completes a job, it will add an edge to itself to increase its in-degree. In steady state, the rate at which jobs arrive would equal the rate at which jobs are completed, and hence the underlying network has a fixed average number of edges. Hence, the generated graph using this protocol will be a strongly connected directed graph.

The increment and decrement of node's in-degree is performed via *Fitted Random Sampling*. Random sampling is the process of randomly picking up the nodes of a network with equal probability. The sampling starts at some fixed node, and at each step, it moves to a neighbour of the current node randomly chosen according to an arbitrary distribution.

Similar techniques have been used for load balancing which produced some significant results [11][18]. However, the proposed scheme has advantage over the previous methods in that the network structure is dynamically changed to efficiently distribute the load, and the load-balancing process will not require any monitoring mechanisms since it is encoded in the network structure.

Moreover, the number of sampling steps will be limited to a finite length, and the nodes' selection will be based on a predefined criteria rather than the last node in the walk. In this paper, *fitted* random sampling will be used where nodes' selection will depend on the free recourses (in-degree) available for each node.

Lov'asz and Winkler [19] mentioned that in undirected graph, if the random walk was long enough, then in stationary state, the probability that the walk will stop at a specific node is proportional to its stationary in-degree distribution. We found that this can be also applied to our directed graphs since the underlying network has fixed average number of edges. Therefore, fitted random sampling technique will be used in our dynamic network to provide the required load balancing, and the edge insertion and deletion strategy assures that the load will be distributed equally across all the nodes in the network.

IV. Stationary Distribution Analysis

To analyse our network system, we will consider a network system with *N* nodes, and we will assume that all the nodes in the network have similar capabilities and jobs can be executed in any node. Suppose p_k is the probability that a node has *k* edges. Then, the average number of edges, *E*, in the network is

$$E = N \cdot \sum k p_k \tag{3}$$

At each step, a randomly chosen edge will be deleted, and a randomly chosen edge will be inserted. Thus, the numbers of edges inserted and deleted are both random variables that are selected to have a fixed average number of edges.

Let *D* be the average number of deleted edges in the network, and let *M* be the maximum number of edges a node can have. To make our system compatible with ER random graphs, it is assumed that each node can have up to *N-1* edges, thus $M \leq N-1$. This assumption is not a limitation of the mechanism; it is only to show that this system is designed for large-scale networks. The expected number of edges in the network is given by

$$E = NM - D \tag{4}$$

Since the probability that a random sampling with a sufficient length will stop at a specific node is proportional to its stationary in-degree, thus, if a node's edges have been deleted uniformly randomly, then the probability that the node will lose one or more of its edges is proportional to its in-degree. Therefore, the rate at which the in-degree of a specific node will decrease is given by

$$R_k = \frac{k}{E} = \frac{k}{NM - D} \tag{5}$$

In the same way, the probability that the in-degree of a certain node will increase is proportional to the number of deleted edges from this node. Thus, the node's in-degree will increase one edge at a rate given by

$$S_k = \frac{M - k}{D} \tag{6}$$

And since the average number of edges is assumed to be fixed, we can describe this network as a Markov Chain [20] with insertion and deletion rates given by Equations (5) and (6). In Markov Chain, the in-degree of a node is represented as a state of chain with the probability of going from one state to another being given by R_k and S_k; see Figure 1.

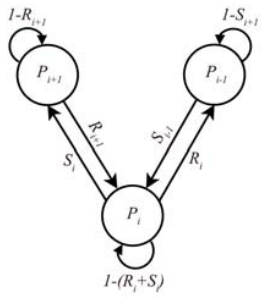

Fig. 1 Transition graph (states) for node's in-degree in the network.

For Markov chain, the stationary distribution for the expected number of jobs per node is defined as

$$V[T-I]=0 \tag{7}$$

where T is the transition matrix, I is the identity matrix, and V is the distribution vector (transition probability) and it is defined as

$$V=\begin{bmatrix} p_{k+1} & p_k & p_{k-1} \end{bmatrix} \tag{8}$$

From Figure 1, we can now obtain the transition matrix T;

$$T=\begin{bmatrix} 1-R_{k+1} & R_{k+1} & 0 \\ S_k & 1-(R_k+S_k) & S_k \\ 0 & S_{k-1} & 1-S_{k-1} \end{bmatrix} \tag{9}$$

By solving Equations (7), (8), and (9), the probability that a node has k edges, p_k, becomes

$$p_k=\frac{S_{k-1}}{R_k}p_{k-1}=\frac{S_{k-1}S_{k-2}\cdots S_0}{R_kR_{k-1}\cdots R_1}p_0 \tag{10}$$

From the above equation, we can see that in steady state, the rate at which node's in-degree increases will equal the rate at which node's in-degree decreases. Therefore, our network system has a fixed expected number of edges.

Now, since the total probability P_k is equal to one, and by inserting equations (5) and (6) into equation (10), and by using the Binomial Expansion Theorem [21] to simplify the equations, we will find that p_k is binomially distributed and it is given by

$$\begin{aligned} p_k &= \binom{M}{k}\left(\frac{NM-D}{D}\right)^k\left(\frac{D}{NM}\right)^M \\ &= \binom{M}{k}\left(1-\frac{D}{NM}\right)^k\left(\frac{D}{NM}\right)^{M-k} \end{aligned} \tag{14}$$

This degree distribution implies that the proposed network system is equivalent to the degree distribution of ER random network as illustrated in Equation (2) in Section II. These analytical results show that the stationary distribution is compatible with ER random networks. Thus, the proposed algorithm gives nearly optimal load distribution by creating almost regular network system where each node's in-degree refers to its free resources

V. Network Implementation

The proposed network system can be easily implemented in Grid networks. We can implement it on top of Grid network as a virtual network [22], or, we can integrate the proposed load-balancing scheme inside Grid network Middleware [23][24]. For example, this network system can be built directly on top of any of the physical transport layers and use the Grid Network as its underlying network. Thus, the network does not need to consist of physical links between nodes; the edges can be a routing table that gives the actual physical links or the possible routes between the nodes in the underlying physical layer. Furthermore, the network can be implemented by using small and fast transport protocols sockets that can be used to represent the edges of the network with minimum overhead. Each node will have local information about its status (i.e. its free resources available), which can be used for resource allocation and load distribution.

For network simulations, we will create a network system with N nodes, and the number of edges in each node will be proportional to its free resources. Node's edges will be added or removed to keep the in-degree of a node proportional to its free resources. Therefore, when a node initiates a new job, the in-degree of the node that will receive the new job will be decreased to show that its load increased and its free resources decreased. Hence, when a node initiates a new job, it randomly samples the network to assigns the new job to the node that has the highest in-degree. A new edge from the node that initiated the random sampling to the node that has the largest in-degree is created, and one of its edges will be randomly deleted.

Similarly, when a job is executed, the in-degree of the node that executed the job will be increased to show that its load decreased and its free resources increased. This done by randomly sampling the network, and then, a new edge will be created to connect it to the last node in the random sampling. A node can process a unit of job at each time step and the number of jobs that will be created or completed is a random variable with Poisson distribution. Simulation timing unit (iteration) is the time required to send a message or a data packet from one node to another node.

The node's edge insertion and deletion process described above will simulate the change in the workload of the network, and the amount of free resources available for the nodes will show the job distribution status of the network. Simulation results will be used to validate the reliability and scalability of the proposed load balancing mechanism.

VI. Simulation Results and Discussion

In order to verify that the proposed network system generates almost regular graphs and matches the analytical results, we made extensive simulations with various parameters and the results are reported in this section.

The steady state in-degree distribution and the in-degree standard deviation have been used to assess the load-balancing performance. It is known that regular graphs have zero standard deviation and zero variance since every node in the graph has the same in-degree. However, a zero standard deviation is only possible if the graph has an even number of nodes. Another balanced network is a network that half of its nodes have the expected in-degree $<k>$, and the other half have in-degree $<k+1>$ or $<k-1>$. In this case, the in-degree standard deviation is *+0.5* and *-0.5* respectively (i.e. the variance is equal to *0.25)*. Thus, the network is also considered a balanced network when its variance is close to *0.25*.

Under ideal conditions, simulation results confirm that the proposed network dynamic creates ER random networks. Figure 2 and Figure 3 show that the steady in-degree distributions are very close to the binomial distribution described in section IV. Here, the network have $N = 512$ nodes, with maximum in-degree $N-1$, and the average in-degrees $<k>$ are *32* and *72*. Thus, random sampling technique can be used to efficiently distribute the load between the nodes.

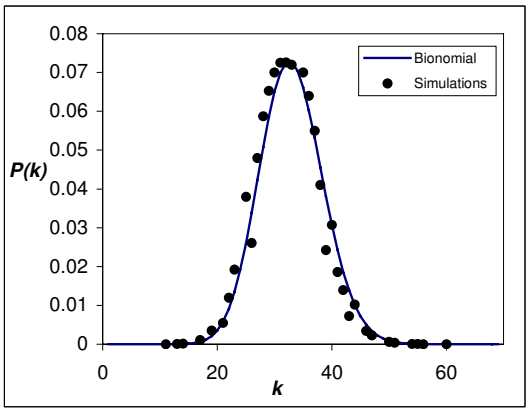

Fig. 2 The steady state in-degree distributions compared with the predicted binomial distribution for networks with $<k> = 32$.

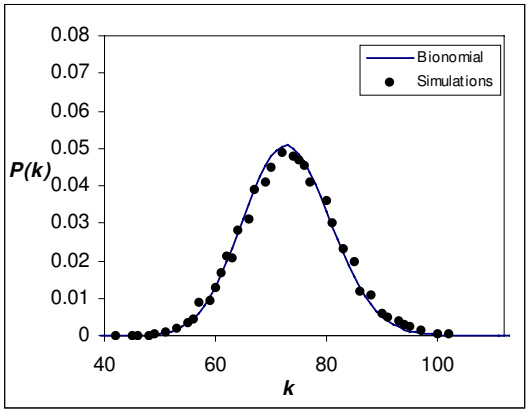

Fig. 3 The steady state in-degree distributions compared with the predicted binomial distribution for networks with $<k> = 72$.

We extended our simulations to analyse our load-balancing technique under several parameters and conditions. For example, to study the performance of the algorithm under different network loads, we examined our network under various load sizes. As we can see from figures 4, 5, and 6, whether the network is overloaded or nearly idle, the load balancing performance is almost identical. Thus, the algorithm is effective for networks with different network loads.

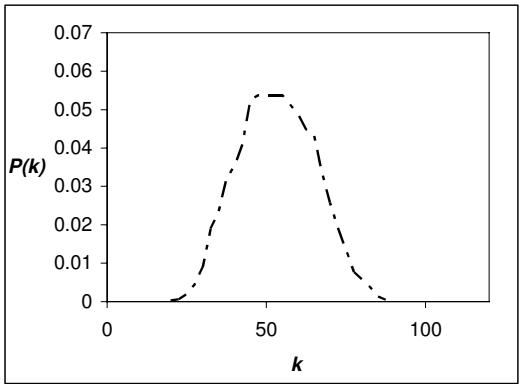

Fig. 4 The in-degree distributions when the network is loaded up to 25% of it capacity with *N=1024* and *M= 64*.

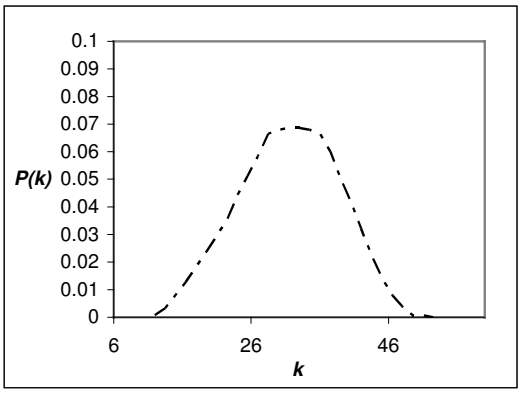

Fig. 5 The in-degree distributions when the network is loaded up to 50% of it capacity with *N=1024* and *M= 64*.

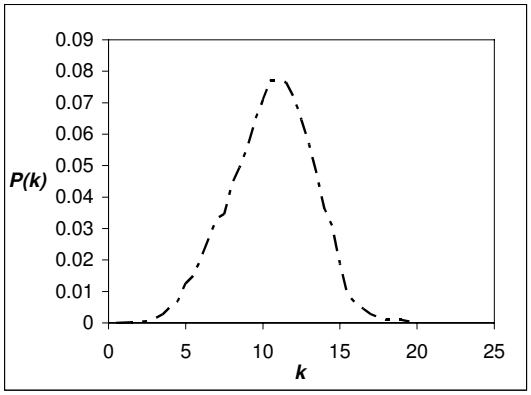

Fig. 6 The in-degree distributions for a network with loaded up to 85% of it capacity with *N=1024* and *M= 64*.

Figures 7 and 8 show that the proposed algorithm is scalable and that the generation of regular graphs using fitted random sampling is effective for various network sizes. As we can see from theses figures, the performance of the algorithm scales well for large network sizes. The figures show in-degree distributions are for graphs with *N=512* and *N=8192* respectively. Furthermore, by increasing the network size *N*, the in-degree distribution is closer to the binomial distribution of regular graphs, which indicate that this algorithm is designed for large-scale networks.

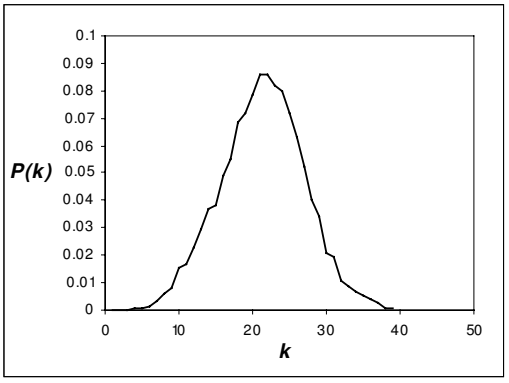

Fig. 7 The in-degree distributions for a network *N=512 & M= 48*.

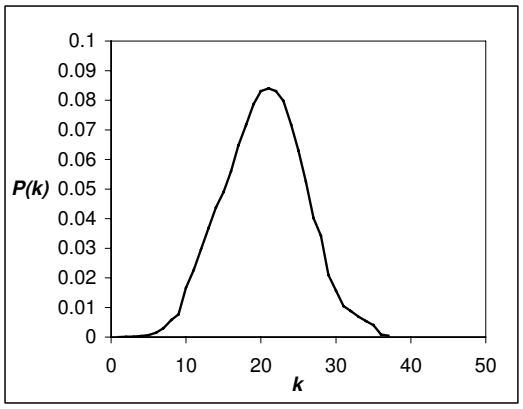

Fig. 8 The in-degree distributions for a network *N=8192 & M= 48*.

Figure 9 shows the state of the in-degree distributions of our network system with time as the network evolves. The network is initialized randomly. For example, it starts with in-degree variance approximately *51*. Then, the network starts reshaping with time by adding and deleting nodes' edges to reach an in-degree variance around *66*. Then, the network starts to settle down and the variance will rapidly decrease until the network become almost regular with in-degree variance close to *0.34*.

Intensive simulations have been done to observe the proper number of steps to efficiently sample the network to achieve the required load distribution, and how it will affect on the performance of load balancing algorithm. We found that the performance of the load-balancing algorithm increases as the sampling length increases.

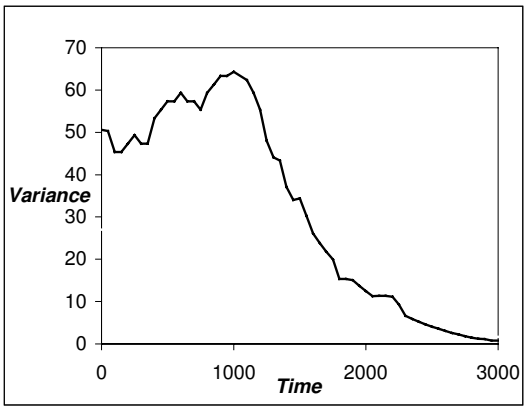

Fig. 9 The variance of the in-degree distribution vs. Time for a network with *N=2048* and *M=48*.

As we can see from Figure 10, increasing the sampling length will decrease the in-degree variance. Here we perform our simulations on a network of *2048* nodes with several sampling lengths. We found that if the random sampling is too short, the load distribution is not very efficient and the variance is very high. However, if the random sampling length is *16* or more, the in-degree variance is small and very close to 0.*25*, which is the variance for balanced networks.

Moreover, we observed that if the number of steps used to sample the network is very large, then the decrement in in-degree variance is very small. This observation is also noticed in larger network sizes, and the performance achieved by using very large sampling steps is very close to use random samples of length close to *log(N)*. Thus, using random samples with length around *log(N)* will be sufficient to reach an in-degree variance very close to the optimal variance, and this confirms that random sampling technique is very efficient in load-balancing.

To further measure the efficiency and robustness the proposed load balancing mechanism, we extended our simulations to investigate how nodes' in-degree and load distribution in the network will be affected by random errors at run time. To do this, we have introduced the possibility of node failure to the network after it has been settled down and distributed the load properly among the nodes. The number of nodes that will fail is random variable with Poisson distribution. Figure 11 shows the in-degree variance with time under this condition. As we can see from the figure, the variance is increased dramatically when nodes failed. However, we found the network starts to heal itself and dynamically reshape itself by re-distributing the load between the nodes. As a result, the in-degree variance will rapidly decrease and the network will become almost regular again. Thus, the proposed load-balancing scheme is reliable and it is robust to random errors.

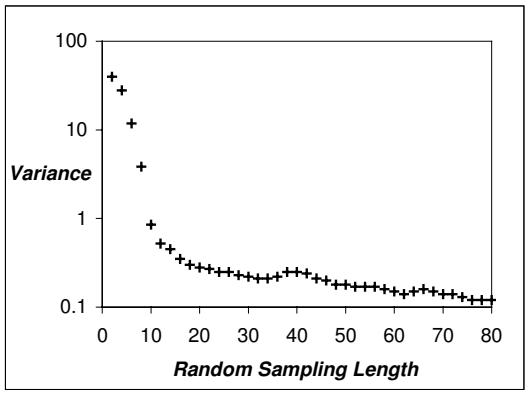

Fig. 10 The variance of the in-degree distribution vs. Random sampling length for a network with *N=2048* and *M=48*.

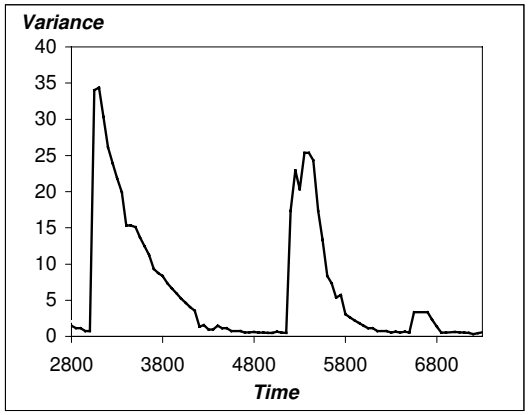

Fig. 11 The in-degree variance vs. Time for a network affected by random errors. *N=2048* and *M=48*.

VII. Conclusions

In this paper, we have presented a stochastic network system, which provides a distributed load-balancing scheme by generating almost regular networks. This network system is scalable, self-organized, robust, and depends only on local information for load distribution and resource discovery. The developed load-balancing scheme is based on fitted random sampling to assign the jobs and to update resource's availability. Therefore, load balancing is achieved without the need to monitor the nodes for their resources availability. Simulation results show that the generated network system provides an effective, scalable, and reliable load-balancing paradigm for the distributed resources accessible on large-scale network systems.

To further measure the efficiency of the proposed load balancing mechanism in various situations, we will extend our simulations to include heterogeneous nodes and cases where jobs may require certain QoS services; such as communications bounded, distance sensitive, and time bounded services. This will help us in understanding how these situations will affect on the nodes' in-degree and load distribution in the network. Examining how these considerations will affect on the efficiency of load balancing is a topic for future work.

References

[1] Foster, I. & Kesselman, K. (1999) *The Grid: Blueprint for A Future Computing Infrastructure*. Morgan Kaufmann.
[2] Lüling R., Monien B. & Ramme F. (1991) A Study of Dynamic Load Balancing Algorithms. *Proceedings of the Third IEEE SPDP*, 686-689.
[3] Peixoto, L. P. (1996) Load Distribution: A Survey. Technical Report. Dept. De inf, Escola De Engenharia, Universidade Do Minho.
[4] Murata, Y., *et al.* (2006) A distributed & cooperative load balancing mechanism for large-scale P2P systems. *SAINT-W*. USA.
[5] Mitzenmacher, M. (2001) The Power of Two Choices in Randomized Load Balancing. *IEEE Transactions on Parallel Distribution Systems*, 12(10).
[6] Drougas, Y., Repantis, T., and Kalogeraki, V. (2006) Load Balancing Techniques for Distributed Stream Processing Applications in Overlay Environments. *ISORC'06,* USA.
[7] Bustos, J., Denis Caromel, D., (2006) Load Balancing: Toward the Infinite Network, 12th Workshop on Job Scheduling Strategies for Parallel Processing, Saint-Malo, France.
[8] Theimer, M. M. & Lantz, K. A. (1989) Finding Idle Machines in A Workstation-Based Distributed System. *IEEE Transactions on Software Engineering*, 15(11).
[9] Oppenheimer, D., Albrecht, J., Patterson, D. & Vahdat, A. (2004) Scalable Wide-Area Resource Discovery. *Technical Report,* CA, USA.
[10] Subramanian, R. & Scherson, I. (1994) An Analysis of Diffusive Load Balancing. *Proc. of the sixth Annual ACM Symposium on Parallel Algorithms & Architectures*, ACM Press.
[11] Montresor, A., Meling, H. & Babaoglu, O. (2002) Messor: Load-Balancing Through a Swarm of Autonomous Agents. *First Intl. Workshop on Agents & P2P Computing*, Italy.
[12] Litzkow, M., Livny, M., & Mutka, M. (1988) Condor: A Hunter of Idle Workstations. *Proceedings of the Eighth International Conference of Distributed Computing Systems.*
[13] Yagoubi, B., and Slimani, Y. (2007) Task Load Balancing Strategy for Grid Computing**.** *Journal of Computer Science,* 3 (3): 186-194.
[14] Lüling, R. & Monien, B. (1993) A Dynamic Distributed Load Balancing Algorithm with Provable Good Performance. *SPAA '93,* ACM Press, New York, USA.
[15] Kremien, O. & Kramer, J. (1992) Methodical Analysis of Adaptive Load Sharing Algorithms. *IEEE Trans. On Parallel Distribution System*, 3(6).
[16] Erdös, P. & Rényi, A. (1959) *On Random Graphs*. Publicationes Mathematicae, (6).
[17] Bollobás, B. (1985) *Random Graphs*. Academic Press, London, England.
[18] Avin, C. & Brito, C. (2004) Efficient and Robust Query Processing in Dynamic Environments Using Random Walk Techniques, *Proc. of the third Intl. Symp on Info. Processing in Sensor Networks*. ACM Press.
[19] Lov'asz, L. & Winkler, P. (1995) Mixing of Random Walks and Other Diffusions on a Graph. *Surveys in Combinatorics*, London Mathematical Society Lecture Note Series.
[20] Kleinrock, L. (1975) *Queueing Systems. Volume I: Theory*. John Wiley & Sons, NY.
[21] Abramowitz, M. & Stegun, I. A. (1972) *Handbook of Mathematical Functions with formulas, Graphs, and Mathematical Tables*. Dover Publications, 9th Edition, New York.
[22] Adabala, S., Chadha, V., Chawla, P., Figueiredo, R., fortes, J., et al. (2005) From Virtualized Resources to Virtual Computing Grids: the In-Vigo System. *Future Generation Computer Systems*, 21(6).
[23] Blair, G.S., F. Costa, G. Coulson, H. Duran, et al. (1999) The Design of a Resource-Aware Reflective Middleware Architecture, *Proceedings of the 2nd international Conference on Meta-Level Architectures and Reflection*, St. Malo, France.
[24] Schantz, R. E. & Schmidt, D. C. (2001) Middleware for Distributed Systems: Evolving the Common Structure for Network-Centric Applications. *Encyclopaedia of Software Engineering*, Wiley & Sons, New York.