A distributed and scalable load-balancing scheme for Grid Networks

O. A. Rahmeh[1], P. Johnson[2]
[1]Faculty of Computing and IT
Sohar University, Sohar
Sultanate of Oman

[2]School of Engineering
Liverpool JMU, Liverpool
L33AF, UK
{o.rahmeh@soharuni.edu.om; p.johnson@ljmu.ac.uk}

ABSTRACT: *To provide viable solutions in computational problems, the Grid networks take advantage of the resources available by many computers connected within the networks. To achieve a scalable and reliable Grid network system, the workload needs to be efficiently distributed among the computing resources accessible on the network. Therefore, a distributed and scalable load-balancing scheme for Grid Networks is proposed in this paper. Besides we develop a latency reduction workload distribution protocol for Grid networks. In this work we document that introducing a latency reduction factor in the random sampling can reduce the effects of communication latency in the Grid network environment. Simulation results show that the resulted network system provides an effective, scalable, and reliable load-balancing scheme for the distributed resources available on Grid networks.*

Keywords: Grid networks, Load-balancing, Latency optimization

Received: 19 October 2010, Revised 17 November 2010, Accepted 29 November 2010

1. Introduction

For decades, numerous methods have been developed to maximize the use of networked computers for large-scale computing, and several protocols have been developed to efficiently utilise the resources within a distributed computing system. All these developments in technology have led to the possibility of using wide-area distributed computers for solving large-scale problems. Largescale computing networks can provide the ability to achieve higher throughput computing by taking advantage of many networked computers that simulates a virtual computer architecture environment where process execution is distributed among the computers in the network. An example of such network system is the Grid Network [1]. Grid networks use the resources of many computers connected within the network to solve large-scale computational problems. With Grid's huge number of distributed resources, an effective load-balancing paradigm to distribute the load among the available computers in the network can lead to improvement in the overall system performance. When one node is overwhelmed by work, it can make use of unused computing power in the network. Therefore, implementing and integrating an efficient load distribution and resource discovery protocol will have an essential role in the self-configuration and self-optimization characteristics of Grid networks.

One of the essential features of the Grid networks is that the resources accessible in the network are distributed geographically. However, one of the fundamental challenges to run Grid applications across geographically distributed computational resources is overcoming the effects of the latency between them. While high performance clusters and supercomputers can deliver data to applications with latencies of few microseconds, latency across the wide area networks is measured typically in milliseconds. Therefore, reducing the effects of communication latency is critical for achieving good performance with Grid applications that involve significant amounts of communication.

In this paper, an efficient biased random sampling (BRS) algorithm is shown to reduce communication latency in Grid networks and thus enabling the network to achieve load balancing which is scalable and reliable.

This paper is organised as follows. Section 2 reviews the related work on load balancing. Section 3 describes the proposed load balancing mechanism and the stochastic network system generated. Section 4 provides a description of network and simulation implementation. Finally, simulation results and conclusion have been discussed in section 5 and section 7 respectively.

2. Related work

Due to the critical role played by the need for load balancing in high-performance computing, there exists a large amount of research addressing various load balancing techniques, and numerous algorithms have been proposed to address this issue [2-6]. The uses of polling, agent-based methods, global random choice, randomised algorithms, and local diffusion methods have 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 smallscale networks or on particular properties of 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 field 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 the networks are analysed. Statistical models of large systems will let the systems detect or predict overall performance problems from the stream of data from individual devices.

Complex networks have been described using *Graph Theory* [16, 17]. Random graph theory was the simplest theory to describe complex network. Pál ErdQs and Alfréd Rényi (ER) were the first to study Random Graphs [16]. In ER model, the probability that an ER graph has more or less than the expected number of edges (*k*) decreases exponentially. This binomial distribution implies that each node will have a degree, which is close 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.

3. Proposed load balancing scheme

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

In order to design such dynamic system, we have to 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 the 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 indegree is performed via *Biased Random Sampling (BRS)*. Random sampling is the process whereby the nodes in the network are randomly picked up with equal probability. The sampling starts at some fixed node, and at each step, it moves to a neighbour of the current node, which is chosen randomly 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 an advantage over the previous methods in that the network structure is dynamically changed to efficiently distribute the load, and the loadbalancing 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, *biased* random sampling will be used where nodes' selection will depend on the free resources (indegree) available for each node.

In [19-21], we proposed a distributed load balancing framework for large scale networks. The addition and deletion of node's edges is performed via Fitted and Dynamic random sampling technique. In this paper, we improve our previous technique by introducing a latency reduction factor in the random sampling to reduce the effects of communication latency in Grid networks.

4. Network implementation and simulation methodology

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 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. Each node in the network is a computer with power equal to its maximum degree. One unit of power can process a unit of load in each unit of time. Two types of simulation experiments were carried out. The first experiment considered the CPU power alone as the key factor for load balancing. In the second experiment, the geographical distance (communication delay) is added as a second factor for load balancing.

Nodes' edges are added or removed to keep the in-degree of a node proportional to its free resources. Hence, when a node initiates a new job, it randomly samples the network to assign 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 to show that its load has increased and its free resources have decreased.

Similarly, when a job is executed, the in-degree of the node that executed the job is increased to show that its load decreased and its free resources increased. This is 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 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.

5. Evaluation and simulation results

We used extensive simulation results with various parameters to evaluate our load-balancing scheme, and to verify that the proposed network system generates almost regular graphs and matches the analytical results. The simulation results are discussed 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 where 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*.

In this section, simulation results have been used to analyse and discuss the performance of the load-balancing algorithm and to determine the length of random sampling required to achieve the required load balancing. Then, we evaluate the scalability and reliability of the algorithm under several conditions. We also study the effect of modifying the random sampling by including localization information on the average communication latency of the network.

5.1 The Load balancing performance

Here, we discuss the performance of proposed load balancing mechanism under ideal conditions, where all nodes have the same resources. The simulation results confirm that the proposed network dynamic creates ER random networks. Figure 1 shows the simulation results for the indegree distribution of the network plotted as the network evolved through different time slots (*T*), which shows the process of reaching the load balancing. Here the time dynamics of the in-degree distributions of the network can be clearly seen. In Figure 1.a, the network is initialised in a completely random state with variance approx. *46.3*. Then, the network starts reshaping itself by balancing the load distribution among the nodes, and in-degree variance decreases to approx. *11.4* at *T=2500*; see Figure 1.b. Over time, the network settles down to a nearly regular graph with variance approx. *0.32* as seen in Figure 1.c. Thus, when all the nodes have the same capabilities, the network will be almost a regular graph.

Figure 2 shows the state of the in-degree variance of our network system with time as the network evolves. The network is initialised randomly; for example, it starts with an in-degree variance of approximately *42.6*. Then, the network starts reshaping with time by adding and deleting nodes' edges to reach an in-degree variance value of around *63.3*. Then, the network starts to settle down and the variance rapidly decreases until the network becomes almost regular with in-degree variance close to *0.38*.

5.2 Random sampling length

Intensive simulations have been carried out to observe the number of steps needed to efficiently sample the network to achieve the required load distribution, and evaluate its effect on the performance of load balancing algorithm. We found that the performance of the load-balancing algorithm improves as the sampling length increases.

Moreover, we observed that if the number of steps used to sample the network is very large, then the decrement in the in-degree variance is very small. This is also observed in larger network sizes, and the performance achieved by using very large sampling steps is very close to that when random samples of length close to *log(N)* are used. 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.

As we can see from Figure 3, increasing the sampling length will decrease the in-degree variance. Here we performed our simulations on a network size of *2048* nodes with several values for sampling lengths.

As expected, 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, then the in-degree variance is small and very close to 0.*25*, which is the variance for balanced networks.

To further evaluate the performance of the proposed biased random sampling algorithm, we examined the bandwidth required by the BRS load distribution mechanism and we compared it with the performance of the centralized mechanism.

It has been observed that the central server algorithm requires less total bandwidth than the BRS algorithm. In the centralised scheme, the central node has to know the load status in each of the nodes that in the network. Therefore, the central node needs to periodically check the status of every node in the network, and the nodes have to inform the central node if they finished executing the jobs so that the central node can update network load status. As a result, the total bandwidth consumed by the network is in order of O(*N*).

For the BRS algorithm, each node that initiates a new job must initiate a random sampling to search for a node to give it the job. And since the random walk will be O(*logN*) length, the total bandwidth of the walk will be in order of O(*logN*). Therefore,

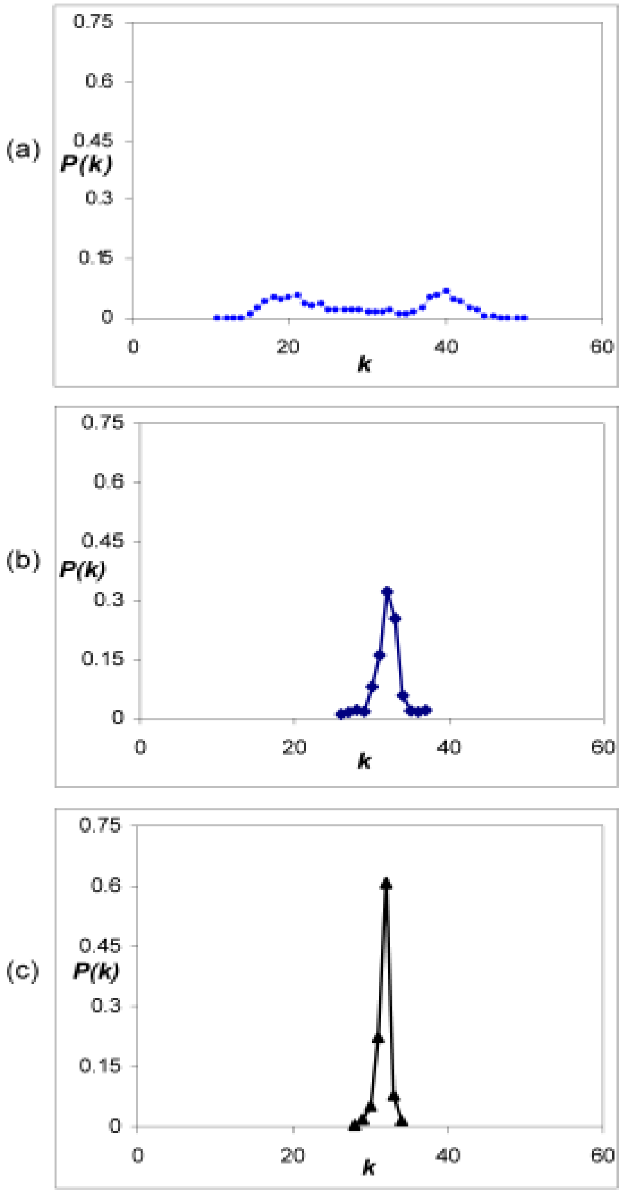

Figure 1. The in-degree distribution plotted as the network evolves over different time slots (*T*)

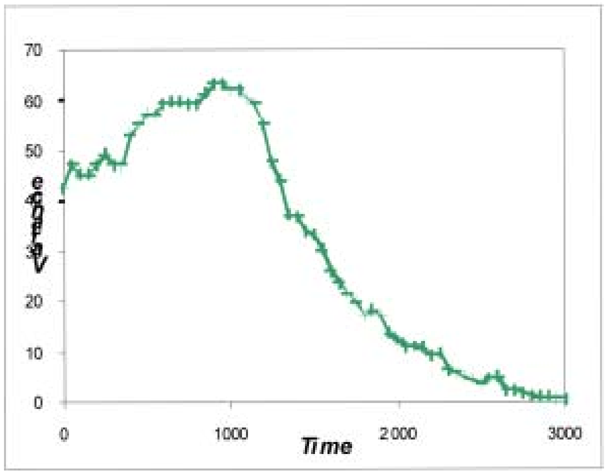

Figure 2. The variance of the in-degree distribution vs. Time for a network with N=2048 and M=48

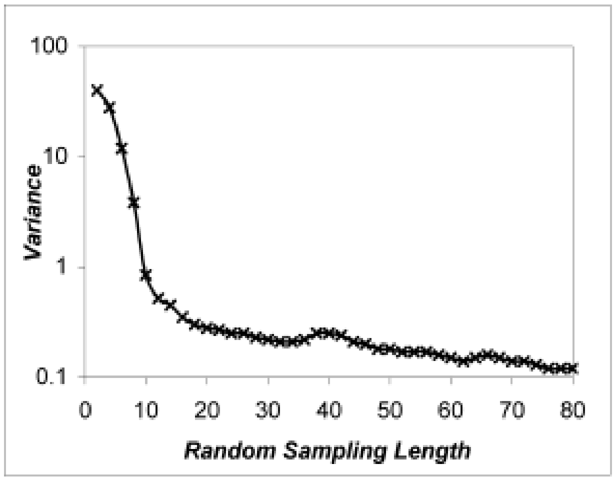

Figure 3. The effect of random sampling length on the in-degree variance. *N=2048* and *M=48*

for *N* nodes network, the total bandwidth consumed by the biased random sampling algorithm will be in order of O(*NlogN*), which is greater than the total bandwidth consumed in the centralised scheme.

However, the biased random sampling scheme decreases the bandwidth consumed by any individual node in the network.

The central node in the centralised scheme is engaged in all jobs and handshaking transfers. Therefore, the central node consumes a O(*N*) bandwidth. For the BRS algorithm, the bandwidth required for each node depends on the node in-degree and on the number of jobs it initiates. Thus, each node in the network will consume a bandwidth in order of O(*logN*).

Although the total bandwidth consumed in the network is a significant performance measurement, the bandwidth consumed by any single node in the network can be a major bottleneck for large-scale networks.

6. Latency reduction load distribution

In reality, load balancing is not restricted to the use of resources or computing power, but also is influenced by the geographical distance between the nodes. Therefore, we included locality information into the random sampling scheme. Thus, the random walk may prefer a geographically closer node even if it is not the highest degree on the walk. We implemented this by adding the geographical distance and communication delay factors in sampling the nodes to distribute the load balancing.

Simulation results show that adding the locality factor reduced the overall network latency. We performed two experiments and recorded the average round trip latencies for each executed job; see Figure 4.

As can be seen from Figure 4, the latencies observed in the offered load using geographicaware scheme are reduced by at least *22%* on average from that observed for the non-geographic aware scheme. For example, for networks with *512* nodes distributed with a radius of *1000km*, the overall average latency decreased from *92.58ms* to *70.17ms*. Moreover, we observed that latencies for individual loads by using this algorithm will always remain close to the average latency with no big overshoots (fluctuations), which make the network stable and reliable and a suitable environment for applications that require specific quality of service.

Furthermore, to examine the efficiency of adding locality factor on load balancing, simulation results were analysed for the network under consideration over several sampling lengths, and compared with the original scheme. As we can see from Figure 5, the Geographic-aware load balancing requires few additional sampling steps to achieve the required variance for balanced network. For example, for a network with *2048* nodes, a random sampling length of *16* was sufficient for the original scheme, while the Geographic-aware scheme required a random sampling length of *20* to balance the load distribution, which still in order of *log(N)*. However, this increase in number of steps is negligible compared to the size of the network.

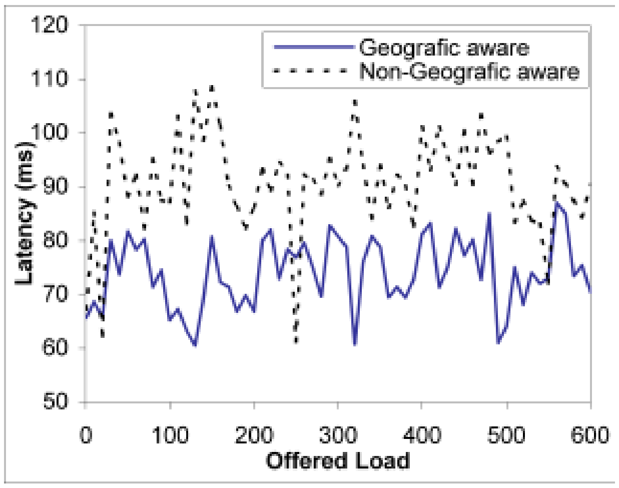

Figure 4. The average round trip latencies observed for finished jobs in a network with *N=512*

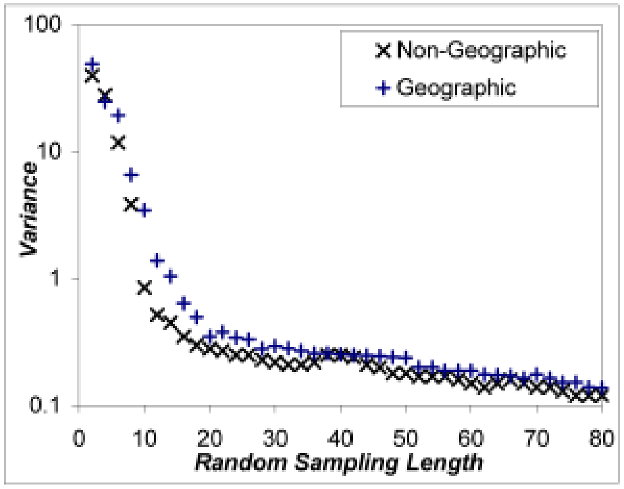

Figure 5. Comparison of the variance vs. random sampling length for a network with *N=2048*

7. Conclusions

In this paper, we proposed an effective, scalable, and reliable mechanism for distributing and balancing the workload between the distributed resources available on Grid networks. The proposed mechanism is scalable, self-organised, robust, and depends only on local information for load distribution and resource discovery. It is based on biased random sampling (BRS) 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. Then, we demonstrated that introducing a latency reduction factor in the random sampling can reduce the effects of communication latency in Grid networks environments. Therefore, the proposed scheme can be used to reduce the effects of communication latency which is essential for achieving good performance with Grid applications that involve significant amounts of communication.

8. References

[1] Foster, I., Kesselman, K. (1999). The Grid: Blueprint for A Future Computing Infrastructure, Morgan Kaufmann.

[2] Lüling, R.B., Monien 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 Randomised Load Balancing, *IEEE Transactions on Parallel Distribution Systems*, 12(10).

[6] Drougas, Y., Repantis, T., Kalogeraki, V. (2006). Load Balancing Techniques for Distributed Stream Proceszing Applications in Overlay Environments, *ISORC'06*, USA.

[7] Bustos, J., Caromel, D. (2006). Load Balancing: Toward the Infinite Network, *In*: 12th Workshop on Job Scheduling Strategies for Parallel Proceszing, 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., 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, 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 Proceszing in Dynamic Environments Using Random Walk Techniques", *Proc. of the third Intl. Symp on Info. Proceszing in Sensor Networks*. ACM Press.

[19] Rahmeh, O. A., Johnson, P., Lehmann, S. (2007). A Fitted Random Sampling Scheme for Load Distribution in Grid Networks, *the International Journal of Information Technology*, V 24.

[20] Rahmeh, O. A., Johnson, P. (2008). Towards scalable and reliable Grid Networks, *IEEE/ACS International Conference on Computer Systems and Applications*, 4 April 2008, 253–259, Doha, Qatar.

[21] Rahmeh, O. A., Johnson, P., Taleb-Bendiab, A., (2008). A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks, *In*: T*he INFOCOMP Journal of Computer Science*, Vol. 8, No. 1,.

[22] Adabala, S. Chadha, V. Chawla, P. et al. (2005). From Virtualised Resources to Virtual Computing Grids: the In-Vigo System, *Future Generation Computer Systems*, 21 (6).

[23] Blair, G. S., Costa, F., Coulson, G., Duran, Het al. (1999). The Design of a Resource-Aware Reflective Middleware Architecture, *In:* 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, N.Y.