A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks

O. A. RAHMEH[1]
P. JOHNSON[2]
A. TALEB-BENDIAB[3]

[(1,2)]School of Engineering, Liverpool John Moores University, L3 3AF, Liverpool, UK
[3]School of Computing and Mathematical Sciences, Liverpool John Moores University, L3 3AF, Liverpool, UK
[(1,2,3)] (enroabur,p.johnson,a.talebbendiab)@livjm.ac.uk

Abstract. The growth in computer and networking technologies over the past decades produced new type of collaborative computing environment called Grid Network. Grid is a parallel and distributed computing network system that possesses the ability to achieve a higher throughput computing by taking advantage of many computing resources available in the network. Therefore, to achieve a scalable and reliable Grid network system, the load needs to be efficiently distributed among the resources accessible on the network. In this paper, we present a distributed and scalable load balancing framework for Grid networks using biased random sampling. The generated network system is self-organized and depends only on local information for load distribution and resource discovery. We demonstrate that introducing a geographic awareness factor in the random walk sampling can reduce the effects of communication latency in the Grid network environment. Simulation results show that the generated network system provides an effective, scalable, and reliable load balancing scheme for the distributed resources available on Grid networks.

Keywords: Load balance, Grid, Modeling, Simulations, Complex Networks, Random algorithms.

(Received March 17, 2008 / Accepted August 25, 2008)

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 utilize 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. Large-scale 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 [7]. 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 be-

tween 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 organized 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 presents the mathematical analysis of the network system and provides a stationary distribution solution. Section 5 provides a description of network and simulation implementation. Finally, simulation results and conclusion have been discussed in section 6 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 [5][12][14][16]. The uses of polling, agent-based methods, global random choice, randomized algorithms, and local diffusion methods have produced great advances in the field of load balancing [8][15][17][19][20]. However, most of these methods depend on central server techniques, which can be efficient in small-scale 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 [10][11]. 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 analyzed. 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* [6][4]. Random graph theory was the simplest theory to describe complex network. Pául Erdős and Alfréd Rényi were the first to study Random Graphs [6]. 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

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

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

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

where C_{N-1}^{k} is the probability space in which k edges are chosen from the total $N-1$ number of edges.

Thus, 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 analyze 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 in-degree 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 neighbor 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 [15][3]. 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 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, *biased* random sampling will be used where nodes' selection will depend on the free resources (in-degree) available for each node.

Lovász and Winkler [13] 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 also be applied to our directed graphs since the underlying network has fixed average number of edges. Therefore, biased random sampling technique will be used in our network to provide dynamic load balancing, and the insertion and deletion strategy of edges assures that the load will be distributed equally across all the nodes in the network.

4 Load distribution analysis

The proposed load distribution mechanism is difficult to analyze mathematically due to the dynamic nature of the network and the way the biased random sampling works. Therefore, the analytical analysis was simplified by restricting the load distribution mechanism to use a simple random sampling scheme that selects the last node in the walk, rather than using the *biased* random sampling [18]. A network system with N nodes is considered for this work, and it is assumed 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 total numbers of edges inserted and deleted in the network 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*. It should be noted that this assumption is not a limitation of the mechanism, but 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, 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. Hence, the rate at which the in-degree of a specific node will decrease is given by

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

Similarly, 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 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 [9] 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 where the probabilities of going from one state to another are given by R_k and S_k.

By mathematically analyzing the above insertion and deletion rates, the probability p_k that a node has k edges is given by

$$p_k = \frac{S_{k-1}}{R_k} p_{k-1} = \frac{S_{k-1} S_{k-2} \cdots S_0}{R_k R_{k-1} \cdots R_1} p_0 \tag{7}$$

From the above equation, we can see that in steady state, the rate at which the 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.

Since the total probability P_k is equal to one, and by inserting equations (5) and (6) into equation (7), and by using the Binomial Expansion Theorem [1] to simplify the equations, we will find that p_k is binomially distributed and it is given by

$$\begin{aligned} p_k &= \begin{pmatrix} M \\ k \end{pmatrix} \left(\frac{NM-D}{D}\right)^k \left(\frac{D}{NM}\right)^M \\ &= \begin{pmatrix} M \\ k \end{pmatrix} \left(1-\frac{D}{NM}\right)^k \left(\frac{D}{NM}\right)^{M-k} \end{aligned} \qquad (8)$$

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 2. 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.

The performance of load balancing technique has been improved by assigning the new job to the least loaded (highest in-degree) node in the walk, instead of the last node in the walk. When the node with the most free resources on a walk is preferred to receive the new job, its resources must be greater than or equal to the resource of the last node on the random walk. Therefore, we can expect that it will have the same scalability as the standard random walk, and the balancing performance is much improved as shown in the following simulation results.

5 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 [2], or, we can integrate the proposed load balancing scheme inside Grid middleware [7]. 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.

6 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 analyze 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.

6.1 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. As can be seen in Figure 1, the simulation show that the steady in-degree distributions are very close to the binomial distribution described in section 4. The network under consideration for this simulation has *N=512* nodes, with maximum in-degree *N-1*. Thus, the random sampling technique can be used to efficiently distribute the load between the nodes.

Figure 2 shows the simulation results for the in-degree distribution of the network plotted as the network evolved

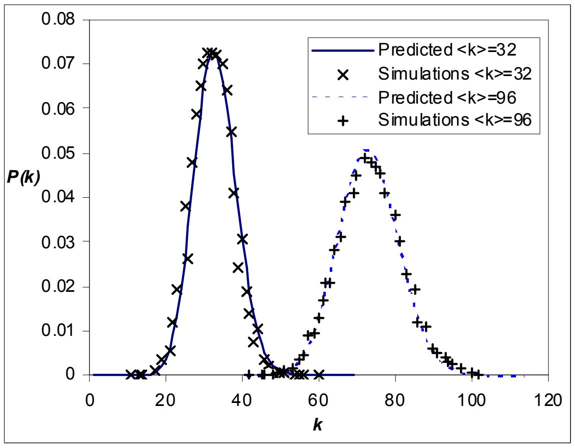

Figure 1: Simulation results of the steady state in-degree distributions compared with the predicted binomial distribution for networks with *N =512*.

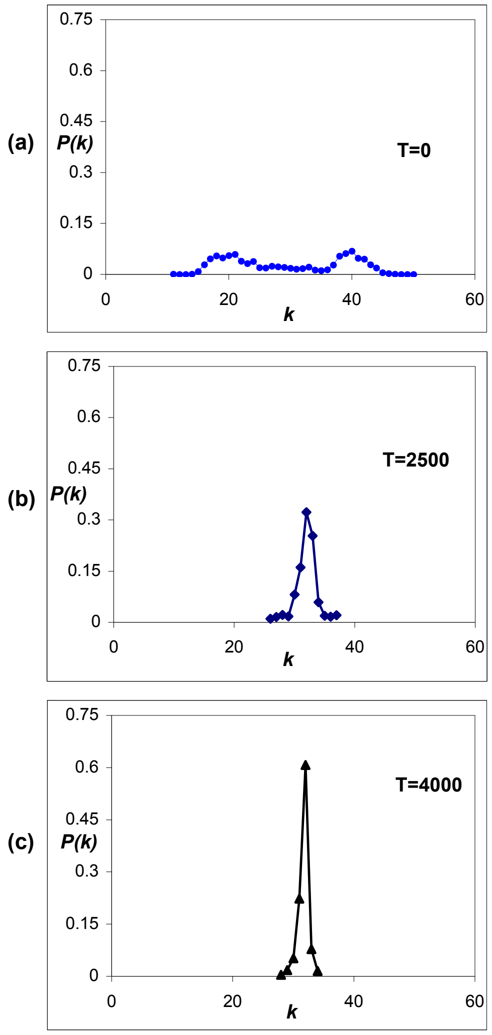

Figure 2: The in-degree distribution plotted as the network evolves over different time slots (*T*).

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 2.a, the network is initialized 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 2.b. Over time, the network settles down to a nearly regular graph with variance approx. 0.32 as seen in Figure 2.c. Thus, when all the nodes have the same capabilities, the network will be almost a regular graph.

We extended our simulations to analyze 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 can be seen from Figure 3, whether the network is overloaded or is nearly idle, the load balancing performance is almost identical. Thus, the algorithm is effective for networks with different network loads.

Figure 4 shows the state of the in-degree variance of our network system with time as the network evolves. The network is initialized 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.

6.2 Scalability

The simulation results as depicted in Figures 5 and 6 show that the proposed algorithm is scalable and that the generation of regular graphs using biased random sampling is effective for various network sizes. Simulations have been carried out for growing values of network size and results presented here to demonstrate the true scalability of the algorithm.

As we can see from theses graphs, the performance of the algorithm scales well specially for large network sizes. The figures show in-degree distributions are for graphs with $N = 512$ and $N = 8192$ respectively. In addition, 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. Simulations have been carried out not only at discrete values of different network sizes, but also to capture the dynamics of network growth. So, this algorithm is found suitable for growing networks.

6.3 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.

As we can see from Figure 7, increasing the sampling length will decrease the in-degree variance. Here we performed our simulations on a network size of 2048

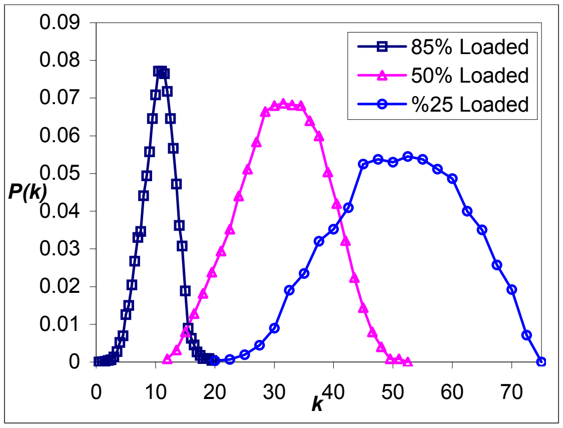

Figure 3: The in-degree distributions under different network loads and *N=1024 and M= 64.*

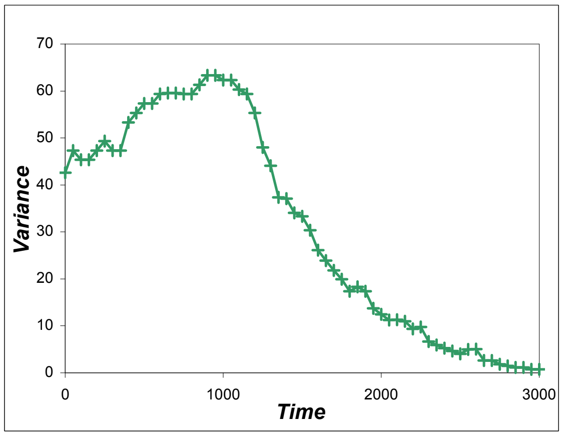

Figure 4: The variance of the in-degree distribution vs. Time for a network with *N=2048 and M=48.*

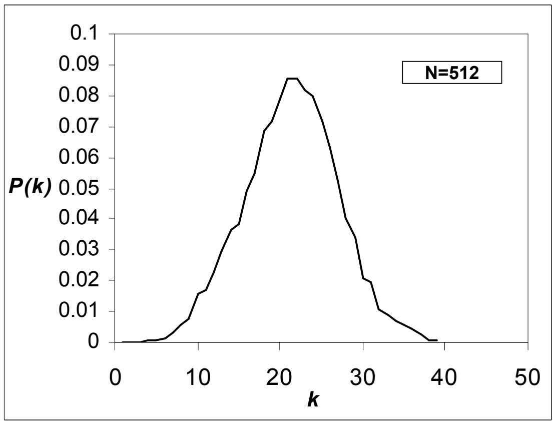

Figure 5: The in-degree distributions for a network with *N=512* and *M= 48.*

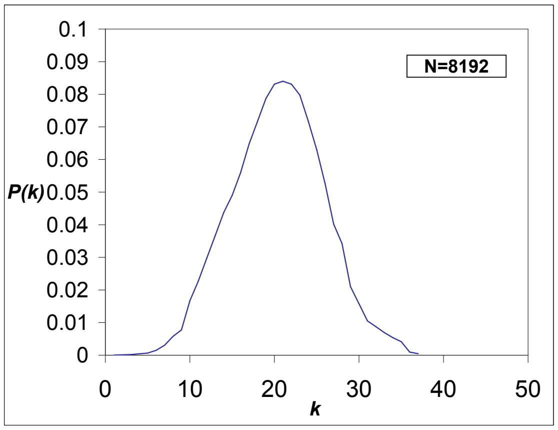

Figure 6: The in-degree distributions for a network with *N=8192* and *M= 48.*

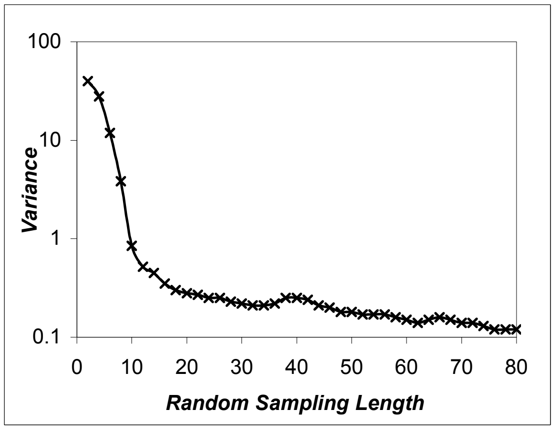

Figure 7: The effect of random sampling length on the in-degree variance. *N=2048 and M=48.*

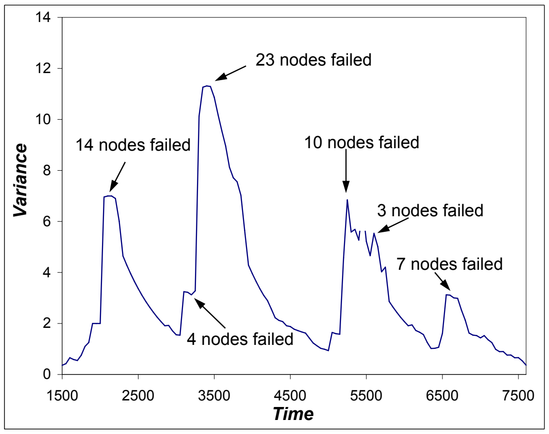

Figure 8: The in-degree variance for a network affected by random attack. *N=2048 and M=48.*

nodes with several values for 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, then 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 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.

6.4 Reliability

To further measure the efficiency and robustness the proposed load balancing mechanism, we extended our simulations to investigate how the 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 settled down and distributed the load properly among the nodes. Node failure can happen due to node errors or attack. The number of nodes that will fail is a random variable with Poisson distribution.

Figure 8 shows the in-degree variance with time under this condition. As we can see from the figure, the variance has increased dramatically when nodes failed. However, the network starts to heal itself and dynamically reshapes 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 robust to random errors, and it will enable the network to dynamically and efficiently reorganize and heal itself against node failure or attack

6.5 Geographic aware 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.

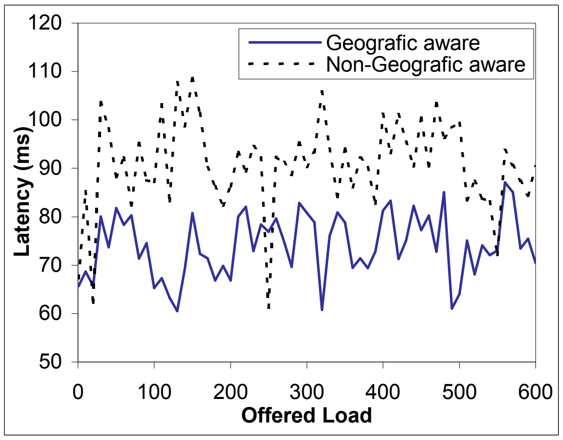

Figure 9: The average round trip latencies observed for finished jobs in a network with *N=512*.

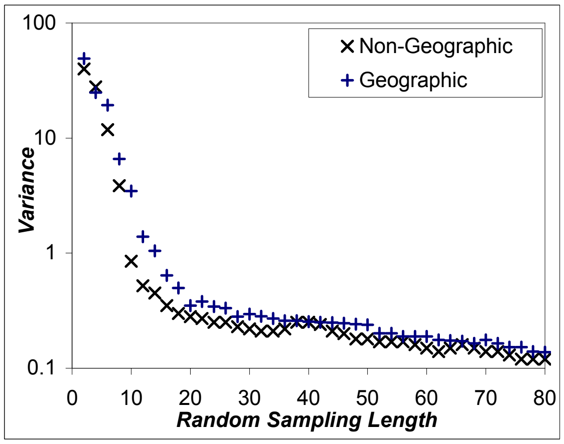

Figure 10: Comparison of the variance vs. random sampling length for a network with *N=2048*.

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 9.

As we can see from Figure 9, the latencies observed in the offered load using geographic-aware 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 analyses for the network under consideration over several sampling lengths, and compared with the original scheme. As we can see from Figure 10, 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.

6.6 Performance comparison

To evaluate the performance of the proposed biased random sampling algorithm, we examined two important performance measurements in distributed systems: the total job throughput achieved and the bandwidth required by the load distribution mechanism. Then we compared the performance the biased random sampling scheme with the performance of the centralized mechanism.

Here, we analyses the bandwidth consumed by the biased random sampling (BRS) algorithm and we compared it with the centralized algorithm. As we can see from Figure 11, the central server algorithm requires less total bandwidth than the biased random sampling algorithm. In the centralized 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 biased random sampling 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, for N modes 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 centralized scheme.

However, the biased random sampling scheme decreases the bandwidth consumed by any node in the network, as shown in Figure 12. The central node in the centralized scheme is engaged in all jobs and handshaking transfers. Therefore, the central node consumes a O(N) bandwidth. For the biased random sampling algorithm, the bandwidth required for each node depends on the node in-degree and on the number of jobs it initiates. Thus, if the N nodes in the network use the total network bandwidth uniformly, then 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.

Another important performance criteria in distributed networks is the system throughput. Throughput is the number of jobs completed during a specified period of time. The objective here is to have the maximum amount of completed jobs (large amount of throughput). Therefore, we analyses the total throughput achieved by the biased random sampling algorithm and compared it with the central system.

Figure 13 show simulation results for the throughput achieved by both the central load balancing scheme and the biased random sampling scheme. For the simulations, the nodes in a network have equal capabilities, and the job sizes and arrival rates are Poisson distributed. The unit of time (iteration) in this figure is the time required to send a message or a data packet from one node to another node. Moreover, we considered the effect of communication delay on the total throughput performance by distributing the nodes in a network of 1000 miles radius area with *10Mbps* communication link speed. As we can see from the figure, the throughput achieved by the biased random sampling scheme is very close to the throughput achieved by the optimal centralized scheme. The total throughput for biased random sampling algorithm is only around *3%* worse than the total throughput of the central algorithm, with the advantage of being a distributed load balancing scheme.

To further measure the efficiency of the proposed biased random sampling mechanism for load balancing in various situations, the simulations could be extended to include heterogeneous nodes and cases where jobs may require certain quality of service (QoS); such as communications bounded, distance sensitive, and time bounded services. Examining how these considerations will affect the efficiency of load balancing is a topic for future work.

7 Conclusions

In this paper, we proposed an effective, scalable, and reliable biased random sampling scheme for balancing the load between the distributed resources available on Grid networks. We presented a stochastic network system, which provides a distributed load balancing scheme by generating almost regular networks. This network

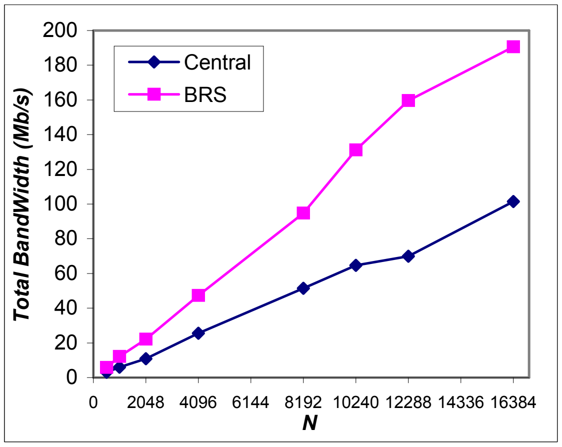

Figure 11: Simulation results for the total bandwidth consumed in different network sizes.

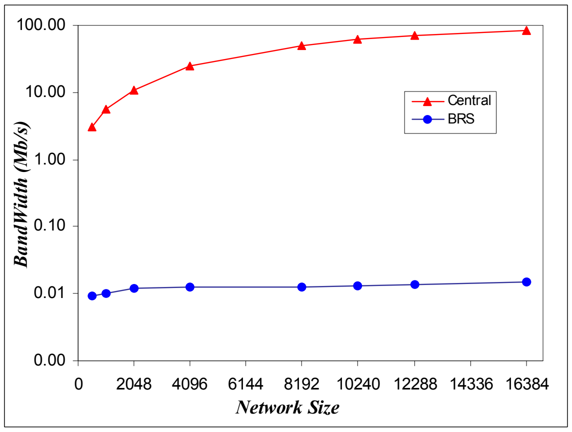

Figure 12: Simulation results for the average bandwidth consumed by single nodes for different network sizes.

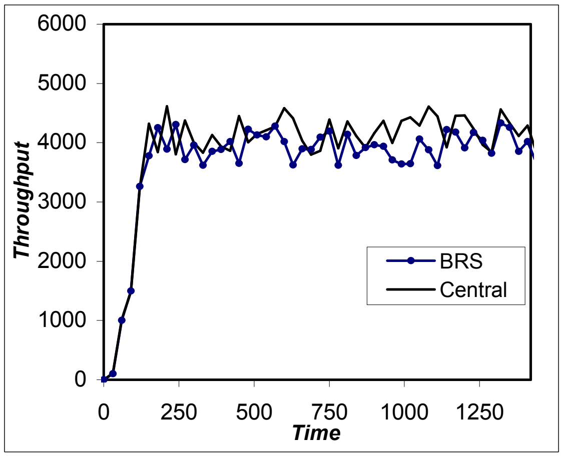

Figure 13: Figure 13. Simulation results for the average bandwidth consumed in single nodes for different network sizes.

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 biased 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 Grid networks resources. In addition, we demonstrated that introducing geographic awareness factor in the random walk sampling could reduce the effects of communication latency in Grid network environments.

A number of potential improvements to our load balancing technique and generalizations of our model deserve further study. We plan to extend this work 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.

References

[1] Abramowitz, M. and Stegun, I. A. *Handbook of Mathematical Functions with formulas Graphs and Mathematical Tables*. Dover Publications, NY, 1972.

[2] Adabala, S., Chadha, V., Chawla, P., and Figueiredo, R. From virtualized resources to virtual computing grids: the in-vigo system. *Future Generation Computer Systems*, 6(21), 2005.

[3] Avin, C. and Brito, C. Efficient and robust query processing in dynamic environments using random walk techniques. *Proc. of the 3rd Intl. Symp on Info. Processing in Sensor Networks*, 2004.

[4] Bollobás, B. *Random Graphs*. Academic Press, London, England, 1985.

[5] Drougas, Y., Repantis, T., and Kalogeraki, V. Load balancing techniques for distributed stream processing applications in overlay environments. *ISORC'06*, 2006.

[6] Erdös, P. and Rényi, A. On random graphs. *Publicationes Mathematicae*, 6, 1959.

[7] Foster, I. and Kesselman, K. *The Grid: Blueprint for A Future Computing Infrastructure*. Morgan Kaufmann, 1999.

[8] J. Bustos, D. C. Load balancing: Toward the infinite network. *12th Workshop on Job Scheduling Strategies for Parallel Processing*, 2006.

[9] Kleinrock, L. *Queueing Systems- Volume I: Theory*. John Wiley and Sons, NY, 1975.

[10] Kremien, O. and Kramer, J. Methodical analysis of adaptive load sharing algorithms. *IEEE Trans. On Parallel Distribution System*, 6(3), 1992.

[11] Lüling, R. and Monien, B. A dynamic distributed load balancing algorithm with provable good performance. *SPAA 93*, 1993.

[12] Lüling, R., Monien, B., and Ramme, F. A study of dynamic load balancing algorithms. *Proceedings of the Third IEEE SPDP*, pages 686–689, 1991.

[13] Lov'asz, L. and Winkler, P. Mixing of random walks and other diffusions on a graph. surveys in combinatorics. *London Mathematical Society Lecture Note Series*, 1995.

[14] Mitzenmacher, M. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel Distribution Systems*, 10(12), 2001.

[15] Montresor, A., Meling, H., and Babaoglu, O. Messor: Load-balancing through a swarm of autonomous agents. *First Intl. Workshop on Agents and P2P Computing*, 2002.

[16] Murata, Y., Inaba, T., Takizawa, H., and Kobayashi, H. A distributed and cooperative load balancing mechanism for large-scale p2p systems. *SAINT-W*, 2006.

[17] Oppenheimer, D., Albrecht, J., Patterson, D., and Vahdat, A. Scalable wide-area resource discovery. *Technical Report*, 2004.

[18] Rahmeh, O. A. and Johnson, P. A distributed load balancing mechanism for large-scale networks. *Intl. Conference on Communications and Computer Applications and Technologies*, 2007.

[19] Subramanian, R. and Scherson, I. An analysis of diffusive load balancing. *Proc. of the sixth Annual ACM Symposium on Parallel Algorithms and Architectures*, 1994.

[20] Yagoubi, B. and Slimani, Y. Task load balancing strategy for grid computing. *Journal of Computer Science*, 3(3):186–194, 2007.