A Periodic Symmetrically-Initiated Load Balancing Algorithm for Distributed Systems

K. Benmohammed-Mahieddine[1], P. M. Dew, and M. Kara

School of Computer Studies, University of Leeds, LS2 9JT (UK)
Fax: +44-532-335468

Abstract

This paper presents a new load balancing algorithm called Periodic Symmetrically-Initiated (PSI). *It is symmetrically initiated and uses periodic polling of a single random node to acquire information about the system state. Its performance on a distributed system is compared to the following algorithms: Random, Sender, Receiver, and Symmetric. These algorithms have been evaluated in [15] where the terms Sender and Receiver are referred to as Forward and Reverse respectively. The algorithms performance under different variants to the system model is also studied to assess the sensitivity of the PSI algorithm to changes in the system attributes. This includes the communication bandwidth, the system size, the file system structure, and the workload model. The results of the study show that the* PSI *algorithm outperforms the existing algorithms and is robust over a range of system attributes.*

Key words: Distributed Systems, Load Balancing Algorithms, Performance Evaluation, System Attributes.

1. Introduction

One important problem that arises in distributed systems, is to improve the performance of a set of computing nodes through a dynamic workload redistribution. This is to avoid the situation where some nodes are idle while others have multiple jobs queued up [18, 15].

A load balancing algorithm can be classified by its information, transfer, and negotiation policies [4]. The *information policy* indicates the amount of information about the system and the information gathering rule used in making the load redistribution decisions. The *transfer policy* determines when to attempt a job transfer and which job to transfer. The *negotiation policy* selects the nodes to or from which jobs will be transferred. Due to the large number of options for each policy the studied load balancing algorithms are based on similar information and negotiation policies but use different transfer policies. One aspect of the transfer policy is the load condition for the node that initiates the load balancing process. In sender-initiated algorithms the load balancing process is initiated by the overloaded nodes, whereas for receiver-initiated algorithms it is initiated by the underloaded nodes. In a load balancing algorithm based on a symmetrically initiated transfer policy [13, 16], the load balancing process can be either initiated by the overloaded node if the load index exceeds a pre-determined threshold, or by the underloaded node if the load index drops below a given threshold.

Among the many algorithms reported in the literature, a set of algorithms called Random, Sender, Receiver, and Symmetric covering the above polices is evaluated. These algorithms have been compared in [16] where the terms Sender and Receiver are referred to as Forward and Reverse respectively. It is concluded that Symmetric algorithm produces the lowest job response time. In this paper a symmetrically-initiated algorithm with periodic probing called *PSI* is proposed. It involves a smaller number of load balancing messages than the Symmetric algorithm and a spreading of probing messages over time.

Three methods can be used to study the performance of load balancing algorithms namely analytical modelling based on queuing theory [15, 20, 10], simulation modelling [19, 22], and prototyping [21, 8]. The first approach is often based on simplified model assumptions (e.g. instantaneous job transfers and at no cost), leading to results which are useful only to set performance bounds [12]. Also as shown in a survey by Wang *et al.* [20] even simple load balancing schemes can lead to unsolved problems in queuing theory. Simulation has been chosen because it allows the building of a model with realistic assumptions and makes it possible to have a complete control over all parameters and events of the system under study and experimentation in virtual time [11]. Once the simulation study has been completed it is then sensible to prototype a real system.

The purpose of this research is to evaluate the *PSI* algorithm on a simulated model of a distributed system. A baseline system which comprises a fixed set of system attributes and workload parameters, is used for most of the experiments. Following the ranking of the algorithms on this system, the performance of the algorithms is investigated on variants to the system model. These variants include the study of the effect of:

- ratio R= system compute rate /communicate rate
- workload model
- system size
- diskless and disk-based system

The main results of this study are:

- the PSI algorithm has superior performance characteristics compared with the other algorithms studied over a range of systems;
- the file system structure, communication bandwidth, workload model, and system size have no significant effect on the ordering of the algorithms but lead to different levels of performance; and
- for the case of adaptive load balancing schemes based on dynamic parametric tuning the essential parameters to monitor are the communication delay, the system load, and the system size, whereas the adjustable parameters are the *threshold* and the *timer period.*

[1] Now at Computer Science Dept, INELEC, Boumerdes, 35000 (Algeria), Fax: +213-281-1400

In the next section, the system architecture and the set of selected load balancing algorithms are described. Thereafter the periodic symmetrically-initiated load balancing algorithm is presented. Section 3 describes the system model, the algorithms performance on the baseline system and the assessment of the robustness of the *PSI* algorithm on variants to the system model. Some concluding remarks are made in Section 4.

2. System Architecture and Load Balancing Algorithms

2.1. System Architecture

The loosely-coupled distributed systems modelled in this study consist of a set of autonomous computers connected by a local area network that exchange information through a message passing mechanism, and operate in a cooperative fashion. A shared file server to support a distributed file system is connected to the same communication device and used to hold all the files and other information needed by the diskless nodes. In this environment the resulting pool of processors can be shared to improve the system performance by relieving overloaded nodes through remote execution of part of their load on less loaded nodes. The essential characteristics of the system under investigation are summarised below. The nodes are homogeneous and assumed to be publically owned, and therefore there is no priority for local jobs over remote jobs of the same category. The job arrival process is assumed to be a Poisson process with a parameter λ_i. The jobs service times and the jobs transfer times are assumed to be exponentially distributed. The mean job service time is E[S]= 1 second. The mean job image size is used as the data unit. The system environment consists of homogeneous users. To assess the sensitivity to the communication delays or job transfer cost, the communication bandwidth is expressed relative to the system job service rate. A system ratio *R= compute rate /communicate rate* is used. The compute rate corresponds to the job processing capacity (jobs/time unit), whereas the communicate rate corresponds to the job transfer capacity (jobs/time unit). The nodes service rate is kept fixed while the communication data transfer rate is varied. A node utilisation level is defined by the following relationship $\rho_i=\lambda_i/\mu_i$, where λ_i represents the node external job arrival rate and μ_i represents the job service rate for node *i*. The system load can be obtained from the formula $\rho=\sum_{i=0}^{i=n}\rho_i$ /n where *n* represents the number of nodes in the system. The jobs are assumed to be executed on the basis of the First-Come-First-Serve local scheduling discipline.

2.2. Load Balancing Algorithms

The algorithms under investigation involve non pre-emptive job transfers and use random polling in their information/negotiation policy. They differ in the job transfer policy which they employ [7]. This includes the following features: dynamic, adaptive, sender-initiated, receiver-initiated, symmetrically-initiated, periodic and aperiodic activation. The load index represents the load level at a node. It corresponds to the number of jobs waiting at a node plus the task currently being executed. The term *load.i* is used at the node which activates the load balancing process whereas *load.j* is used at the node being polled. A threshold *(T)* is provided for each algorithm.

1) Random Algorithm

This is the simplest algorithm. When a node load level crosses the threshold (load.i > T+1), it sends the newly arrived job to a randomly selected node. Only the local information is used.

2) Sender Algorithm

This algorithm is based on the Sender policy [9]. When a node becomes overloaded after the arrival of an external job *(load.i* > T+1*)*, it sequentially polls a set of (L_p) random nodes looking for one whose load is below the *threshold (load.j* < T+1 used for remote job acceptance). If it is so an ACCEPT message is sent back, otherwise it replies with a REJECT message. If the requesting node is still overloaded when the ACCEPT reply arrives, the newly arrived job is transferred, otherwise the job is processed locally. The job is also processed locally when the maximum number of probes (L_p) is reached or if the node is no longer overloaded before the probing is exhausted or if a polling session is already in progress when the job arrives. The probing is sequential and no simultaneous negotiations are allowed.

3) Receiver Algorithm

This algorithm is based on the Receiver policy [9]. If the completion of a job brings the load of a node below the *threshold (load.i* < T+1*)* , this node polls a random set of nodes up to a *probe limit* looking for an overloaded node *(load.j* > T+1*)*. When an overloaded node is found, a non pre-emptive "migration" of a job from the ready queue of the overloaded node takes place. The transfers are receiver-initiated. A special case is the threshold value $T = 0$ *(load.i < 1)* where the idle node initiates the load balancing process.

4) Symmetric Algorithm

This algorithm is a combination of the Sender and Receiver algorithms. It involves a symmetric initiation of the load balancing process [16]. The load balancing strategy is dynamically adjusted based on the current load level of the node by allowing the algorithm to switch automatically between a sender-initiated (SI) policy when the load level goes above the *threshold (load.i* > T+1*)* and a receiver-initiated (RI) policy when the load level drops below the *threshold (load.i* < T+1*)*.

2.3. The PSI Load Balancing Algorithm

The *PSI* algorithm has been designed to address some disadvantages of the Symmetric algorithm. It has been found [4] that the latter involves a larger number of load balancing messages at heavy load levels. This is due to its multiple probing nature. Also a higher negotiation failures result from the concentration of the probing on the time scale. The information exchange policy in Barak and Shiloh algorithm [2] gave us an inspiration for the design of the *PSI* algorithm. The latter emulates a "gas diffusion" process in its negotiation policy as opposed to the information policy used by them. It is symmetrically initiated and uses periodic polling of a single remote and random node. This algorithm aims at fixing (reducing) the L_p parameter at one, and at the same time fixing the frequency of algorithm invocation through a timer parameter *Pt*. This would result in a spreading of the probing messages over time and limit the overhead at high system loads in comparison to the Symmetric algorithm where the algorithm is invoked each time a job departs or starts. For every timer period the node load is checked against the threshold T.

- if exceeding the *threshold (load.i* > T+1*)*, a request is sent to a random node (L_p =1), the node replies with an

ACCEPT message if it is underloaded (*load.j* < T+1), otherwise it ignores the request. The requesting node transfers a job from its transferable jobs queue as a response to an ACCEPT message (*pushed job*), or ignores the request if it is no longer overloaded.

- if below the *threshold* (*load.i* < T+1), a request to receive a job is made to a random node. The chosen node will respond by sending a job from its transferable jobs queue (*pulled job*), or just ignores the message if it is also underloaded (load.j < T+1).
- if the load is normal (*load.i* = T+1), no load balancing is attempted.

This algorithm is adaptive in the sense that based on the current load level, it activates either its sender-initiated (SI) component or its receiver-initiated (RI) component.

3. Experimental Results

In this section the results of a performance study of the selected load balancing algorithms on a baseline system and several variants of the simulated distributed system are reported. Care is needed to minimise the overheads of moving jobs around the system. The main measure of the performance of the load balancing algorithms is the metric: *job mean response time*. This measures the average time that a job spends in the system. The trade-offs involved between reducing the job response time and the overheads incurred in bringing this about, are also discussed.

3.1. Baseline System Model

As shown in [21], a load balancing system modelling involves a representation of the load index, the load balancing algorithm, the workload, and the distributed system attributes. In order to evaluate the performance of load balancing algorithms against particular distributed system attributes and workload models, a simulated system is developed [5]. The testbed software consists of a pre-processing tool (a set of Unix shell scripts and C programs), a Network II.5 [6] based simulator, and a post-processing tool (a set of Unix shell scripts and C programs).

3.1.1. Baseline System Characteristics

The choice of the default values for the baseline system parameters are based on the assumptions made in the literature and the authors own experience of real systems. The values of parameters such as the load balancing algorithms adjustable parameters have been determined through experimentation. The values assigned to job and message parameters are based on the work by Zhou [22], Krueger *et al.* [13], and Mirchandaney *et al.* [16], and accepted as being realistic. Other parameters are based on the performance characteristics for current computer systems. The algorithm execution overhead which represents a small proportion of the overall overhead [22], is assumed negligible. This assumption would not affect the performance results because all the algorithms under investigation are based on similar information and negotiation policies. The default values assigned to parameters of the baseline system are shown below:

number of nodes	10
node service rate	1 job/sec
compute/comms. ratio R	0.13
communication model	token passing protocol
file server I/O time	3.75 msecs evenly distributed
file server I/O overhead	I/O time + network delay
mean job service time	1.0 sec
mean job image size	50 Kbytes
job information size	1 Kbytes
negotiation message size	100 Bytes
message overhead	5.0 msecs
threshold (T)	1
probe limit(L_p)	2
timer period (Pt)	0.4 sec

3.1.2. Model Validation

Three techniques were used for the system model validation. For the no load balancing case (NOLB), the simulator results have been successfully compared to the M/M/1 queues results. Then the algorithms evaluated in [16] were implemented. Similar results were obtained on our simulation software. This involved the tuning of the simulation parameters. Also a verification of the results was undertaken through an extensive checking of the simulation output for "reasonableness" [1]. A modular testing approach was followed during the development of the simulation software.

3.1.3. Validity of the Results

The minimum simulation run length has been experimentally determined for one seed. This is achieved by recording the job mean response time (main performance metric in this study) as a function of the run length for a large time interval at different load levels. Experimentally it was found that a 4000 seconds run length is long enough to ensure the effect of initial conditions and "left over" jobs can be ignored, and a reliable ranking of the algorithms is obtained. The minimum run length of 4000 seconds corresponds to a generation of about 3,600 jobs population on each host at a very heavy arrival rate for our workload model. To increase the confidence in the results further, the same experiments are repeated at $\rho = 0.8$ and $\rho = 0.9$ for a 4000 seconds simulation run length but for 9 different random number seeds. This has been done to smooth out the perturbations caused by the statistical nature of the random number generator. On these replications the 95 percent confidence interval has been evaluated using the Minitab [2] data analysis software, from which the margin of error was computed. It has been found that on the average for this run length the mean response time stabilises and the percentage of error is less than 3 percent for a system load $\rho \leq 0.8$, and less than 5 percent for a system load $\rho = 0.9$. For subsequent experiments one seed is chosen. The confidence levels in the job response time numerical results are shown below:

System Load	95% Confidence Interval	Margin of Error (%

[2] Minitab is a trademark of Minitab Inc.

$\rho \leq 0.8$	±0.077 (all algorithms average)	3
	±0.024 (PSI)	1.69
$\rho = 0.9$	±0.176 (all algorithms average)	5
	±0.065 (PSI)	3.65

Further details on the validity of results can be found in [4]. This statistical analysis of the simulation results shows that the estimates of the performance are sufficiently accurate to make the use of the simulation model and the conclusions drawn on the performance ranking of the *PSI* and other algorithms, reliable.

3.2. Tuning the Load Balancing Algorithms

To obtain a correct ranking of the load balancing algorithms, each algorithm has a number of parameters that have been tuned for optimal performance. For the considered algorithms (Random, Sender, Receiver, and Symmetric), the essential parameters to be tuned are the *probe limit* (L_p) and the *threshold (T)*. An optimal probe limit value of 2 was obtained experimentally. Only a marginal benefit would result from an increase of the probe limit. A theoretical treatment of this issue is done by Eager *et al.* [10].

The local load threshold used by the load balancing algorithm is a very important parameter. Its optimal value depends on the communication delay. For a compute/communicate ratio R= 0.13 (short delays) a value of T=1 was found optimal (see Figure 1). A large value of the threshold (e.g. T=99) leads to a response time worse than the NOLB case. For such a threshold the receiver-initiated component of the Symmetric algorithm is activated practically for every job arrival causing an excessive overhead.

For the *PSI* algorithm, the parameter L_p is fixed to one by design. In addition to the threshold value, another parameter called timer period *(Pt)* is to be tuned. Figure 2 shows the effect of the timer period on the performance of the *PSI* algorithm. For jobs with 1.0 second average service time, the useful interval of values for the timer period is 0.2 to 1.0 second. Although Pt = 0.2 second is the optimal value for the baseline system, the suboptimal value 0.4 second is chosen because for 0.2 second the system operates near the saturation. For values lower than 0.2 second, the algorithm leads to a deterioration of the performance. This behaviour first appears at the heavy load levels, then at the lower load levels as Pt is shortened. For a large value of Pt (e.g. Pt > 50 seconds) the algorithm becomes less and less activated till at the limit no load balancing takes place and the algorithm behaves similarly to NOLB case.

3.3. Performance of the *PSI* on the Baseline System

The performance factors considered in this section are the load balancing algorithm and the load level. Two performance metrics are considered namely the job mean response time and the level of load balancing overhead incurred.

1) Job Mean Response Time

The results for the baseline system with homogeneous jobs are shown in Figure 3. The following remarks can be made:

- The *PSI* algorithm (i.e. periodic version of Symmetric) produces the lowest mean job response time. This algorithm results in fewer number of messages and job movements than the Symmetric one. The advantage of *PSI* is due to its periodic single probing of remote nodes.
- The Random algorithm which has the lowest overhead, since no system information collection is needed, has the poorest performance due to large job movements and high percentage of wrong job transfers.
- While Sender performs better than Receiver at low to moderate load levels, the latter is better at at heavy load levels. This is because all the nodes become heavily loaded, it is more difficult to find an idle or underloaded node through a sender-initiated load balancing. These results confirm the findings of Eager *et al.* [9].
- The Symmetric algorithm, which is a combination of Sender and Receiver, does well over all the whole range of load levels. However it involves a higher number of load balancing messages and job movements. This tends to increase the percentage of CPU utilisation significantly.

A reduction of the job mean response time of 80 percent is possible for the baseline system under very heavy load levels. For a load level between 0.65 to 0.9, the performance ordering for the algorithms is: *PSI*, Symmetric, Receiver, Sender, Random.

2) Load Balancing Overheads

A load balancing scheme consumes CPU cycles for the execution of its policies and adds message traffic onto the communication device. To assess the average CPU utilisation due to the load balancing activities, the percentage of processor busy time of individual nodes is monitored for each algorithm and compared to the NOLB case. The percentage average increase is then computed. From the results shown in Figure 4, it can be concluded that the best performing algorithms have higher overheads. Except for the Receiver algorithm whose activities are slowed down at high load levels, the level of overhead increases with the load level. The increase in the percentage of communication device utilisation is less than two percent for all load balancing algorithms and at all load levels. From this we conclude that on a diskless model of distributed systems, the load balancing overhead is mainly on the CPU utilisation.

3.4. Robustness of the PSI Algorithm

The experiments carried out on the baseline system have shown that the *PSI* algorithm produces the highest improvement of the job mean response time and that the algorithms ranking based on the increase in the CPU utilisation is in the reverse order of the improvement in job response time. To assess the sensitivity of the *PSI* algorithm to changes in the system attributes, the algorithms are re-evaluated under the following system variants.

3.4.1. Communication Bandwidth

In earlier studies on the performance of load balancing algorithms, it has been commonly assumed that a large communication bandwidth is available, so there is no contention on the communication device. However, this is not realistic since in present day technology the processor speed is increasing at a faster rate than the bandwidth of the communication network. Therefore it is worthwhile to investigate the performance of the load balancing algorithms under a smaller communication bandwidth. The experiments carried out on the baseline system, were repeated using a compute/communicate ratio R= 0.4. This makes it possible

to compare the effect of the compute/communicate ratio on the algorithms performance. A higher value of the threshold was found more appropriate when a large compute/communicate ratio is used (T=2 for R= 0.4).

The results of using this slower communication device are shown in Figure 5. It is possible to draw the following conclusions.

- The relative performance order of the algorithms is unchanged. However, for the Symmetric algorithm the job mean response time tends to saturate at very heavy load level. This can be explained by the large number of load balancing messages inherent to this algorithm.
- Although the relative ordering of the algorithms is not affected, the communication bandwidth does affect the level of performance improvement. Due to the longer communication delay, the level of improvement of all the algorithms drops by a maximum of 10 percent. Even under the NOLB case, the job mean response time degrades because it takes longer to access the file server.
- The performance curves of all the algorithms tend to cluster making the choice of the load balancing algorithm less relevant. An exception is the *PSI* algorithm which maintains a more significant improvement of the mean response time. Another effect is a reduced level of job movement and a slight increase in the percentage of wrong job transfers.

3.4.2. Heterogeneous Jobs

The Poisson arrival- exponential service demands workload is commonly assumed as the default workload model. Zhou [22] and Leland *et al.* [14] have shown that exponential distributions approximate poorly to process service demands, instead hyperexponential distributions are to be used. In this experiment a model with hyper-exponential service demands is investigated. The workload consists of homogeneous users but with heterogeneous jobs. A 70/30 proportion of short/long jobs and non-selective transfers of jobs is assumed. For hyper-exponential job service times the Round Robin local scheduling discipline [17] is more appropriate. On this experiment a larger timer period (Pt= 1.6 secs) was used for better performance of the *PSI* algorithm. From the results shown in Figure 6, it can be concluded that the performance order of the algorithms remains the same as for homogeneous jobs.

3.4.3. System Size

An important feature of any load balancing algorithm is that performance improvements are maintained as the number of processors in the system increases. This is referred to as scalability and some scalability principles have been reviewed in [3]. For scalability it is important to avoid algorithms that use system wide information. Instead it is better to use algorithms that make their decisions based on a small subset of the nodes.

To assess the scalability of the results obtained, some additional experiments were conducted. The performance of the load balancing algorithms under a heavy load level for 5, 10, 20 nodes (with 0.06, 0.13, and 0.26 compute/communicate ratio respectively) is shown in Figure 7. From these results it can be seen that the relative performance ordering of the algorithms holds for different system sizes and that the level of performance improvement increases with the system size. This agrees with the conclusions drawn by Zhou [22]. The effect of the system size was evaluated [4] for the *PSI* and Symmetric algorithms over the whole range of load levels. The best results were obtained for a 20 nodes system. We anticipate that these results remain valid for system size of few tens of nodes larger, but as shown in [15] and [22], even for scalable algorithms, the performance becomes insensitive to the number of nodes as the number of nodes increases beyond a certain limit.

3.4.4. Disk-based File System

The distributed system considered in this section consists of a set of identical autonomous nodes. Each node has its own local file system and is connected to a broadcast communication device. In this environment load balancing involves the actual transfer of the complete job information (i.e. programs, files) to the remote host, and the return of the results data and files to the job originating host. Except for the file server related experiment, which does not apply to the disk-based system, the experiments carried out on the diskless system model were repeated on a disk-based system model. Apart from the file system structure which is changed to a disk-based model, all the characteristics of the system under study are the same as those used in the baseline system described in Section 3.1. Since no file server is used, all I/O operations are handled by a local disk. The time to service an I/O operation is assumed evenly distributed and fixed to 20 msecs. The results obtained under the disk-based system model are shown in Figure 8. It is interesting to note that there is no significant difference between the performance ordering of the algorithms for the diskless and disk-based models of the baseline system. These results also support the findings by Mirchandaney *et. al* [16], in terms of both the relative ordering of the algorithms and the level of performance improvement obtained. It is to be noted that these results may not generalise to all practical systems because only non pre-emptive transfers were considered.

3.4.5. Other Experiments

In the baseline system a token passing *communication protocol* was assumed. To assess the effect of the choice of the communication protocol, the performance of the two more promising algorithms (i.e. *PSI* and Symmetric) is evaluated [4] under "First-Come First-Serve" and "CSMA/CD" communication protocols. It was carried out under both large (R= 0.4) and small (R= 0.13) compute/communicate ratios. No significant effect on the mean response time was noticed. This can be explained by the low device utilisation level which was less than 50 percent for slow device and less than 20 percent for fast device, and the similar load applied by all the nodes on the communication device. For this level it can be justified to assume that there is no conjestion on the communication device and that the communication protocols perform similarly.

The effect of the *file server speed* on the mean response time is assessed [4] for the *PSI* and Symmetric algorithms. The file server speed attribute used is the fixed overhead time which it takes to execute an I/O operation. To remove any side effect due to the conjestion on the communication device, short communication delays are assumed. As would be expected, there is a large range of server speeds which has little effect on the job mean response time. However, when the file server takes over 30 msecs to execute an I/O operation, a sharp increase of the mean response time occurs.

4. Conclusion and Further Work

The performance level provided by the *PSI* algorithm, and its robustness over a range of system attributes and workload makes it a very promising algorithm. However, this algorithm does involve a higher level of wrong job transfers. Also some care is needed in interpreting these results because the timer period of this algorithm has been tuned for optimal performance for the simulated system.

This study has demonstrated the value of simulation in the design of load balancing algorithms. It has enabled a range of algorithms to be evaluated; and a new algorithm (*PSI*) has been proposed. For example this has enabled us to study more realistic file system structures. Our investigations suggest that in future experiments it is not necessary to consider the *communication protocols* and the *heterogeneous workload* model. The same conclusions can be drawn from a homogeneous model if only the ordering of the algorithms is sought. However, the modelling of other system attributes can affect the quality of the results significantly. The correct representation of the *communication bandwidth* and the *algorithm parameters* tuning (i.e. *T, Pt*) is important to get an accurate ordering of the algorithms. Assuming non pre-emptive transfers, any *file system structure* is sufficient for load balancing algorithms ordering purposes. However, to get a clearer idea on the level of performance improvement the modelling of the specific file system structure is necessary. In order to realise the potential benefits of load balancing, for the diskless file system structure a minimum *file server speed* is needed to avoid a major I/O bottleneck (i.e. incorrectly configured system). The algorithms evaluated in this study adhere to the scalability principles. For these algorithms the ranking is not affected by the *system size* (e.g. ≥10 nodes). However, the level of performance improvement increases with the increase of the system size up to some point, then it levels off. When a large number of nodes are available, clustering of the nodes to reflect some organisation groupings is the way forward to improve the performance through load balancing.

References

1. J. Banks and J. S. Carson, *Discrete-Event System Simulation,* Prentice-Hall International (1984).
2. A. Barak and A. Shiloh, "A Distributed Load-balancing Policy for a Multicomputer," *Software- Practice and Experience* **15** (9) pp. 901-913 (September 1985).
3. A. Barak and Y. Kornatzky, "Design Principles of Operating Systems for Large Scale Multicomputers," pp. 104-123 in *International Workshop on Experiences with Distributed Systems*, ed. J. Nehmer ,Springer Verlag (September 1987).
4. K. Benmohammed-Mahieddine, "An Evaluation of Load Balancing Algorithms for Distributed Systems," PhD Thesis, Leeds University, U.K. (February 1992).
5. K. Benmohammed-Mahieddine, "A Testbed for a Study of Load Balancing Algorithms on Distributed Systems," *Proceedings AMSE Internat. Conference on "Systems", London (UK)* **2** pp. 131-142 (September 1-3, 1993).
6. CACI, *NETWORK II.5 User's Manual Version 5.0,* CACI Products Company (August, 1989).
7. T. L. Casavant and J. G. Kuhl, "A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems," *IEEE Transactions on Software Engineering* **14** (2) pp. 141-153 (February 1988).
8. P. Dikshit, S. K. Tripathi, and P. Jalote, "SAHAYOG: A Test Bed for Evaluating Dynamic Load-sharing Policies," *Software- Practice and Experience* **19** (5) pp. 411-435 (May 1989).
9. D. L. Eager, E. D. Lazowska, and J. Zahorjan, "A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing," *Proc. of the 1985 ACM SIGMETRICS Conference on Measurement and Modelling of Computer Systems*, pp. 1-3 (August 1985).
10. D. L. Eager, E. D. Lazowska, and J. Zahorjan, "Adaptive Load Sharing in Homogeneous Distributed Systems," *IEEE Transactions on Software Engineering* **SE-12** (5) pp. 662-675 (May 1986).
11. C. Jard, J. F. Monin, and R. Groz, "Development of Veda, a Prototyping Tool for Distributed Algorithms," *IEEE Transactions on Software Engineering* **14**(3)(March 1988).
12. P. Krueger and M. Livny, "The Diverse Objectives of Distributed Scheduling Policies," *IEEE Proc. 7th Internat. Conf. on Distributed Computing Systems*, (September 1987).
13. P. Krueger and M. Livny, "A Comparison of Preemptive and Non-Preemptive Load Distributing," *IEEE Proc. 8th International Conference on Distributed Computing Systems*, pp. 123-130 (June 1988).
14. W. E. Leland and T. J. Ott, "Load-balancing Heuristics and Process Behavior," *Proc. of the ACM SIGMETRICS Conference*, pp. 54-69 (May 1986).
15. M. Livny, "The Study of Load Balancing Algorithms for Decentralized Distributed Processing Systems," Computer Sciences Technical Report #570, University of Wisconson-Madison (December 1984).
16. R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Analysis of the Effects of Delays on Load Sharing," *IEEE Transactions on Computers* **38** (11) pp. 1513-1525 (November 1989).
17. I. Mitrani, *Modelling of Computer and Communication Systems,* Cambridge University Press (1987).
18. C.G. Rommel, "The Probability of Load Balancing Success in a Homogeneous Network," *IEEE Transactions on Software Engineering* **17** (8)(September 1991).
19. J. A. Stankovic, "Simulations of three Adaptive, Decentralized Controlled, Job Scheduling Algorithms," *Computer Networks* **8** pp. 199-217 Elsevier Science, (1984).
20. Y. T. Wang and R. J. T. Morris, "Load Sharing in Distributed Systems," *IEEE Transactions on Computers* **C-34** (3) pp. 204-217 (March 1985).
21. S. Zhou and D. Ferrari, "A Measurement Study of Load Balancing," *IEEE Proc. 7th Internat. Conf. on Distributed Computing Systems*, (September 1987).
22. S. Zhou, "A Trace-Driven Simulation Study of Dynamic Load Balancing," *IEEE Transactions on Software Engineering* **14** (9) pp. 1327-1341 (September 1988).

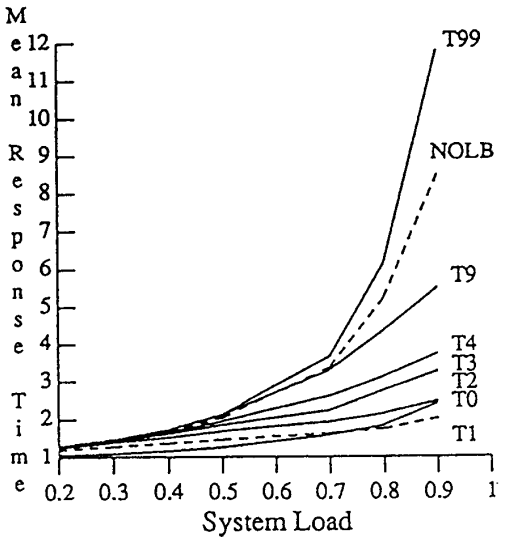

Figure 1 Effect of Threshold (T) on Symetric Algorithm Performance

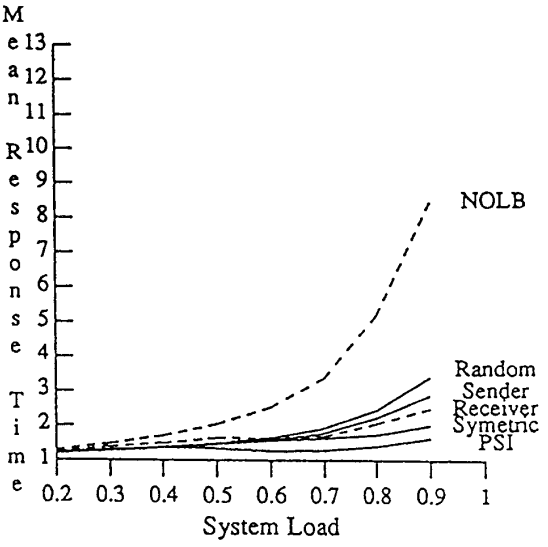

Figure 3 Performance under Small Compute/Communicate Ratio (R= 0.13)

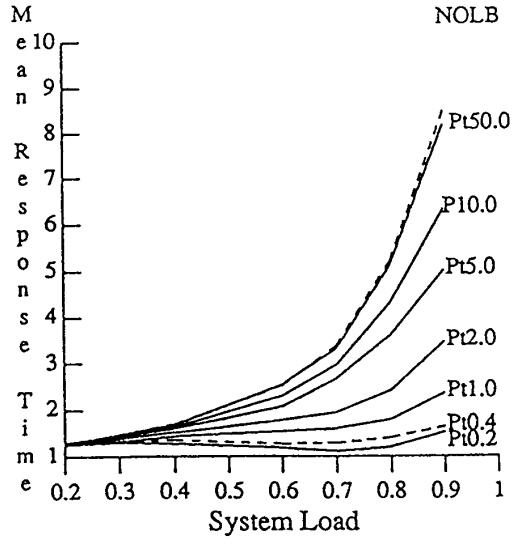

Figure 2 Effect of Timer Period (Pt) on PSI Algorithm Performance

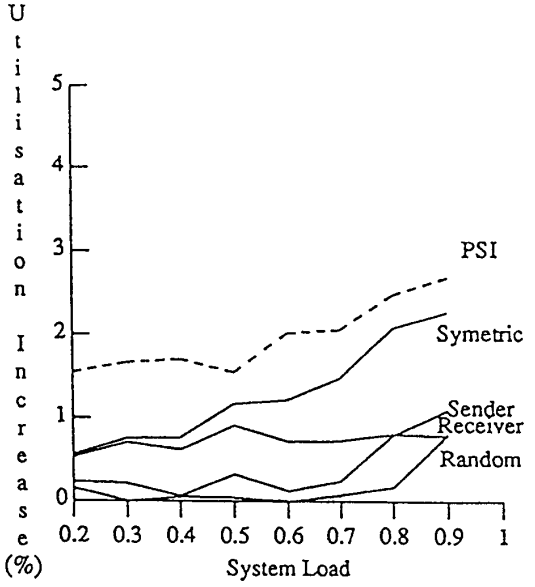

Figure 4 Load Balancing Overhead for the Baseline System (R= 0.13)

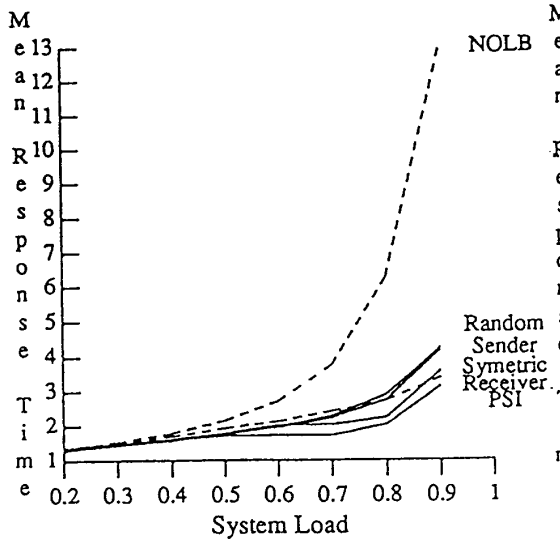

Figure 5 Performance under Large Compute/Communicate Ratio (R= 0.4)

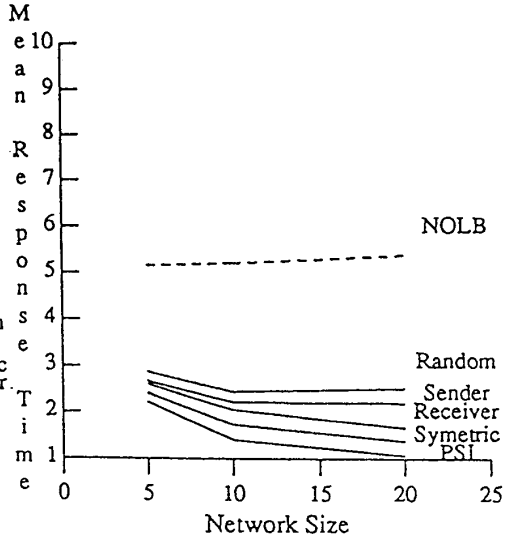

Figure 7 Scalability of Algorithms (Diskless Model)

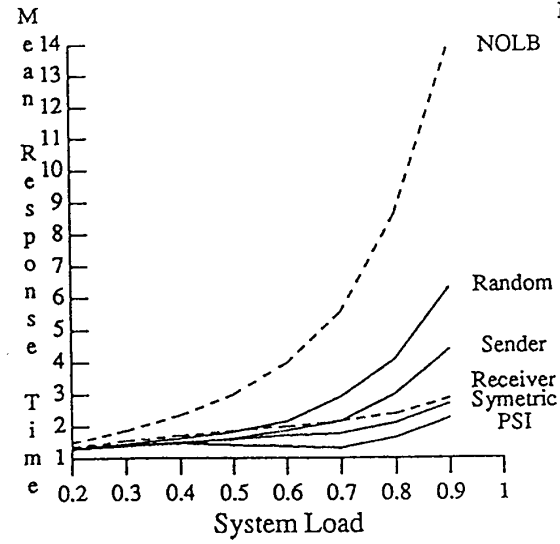

Figure 6 70/30 Jobs Proportion with Non-selective Transfer

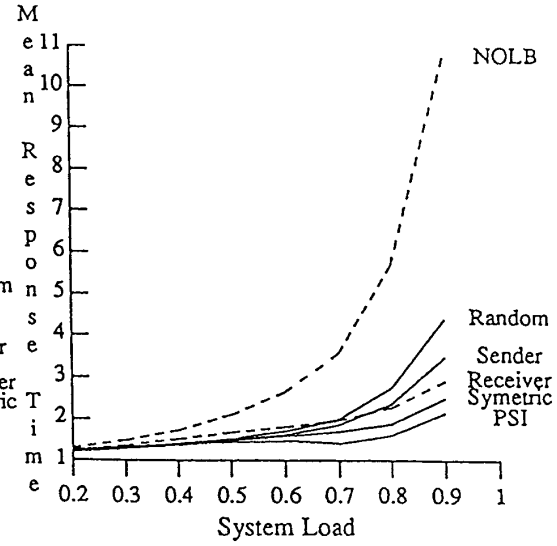

Figure 8 Performance for Disk-based File System Structure (R= 0.13)