Performance Modeling and Analysis of Load Balancing Policies with Priority Queueing

Rong-Chau Liu
Chung Shan Institute of Science and Technology, Lungtan, Taiwan, R.O.C.

Sheng-De Wang
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, R.O.C.

In this article, we study an adaptive load-balancing algorithm in the homogeneous distributed systems in which only local status information is used. The parameters affecting the performance of the load-balancing algorithm are investigated. To analyze the effects of service disciplines on load balancing, we study two classes of service disciplines, impartial discipline and partial discipline. In impartial discipline, all tasks in the system are treated alike. Partial disciplines divide tasks into two classes, local tasks and remote tasks, and then assign different priorities to them. Five partial disciplines with different priority assignment rules are compared. The numerical results are presented and used to shed light on the characteristics of the load-balancing process.

1. INTRODUCTION

Distributed computer systems have many advantages, including the capability to share processing of tasks in the event of overloads and failures, transparency, and modularity. In a distributed system a task might wait for service at the queue of one server while at the same time another server capable of serving the task is idle. This phenomenon was clearly demonstrated by Livny and Melman [1], who showed that in a network of homogeneous, autonomous nodes, there is a high probability that at least one node is idle while tasks are queueing at some other node or nodes over a wide range of network size and node utilizations. A load-balancing policy designed to minimize the mean turnaround time of the task will tend to prevent the system from reaching such a state. System performance can be improved by transferring workload from heavily to lightly loaded nodes.

Load-balancing policies may be either static or adaptive. Static policies only use information about the average behavior of the system; transfer decisions are independent of the actual current system state. Numerous static load-balancing policies have been proposed. Stone [2, 3] and Bokhari [4] examined the deterministic algorithms of task assignments. Tantawi and Towsley [5] developed a technique to find optimal probabilistic assignment rules.

The principal advantage of static policies is their simplicity: there is no need to acquire and maintain information on the system state. By contrast, adaptive policies are more complex, since they use information of the current system state in making transfer decisions. Several adaptive load-sharing policies have been reported. Two strategies for adaptive load sharing with distributed control were investigated by Eager, Lazowska, and Zahorjan [6]. Wang and Morris [7] compared a number of server- and source-initiative adaptive algorithms.

The motivation behind this article is to investigate the effects of processor service disciplines on load-balancing policy, which have not been discussed in the reports mentioned above. First, we study a simple adaptive load-balancing policy and analyze the effects of its parameters on system performance. Owing to different service disciplines, tasks in the

Address correspondence to Professor Rong-Chau Liu, Chung Shan Institute of Science and Technology, P.O. Box 90008-9-3, Lungtan, Taiwan, R.O.C.

© Elsevier Science Publishing Co., Inc.
655 Avenue of the Americas, New York, NY 10010

system are divided into two classes: local tasks, which are processed at their site of origination, and remote tasks, which are processed at some other sites in the system after being transferred through an interconnection network. We also examine the effects of assigning different priorities to two classes of tasks. Five such policies are developed, ranging from one favoring local tasks to one favoring remote tasks. Analytic queueing model and simulation results are used to study the interdependence between the system parameters and the behavior of these load-balancing policies.

The remainder of this article is organized as follows. In section 2, we briefly describe the system architecture. Section 3 comprises the description of load-balancing policies and the mathematical analysis of the decomposition approximation model. In section 4, we describe the numerical and simulation results of this research. Section 5 concludes the article.

2. SYSTEM MODEL AND ASSUMPTIONS

We present a distributed system as a collection of N identical nodes. The nodes are connected by an interconnection network (IN). The system architecture is shown in Figure 1. Each node in the system consists of a processor, a network interface processor (NIP), and a queue. The NIP performs the load-balancing algorithm and is responsible for communicating with the interconnection network. Tasks entering the node from the external world (λ) or from the interconnection network (γ) are selected by the NIP, based on a specific load-balancing policy. Then the accepted tasks enter the queue and wait for the service of the processor; the remaining tasks are transferred to the interconnection network.

There are several assumptions to simplify the system model, which are stated as follows.

Homogeneity. The nodes are identical and are subject to external task arrival Poisson processes with the same rate λ; the service time of the processor is exponentially distributed with mean $1/\mu$.

When a task is transferred to the interconnection network for remote processing, the task incurs a communication delay. This delay includes the communication cost, which is a cost between the NIP and the interconnection network and includes packing data, transmitting data, receiving data, unpacking data, and the transmission delay in the interconnection network. We lump these effects together as communication overhead (Coh), which is incurred by the load-balancing algorithms when a task is transferred to the other nodes through the IN. We also model the communication channel as a single-server queueing system whose service time is an exponentially distributed random variable with mean $1/\mu_{\text{in}}$.

The processing cost of NIP is much smaller than that of the processor. That is, the load-balancing algorithm performed by the NIP does not interfere with the processor; therefore, the processing cost of the NIP is ignored.

Decomposability. We assume, in the steady state, that the task arrivals from the interconnection network are of Poisson type and the interarrival time of the transferred task is an exponentially distributed random variable with mean $1/\gamma$. Based on this decomposition approximation, we can decompose the system model so that the model for each node can be independently analyzed.

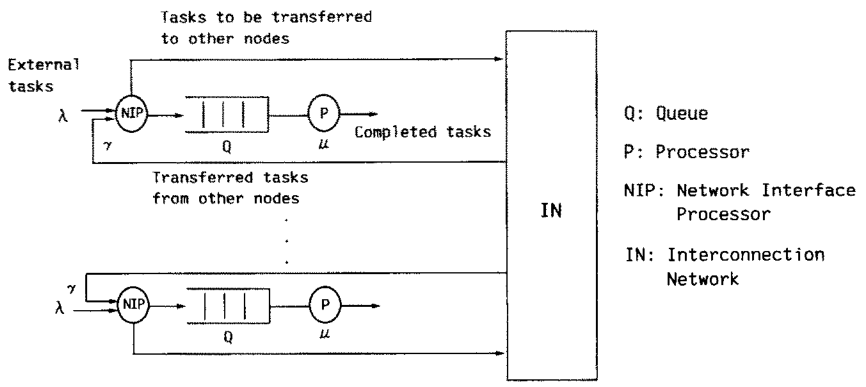
Tasks to be transferred to other nodes
External tasks
λ
γ
NIP
Q
P
μ
Completed tasks
Transferred tasks from other nodes
IN
Q: Queue
P: Processor
NIP: Network Interface Processor
IN: Interconnection Network

Figure 1. System queueing model.

3. LOAD-BALANCING POLICIES AND QUEUEING ANALYSIS

3.1 Load-Balancing Algorithm

A distributed load-balancing algorithm is composed of two main elements, the probing element and the decision element. The decision element determines when, from where, and to whom to transfer a waiting task. The decision is made according to the current available information on the state of the system. The probing element collects this status information for the decision element. We consider here a simple and straightforward algorithm. The probing element only collects the local status (queue length) of the node, and the decision element accepts rejects an arriving task, depending on this local status information and a predefined threshold (Th) value. The algorithm is as follows: when an arriving task enters the node, if the NIP finds the local queue length, including tasks waiting in the queue and the task in service, is Th, then this task is transferred through the interconnection network to a randomly selected node in the system with equal probability. Otherwise, this task is accepted and goes into the queue.

This algorithm can cause instability. That is, in a heavily loaded network, when all nodes in the system have a queue length larger than the Th, the system will enter a state in which the task arrivals are always transferred back and forth within the network and no processor accepts them. This instability phenomenon can be overcome by adding an appropriate control policy to the decision element. The simple control policy we adopt here is to restrict the number of times that a task can be transferred to a predefined transfer limit (Tl) value [8]. Then the load balancing algorithm is updated as follows.

When an arriving task enters the node, if the NIP finds the local queue length of the node is lower than the Th or the number of times that the task has been transferred and a message carried by the task is equal to the Tl, then this task is accepted. Otherwise, the transfer count of the task is increased by one and the task is transferred to a randomly selected node in the system. The initial value of the transfer count of a task from the external world is set at 0.

3.2 Service Disciplines

Another question that arises in considering the behavior of the refined algorithm is how the destination node should treat an accepting transferred task from the interconnection network: tasks may be given different service attributes in the local site or in the remote site. We consider two classes of service disciplines and describe them in the following discussion.

3.2.1 Impartial discipline. In this discipline, the destination node treats an arriving transferred task as a task originating at that node. Then all accepted tasks, assigned with the same priority, enter the queue and follow the first come, first-served scheme to wait for the service of the processor. We call this an F1 discipline.

3.2.2 Partial disciplines. In these disciplines, owing to the different service attributes, tasks are divided into two classes—local tasks, referring to tasks processed at their own site, and remote tasks, which processed at some other site in the system after being transferred through the interconnection network. Local tasks come either from the external world directly or from the interconnection network. Tasks transferred to another node because of the limitation of the local task Th are later transferred back to the same node because of the effect of the Tl. Since the system has two classes of tasks, we define LQ and RQ, respectively, as random variables denoting the number of local tasks and the number of remote tasks at a node; Th_l and Th_r are the values of the threshold of the local task and remote task, respectively. Two types of partial disciplines use different priority assignment schemes as defined below.

1. Priority Queueing [9] Disciplines
   a. Local tasks first (LTF) policy
      i. If an arriving task from the external world finds the local task queue length of the node is Th_l, then the transfer count of the task is incremented by one and the task is transferred to a randomly selected node in the system with equal probability. Otherwise, it is processed locally.
      ii. Arriving tasks from the interconnection network can be divided into two cases. Local tasks: if the transfer count is Tl, then this task is accepted. Otherwise, its transfer count is increased by one and it is transferred to a randomly selected node. Remote tasks: if the remote task queue length of the node is lower than Th_r or the trans-

fer count is Tl, then this task is accepted for processing locally. Otherwise, its transfer count is increased by one and it is transferred to a randomly selected node.

   iii. Tasks at each node are processed according to a priority discipline with nonpreemption allowed. Tasks having the same priority are processed in a first come, first served manner.
   iv. Local tasks are given higher priority than remote tasks.

b. Remote tasks first (RTF) policy
   i. Same as LTF policy.
   ii. Same as LTF policy.
   iii. Same as LTF policy.
   iv. Remote tasks are given higher priority than local tasks.

c. Long queue first (LQF) policy
   i. Same as LTF policy.
   ii. Same as LTF policy.
   iii. Same as LTF policy.
   iv. Local tasks are given higher priority than remote tasks when LQ $\geq$ RQ. Otherwise, remote tasks are given higher priority than local tasks.

d. Short queue first (SQF) policy
   i. Same as LTF policy.
   ii. Same as LTF policy.
   iii. Same as LTF policy.
   iv. Local tasks are given higher priority than remote tasks when LQ $\leq$ RQ and LQ $>$ 0. Otherwise, remote tasks are given higher priority than local tasks.

2. First come, first served (F2) policy
   i. Same as LTF policy.
   ii. Same as LTF policy.
   iii. Tasks at each node are assigned with the same priority and follow the first come, first served scheme to wait for the service of the processor.

3.3 Queueing Analysis

Based on the decomposition approximation mentioned above, we can simplify the queueing analysis by assuming that the state of each node is stochastically independent of that of any other node. Each node can then be analyzed in isolation. The effect of the remainder of the system on an individual node is represented by an arrival process of transferred tasks from the interconnection network. Because the network is homogeneous, system performance measures can be obtained by analyzing a model of any individual node. We will discuss the accuracy of this decomposition approximation at the end of section 4.

Owing to the complexity, the mathematical analysis is only tractable for cases when Tl = 1. The independent analysis of each individual policy will be described in the following subsections.

3.3.1 Tl = 1: Impartial discipline (F1). In this case, p (the probability that the portion of transferred tasks have their transfer count equal to Tl) = 1. Let E[Q] be the mean queue length of the node. We can show that

$$\mathrm{E[Q]} = \left\{ (\rho + \rho_1)\left[\frac{1-(\rho+\rho_1)^{\mathrm{Th}}}{(1-\rho-\rho_1)^2} - \frac{\mathrm{Th}(\rho+\rho_1)^{\mathrm{Th}}}{1-\rho-\rho_1}\right] + \rho_1(\rho+\rho_1)^{\mathrm{Th}}\left[\frac{\mathrm{Th}}{1-\rho_1} + \frac{1}{(1-\rho_1)^2}\right]\right\} P_0 \tag{1}$$

where

$$\rho = \frac{\lambda}{\mu},\ \rho_1 = \frac{\gamma}{\mu},\ \text{and } P_0 = \left[\frac{1-(\rho+\rho_1)^{\mathrm{Th}}}{1-\rho-\rho_1} + \frac{(\rho+\rho_1)^{\mathrm{Th}}}{1-\rho_1}\right]^{-1}.$$

Let E[T] be the mean task response time. From Little's formula [10] and the fundamental queueing theory [11], we have

$$\mathrm{E[T]} = \frac{\mathrm{E[Q]}}{\lambda} + \frac{\gamma}{\lambda}\mathrm{E[Coh]}, \tag{2}$$

where $\mathrm{E[Coh]} = \dfrac{1}{\mu_{\mathrm{in}} - N\gamma}$ denotes the mean communication overhead. The γ (mean transferred task

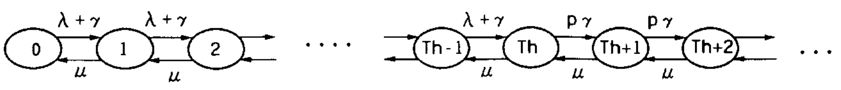

Figure 2. Markov chain of F1 policy (Tl = 1).

arrival/departure rate) is a function of λ, Th, and Tl, which can be calculated by a recursive substitution technique (see Appendix A).

3.3.2 Tl = 1: Partial disciplines. In this case, the remote task threshold, Th_r, has no effect on the load-balancing policy. As the decomposition approximation stated above, each node in the system under load-balancing policies can be represented by a continuous time and discrete state Markov chain. Figure 3 shows the state transition diagrams for a node executing four priority queueing disciplines, respectively, when $\mathrm{Th}_l = 3$ and $\mathrm{Tl} = 1$. The state (l, r) denotes that there are l local tasks and r remote tasks in the node, respectively, and $P(l, r)$ is the probability that the node is in the state (l, r).

Let E[LQ] and E[RQ] be the mean number of local tasks and mean number of remote tasks in the node, respectively. For the LTF policy, we can find the closed-form expression for γ, E[LQ], and E[RQ]. Using the Z transform technique (see Appendix B), there are

$$\gamma = \frac{\lambda\rho^3}{1 + \rho + \rho^2 + \rho^3}, \tag{3}$$

$$\mathrm{E[LQ]} = \frac{\rho + 2\rho^2 + 3\rho^3}{1 + \rho + \rho^2 + \rho^3}, \text{ and} \tag{4}$$

$$\mathrm{E[RQ]} = \frac{\rho_l(\rho^6 + 3\rho^5 + 6\rho^4 + 10\rho^3 + 6\rho^2 + 3\rho + 1)}{(1 + \rho + \rho^2 + \rho^3)\left[1 - \rho_l(1 + \rho + \rho^2 + \rho^3)\right]}. \tag{5}$$

The other three priority queueing disciplines can be analyzed by the matrix-geometric solution technique [12]. This technique has been used in the performance modeling and analysis of the nonproduct form queueing networks [13], which yield exact solutions of E[LQ] and E[RQ] for each node (see Appendix C). Thus, the mean task turnaround time (E[T]) is

$$\mathrm{E[T]} = \frac{\mathrm{E[LQ]} + \mathrm{E[RQ]}}{\lambda} + \frac{\gamma}{\mu}\mathrm{E[Coh]}, \tag{6}$$

where E[Coh] is the same as in Equation 2. As mentioned above, the value of γ can be calculated by a recursive substitution technique.

3.3.3 Tl > 1. In this case, for F1 discipline, $p < 1$. It is then difficult to find the exact solution of p and to solve the birth-death process. We obtain the performance measures from simulations. In the same way, the Markov chain of the LTF policy ($\mathrm{Th}_l = \mathrm{Th}_r = 3$ and $\mathrm{Tl} > 1$) is shown in Figure 4. In this figure, p is the probability that the portion of transferred

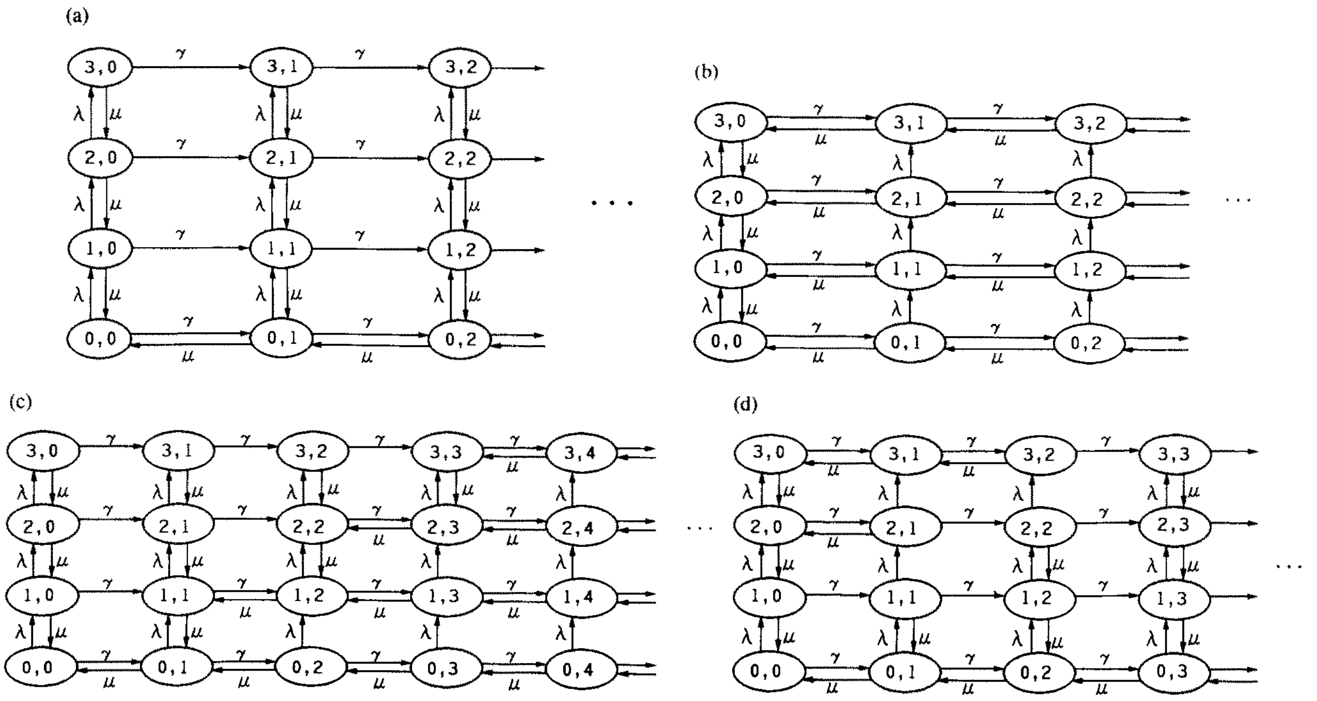

Figure 3. a, Markov chain of LTF policy (Tl = 1, $\mathrm{Th}_l = 3$). b, Markov chain of RTF policy (Tl = 1, $\mathrm{Th}_l = 3$). c, Markov chain of LQF policy (Tl = 1, $\mathrm{Th}_l = 3$). d, Markov chain of SQF policy (Tl = 1, $\mathrm{Th}_l = 3$).

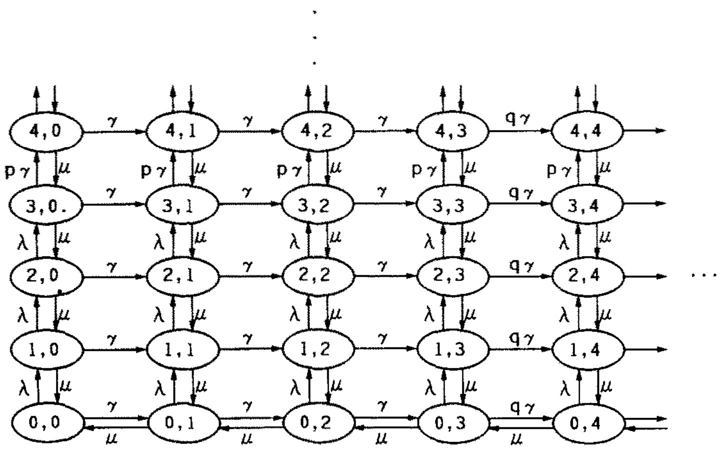

Figure 4. Markov chain of LTF policy (Tl > 1, $\mathrm{Th}_l = 3$).

local tasks have their transfer count equal to Tl and q is the probability that the portion of transferred remote tasks have their transfer count equal to Tl. We note that $p + q < 1$. The exact solution of this Markov chain is too complicated to be obtained by mathematical methods. Thus, we again obtain the performance measures from simulations for partial priority queueing disciplines.

3.3.4 F2 discipline. Even when the decomposition approximation is valid and Tl = 1, the analytic analysis for the F2 discipline is not tractable. Thus, we obtain the performance measures of this discipline from simulations.

4. RESULTS

In this section, we present the performance measures of the load-balancing policies stated in section 3. As mentioned above, some situations, such as Tl > 1, are too complicated to analyze using the mathematical methods, simulation is the most popular approach for the analysis of the performance of a computer system because of its generality and simplicity. Therefore, we use results from simulations to depict the actual sample means. At the end of this section, we will discuss the relationship between simulation results and those of the mathematical analysis.

The simulation model is constructed on a SUN/4 workstation using PAWS (performance analyst's workbench system) simulation language [14]. Over 100 cases have been simulated. The lengths of the simulation runs are determined over a considerable range of parameter values by a suitable compromise between numerical accuracy and the completion time for simulations. All results have confidence intervals [15] of $\leq 5\%$ a 90% confidence level. The key performance metric of these experiments is the mean turnaround time of tasks (Rt). Furthermore, in the following experiments, we let the mean task service time, $1/\mu$, equal one unit; all measurements of the turnaround time are in terms of this unit.

Since many parameters, such as the system load (ρ), threshold (Th), transfer limit (Tl), communication overhead (Coh), etc., can affect the performance of load-balancing policies, we adopted a "one factor at a time" approach in our simulation experiments; that is, each experiment involves the varying of one parameter while keeping the others at a constant value. The variable parameter in each experiment is represented by the abscissa of the corresponding result graph, whereas the values of the fixed factors are included in the figure legend. The number of nodes in the system (N) is fixed at 10 in all simulation runs. In general, although it is possible for the communication overhead of the interconnection network to be $> 10\%$ of the mean task service time, it would not occur frequently. Hence, we assume the communication overhead is 5% of the mean task service time, i.e., $\mu_{\text{in}} = 20$, unless otherwise mentioned.

In the following experiments, we first investigate the effects of system parameters on the performance of impartial discipline (F1), and then compare the performance of F1 discipline with those of partial disciplines.

4.1 F1 Discipline

4.1.1 Performance comparison. The preliminary performance measures of the load-balancing algorithm are shown in Figure 5. The results are compared with two bounds, represented by the M/M/1 model (no load balancing) and the M/M/N [11]

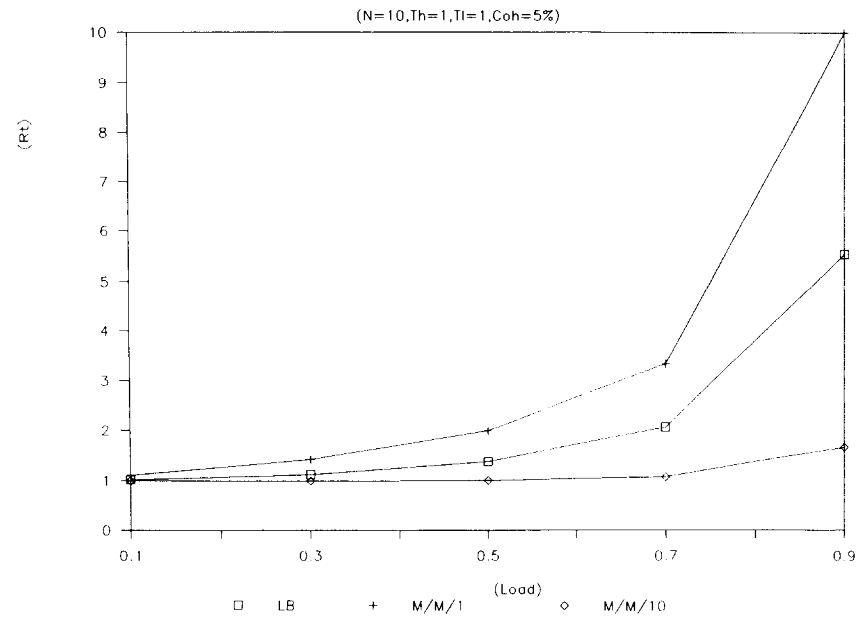

Figure 5. Load balancing performance (N = 10, Th = 1, Tl = 1, Coh = 5%).

model (perfect load balancing with no costs). From this comparison, we find that even an extremely simple load-balancing algorithm can achieve substantial performance improvement over no load balancing. The improvement is much more pronounced as the load increases. For instance, at $\rho = 0.9$, the turnaround time of the load balancing is about 5.5 units, whereas the M/M/1 turnaround time is 10 units, a significant difference. It is clear that the algorithm performs worse than the exact M/M/10 model. For instance, at $\rho = 0.9$, M/M/10 results in a turnaround time of about 1.6 units. 5

To clearly present how much performance improvement (PI) can be achieved by the load-balancing policy, we define the PI factor as follows:

$$\mathrm{PI}(\%) = (T_{\mathrm{nlb}} - T_{\mathrm{lb}})/T_{\mathrm{nlb}} \times 100\%, \qquad (7)$$

where T_{lb} (T_{nlb}) is the mean task turnaround time of the node with (without) load balancing. At extreme load condition, $\rho = 0.9$, the PI value is 44.3%. The performance of the load-balancing policy can achieve much more improvement because of the variation of system parameters, which will be described in the following subsections.

4.1.2 Effect of Th. Threshold is a fundamental parameter of the load-balancing policy. It determines when a task transfer will be attempted. Intuitively, a low Th is appropriate at low system loads because the probbility is high for a transferred task to "find" a node where the queue length is lower than the Th. However, at high system loads, a high Th is appropriate because in these conditions, most nodes have high queue length. However, high Th will reduce the task transfers and hence reduce the effects of load balancing, so the Th should be selected carefully.

Figure 6 shows the phenomenon stated above. The optimal Th is 1 for low and medium system loads ($\rho \leq 0.5$), 2 for high system loads ($\rho = 0.7$), and 3 for extreme high loads ($\rho = 0.9$). Based on this attribute, in the following experiments we use the adaptive threshold (A = Th) policy (i.e., Th = 1 for $\rho \leq 0.5$, Th = 2 for $\rho = 0.7$, and Th = 3 for $\rho = 0.9$) for simulations.

Another parameter affecting the value of optimal Th is the Coh. It is clear that low Th are appropriate for low Coh; high overheads demands higher Th.

4.1.3 Effect of Tl. With a Tl, tasks are allowed to be transferred multiple times in searching for a suitable node. Thus, the probability of a successful transfer at Tl > 1 is higher than at Tl = 1; system performance will also be improved.

Figure 7 shows mean turnaround time versus Tl for different system loads. From this figure, we observe that the higher the value of the Tl, the better the system performance will be. However, this improvement will be saturated when the value of the Tl is larger than a specific value. This phenomenon occurs because the performance improvement caused by the high probability of successful task

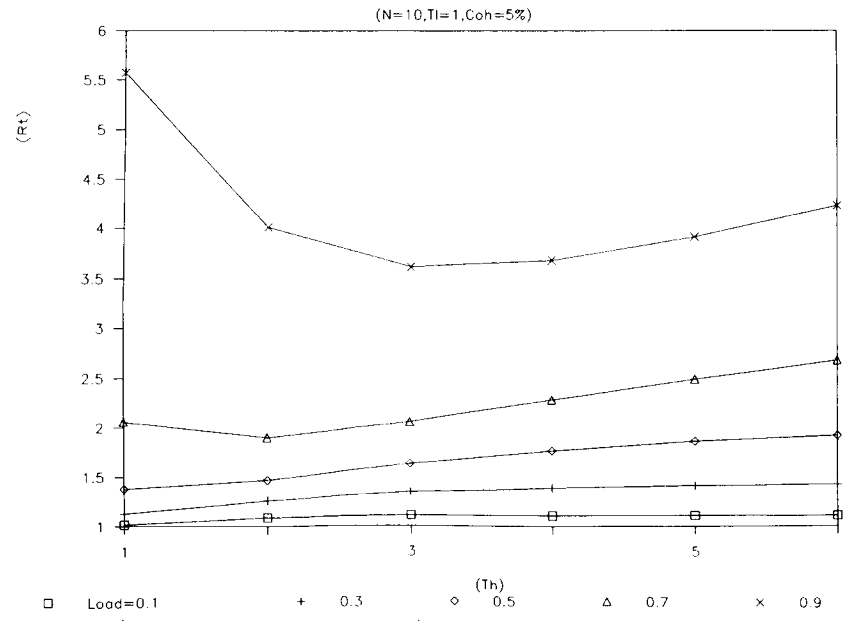

Figure 6. Effects of threshold (N = 10, Tl = 1, Coh = 5%).

transfer will be offset by the Coh under high-Tl conditions. From our simulation experiments, the specific Tl is about 3 for $\rho < 0.5$, about 4 for $0.5 \leq \rho \leq 0.7$, and about 5 for $\rho > 0.7$. As for Th, the Coh will also affect the selection of Tl. Clearly, at high Coh low Tl is appropriate; high Tl are for systems with lower overheads.

Based on these observations, we can make optimal choices for different conditions. For instance, at $\rho = 0.9$, Th = 3, and Tl = 5, the PI factor is 70.3%, which is a significant improvement over no load balancing.

4.1.4 Effect of Coh. As mentioned above, cases where the Coh of the interconnection network is > 10% of the mean task service time do not occur

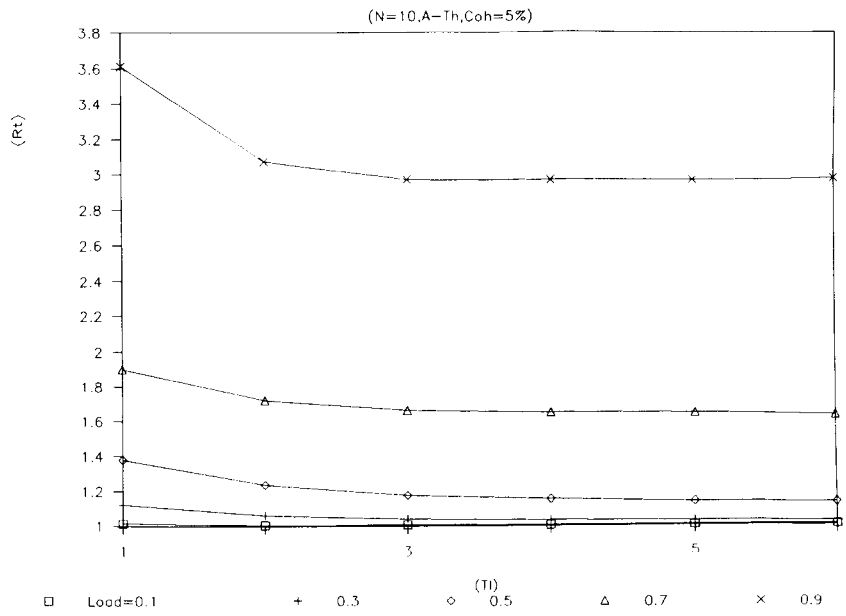

Figure 7. Effects of transfer limit (N = 10, A-Th, Coh = 5%).

frequently. However, we investigate the effect of this parameter just for theoretic analysis. Figure 8 shows the mean task turnaround time versus Coh for different system loads. In this figure, the results of $\rho = 0.5$ at Coh = 30% and $\rho = 0.9$ at Coh $\geq$ 20% are beyond the scale of the graph. For $\rho \leq 0.3$, the Coh has almost no effect on system performance. However, the system turnaround time increases dramatically at high overheads. This phenomenon proves that the queueing delay in the communication channel is a predominant factor in the mean task turnaround time at high overheads.

An interesting phenomenon occurs at Coh = 30%, where the turnaround time of $\rho = 0.7$ is lower than that of $\rho = 0.5$. This is due to the A-Th policy (i.e., Th = 1 for $\rho = 0.5$ and Th = 2 for $\rho = 0.7$) that we applied in simulations. Thus, we can imagine that these Th are not the optimal values at high overhead conditions. The description related to the effect of Coh on Th in section 4.1.2 is verified again. This phenomenon will be clearer in the following experiments. Figure 9a (some results are beyond the scale of the graph) shows the effect of Coh on the selection of optimal Th. We can observe that under Tl = 2 and Coh = 15% environments, the optimal Th at $\rho = 0.5$ is 2 instead of 1 and at $\rho = 0.9$ is 5 instead of 3. Figure 9b shows the effect of the network server on system performance. We focus experiments on extreme high load ($\rho = 0.9$) conditions. We observe that enhancing the capability of the communication channel will significantly improve system performance for high-load and low-Th environments. However, this improvement is not so clear in high-Th environments. These observations provide some tradeoffs in the design alternatives of the interconnection network.

4.1.5 Network throughput. In this section, we consider the network traffic, $N \times \gamma$, resulting from task transfers. It is clear that network throughput increases (decreases) with an increase in Tl (Th). Figure 10a shows the network throughput versus system load for different Th. We observe that the higher the Th the lower the network throughput will be. The network throughput versus Tl for different system loads is shown in Figure 10b. We observe that when the system load is < 0.5, the network traffic is small; therefore, it is easier to approach saturation. This is due to the fact that, at low system loads, there is a high probability of successful task transfer. However, when the system load is > 0.5, the network traffic increases substantially and approaches saturation at high Tl. For instance, at $\rho = 0.9$, the network traffic will approach saturation when Tl > 10 and the throughput is about 17 (not shown in Figure 10b). Another interesting phenomenon is that the network throughput at $\rho = 0.7$ is smaller than at $\rho = 0.5$, when Tl > 3. This is due to the effect of the A-Th policy (Th = 1 at $\rho = 0.5$ and Th = 2 at $\rho = 0.7$) that we applied to simulations.

Intuitively, the greater the activity of load balancing, the heavier the network traffic will be. Thus, one may imagine that the larger the network

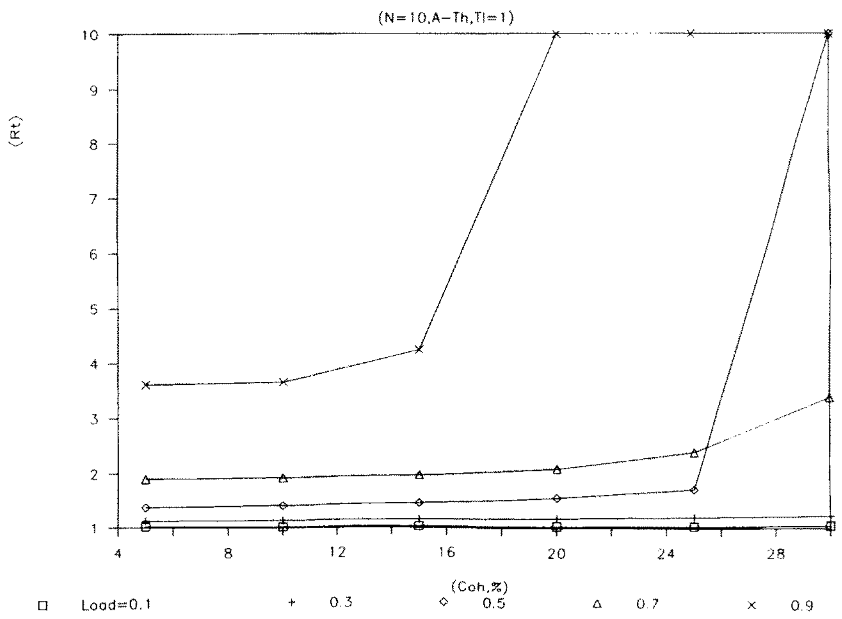

Figure 8. Effects of Coh (N = 10, A-Th, Tl = 1).

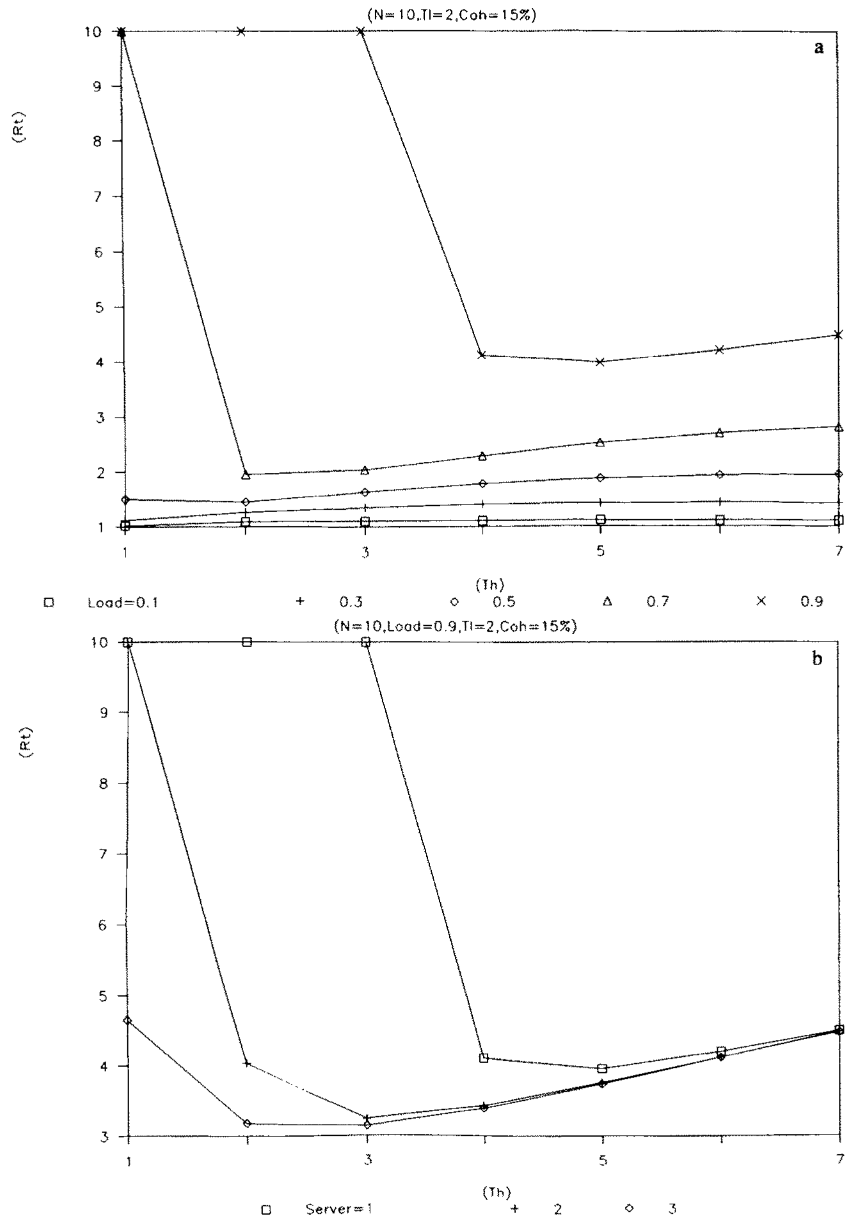

Figure 9. a, Effects of Coh on Th (N = 10, Tl = 2, Coh = 15%). b, Effects of network server (N = 10, Load = 0.9, Tl = 2, Coh = 15%).

throughput, the higher the PI will be. However, this is valid for environments with fixed Th and Coh. This phenomenon is verified by comparing Figures 10b and 7. When Th is a variant factor, the above statement is no longer true, as can be seen by comparing Figures 10a and 6.

The network throughput can be used to determine the minimum network bandwidth (BW). For example, if the size of a task is K bits, the minimum BW is $N \times \gamma \times K$. In other words, for a given network bandwidth, the network utilization is $(N \times \gamma \times K/\mathrm{BW}) \times 100\%$.

4.2 Partial Disciplines

We apply the A-Th policy stated above ($\mathrm{Th}_l = \mathrm{Th}_r = 1$ for $\rho \leq 0.5$, $\mathrm{Th}_l = \mathrm{Th}_r = 2$ for $\rho = 0.7$, and $\mathrm{Th}_l = \mathrm{Th}_r = 3$ for $\rho = 0.9$) to the following experiments.

4.2.1 Performance comparison. Figure 11a shows a comparison of the PI factor of the F1 discipline with those of five partial disciplines. In this figure, at $\rho = 0.95$, the performance of the SQF policy is worse (PI < 0) than that of the condition in which no load balancing is performed. We observe that the perfor-

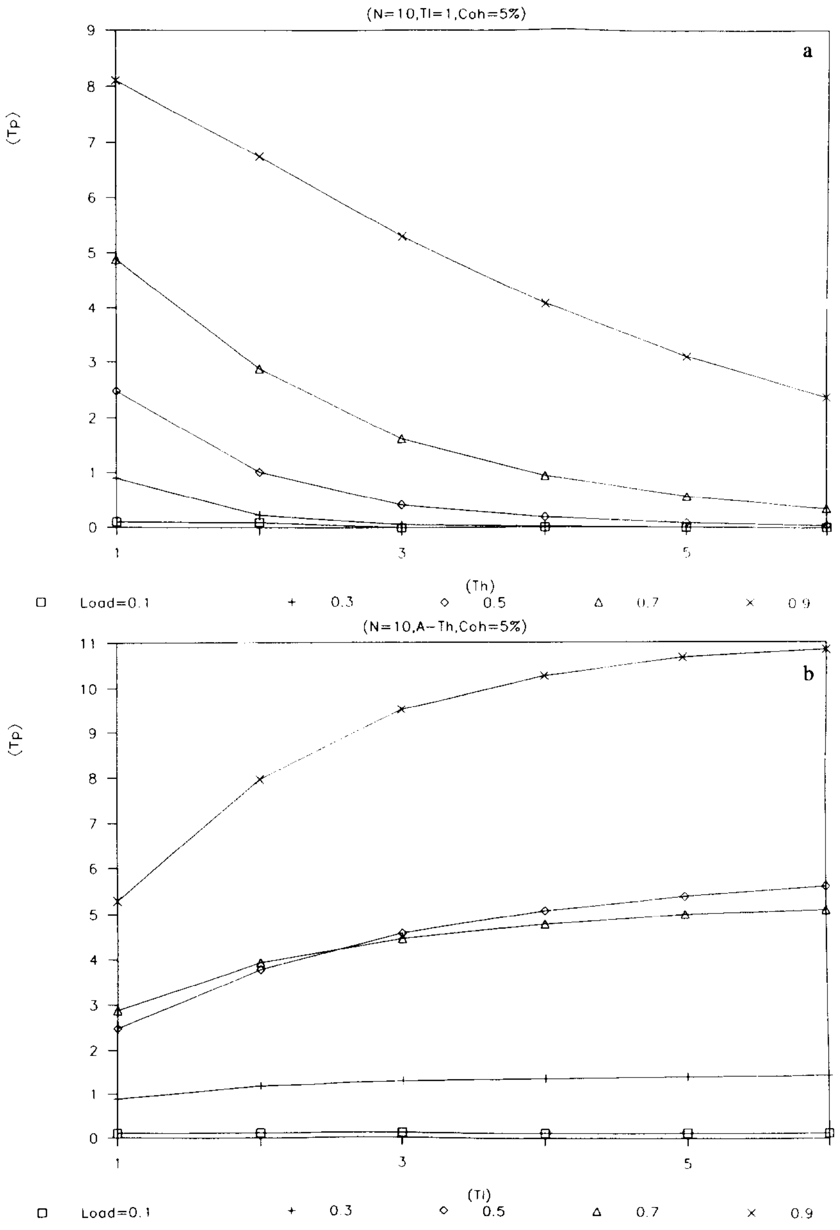

Figure 10. Network throughput. a, N = 10, Tl = 1, Coh = 5%. b, N = 10, A-Th, Coh = 5%.

mance of the F1 discipline is better than those of the partial disciplines. This is not hard to understand: under the F1 discipline, the transferred tasks from the interconnection network are treated alike, with tasks originating from the external world at that node. However, under the partial disciplines, if the local and remote tasks are assigned different priority levels for processing in the processor, then the queueing structure of the tasks in the node will be changed because of the transferred task arrivals. Intuitively, we can imagine that the disciplines favoring the remote tasks will result in shorter mean remote task turnaround time and longer mean local task response time. However, in the disciplines favoring the local tasks, shorter mean local task turnaround time and longer mean remote task turnaround time will be obtained. Let E[LT] and E[RT] be the mean local task turnaround time and the mean remote task turnaround time, respectively. When Tl = 1, the mean task turnaround time is

$$E[T] = \frac{\lambda - \gamma}{\lambda} E[LT] + \frac{\gamma}{\lambda} E[RT], \qquad (7)$$

where E[LT] = E[LQ]/($\lambda - \gamma$) and E[RT] = E[RQ]/γ + E[Coh]. For small γ, because E[T] $\approx$ E[LT] and tasks in the system are almost all of

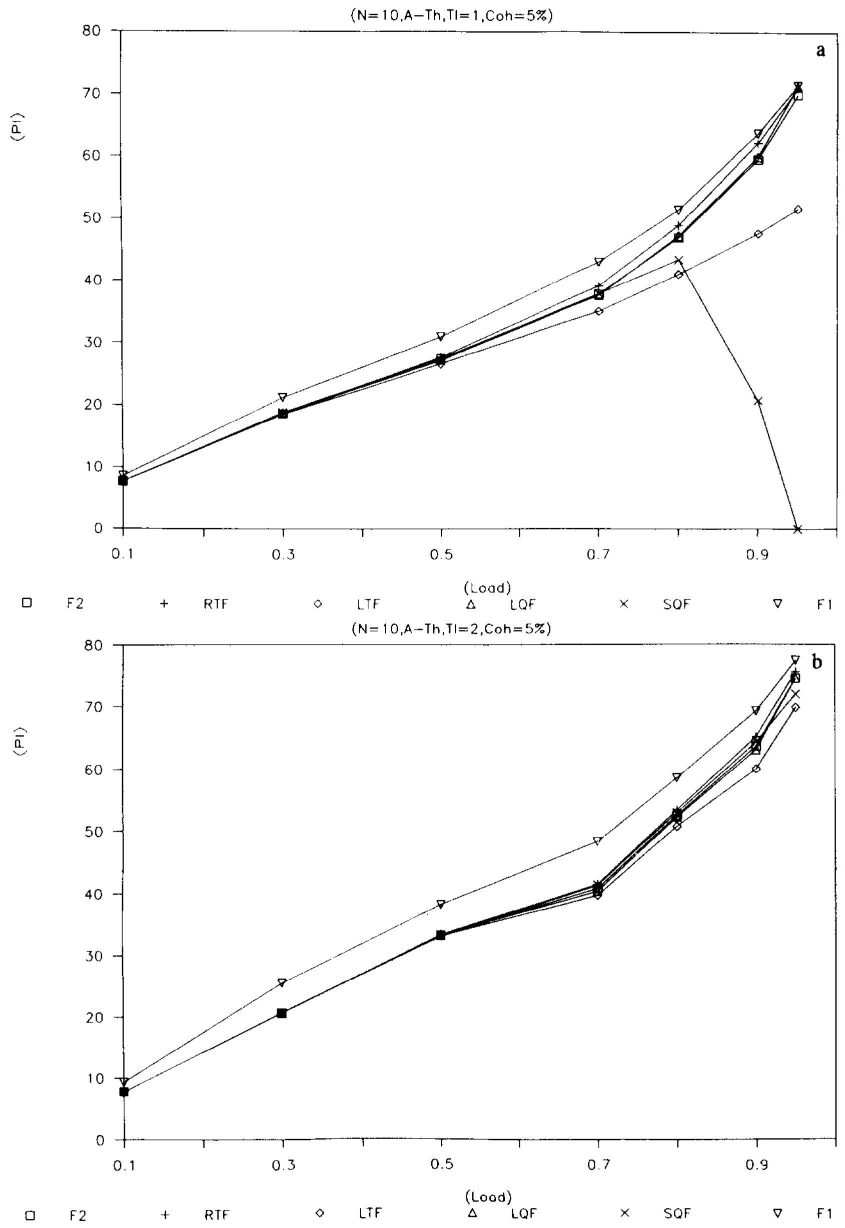

Figure 11. Performance improvement. a, $N = 10$, A-Th, Tl = 1, Coh = 5%. b, $N = 10$, A-Th, Tl = 2, Coh = 5%.

"local" type, all partial disciplines will have almost the same performance. However, if the γ is large, the E[RT] becomes a dominant factor in the E[T] and the performance of the disciplines favoring remote tasks will be better than those of the disciplines favoring local tasks.

The descriptions stated above can be verified in Figure 11a. At low and medium system loads ($\rho \leq 0.5$) and small γ, disciplines have almost the same performance. However, at high system loads ($\rho \geq 0.7$) and large γ, the RTF policy results in the best and the SQF policy results in the worst performance. This is due to the fact that, at high system loads, the SQF policy usually assigns higher priority to the local tasks. A comparison of the performance improvement of different disciplines at Tl = 2 is shown in Figure 11b. In the same way, the F1 discipline results in the best performance. However, in the partial disciplines, the RTF policy performs best, the LTF policy performs worst, and the LQF and F2 policies have almost the same performance over the entire range of the system loads.

4.2.2 Throughput comparison. Figure 12a shows the comparison of the network throughput of the F1 discipline with those of the partial disciplines. From

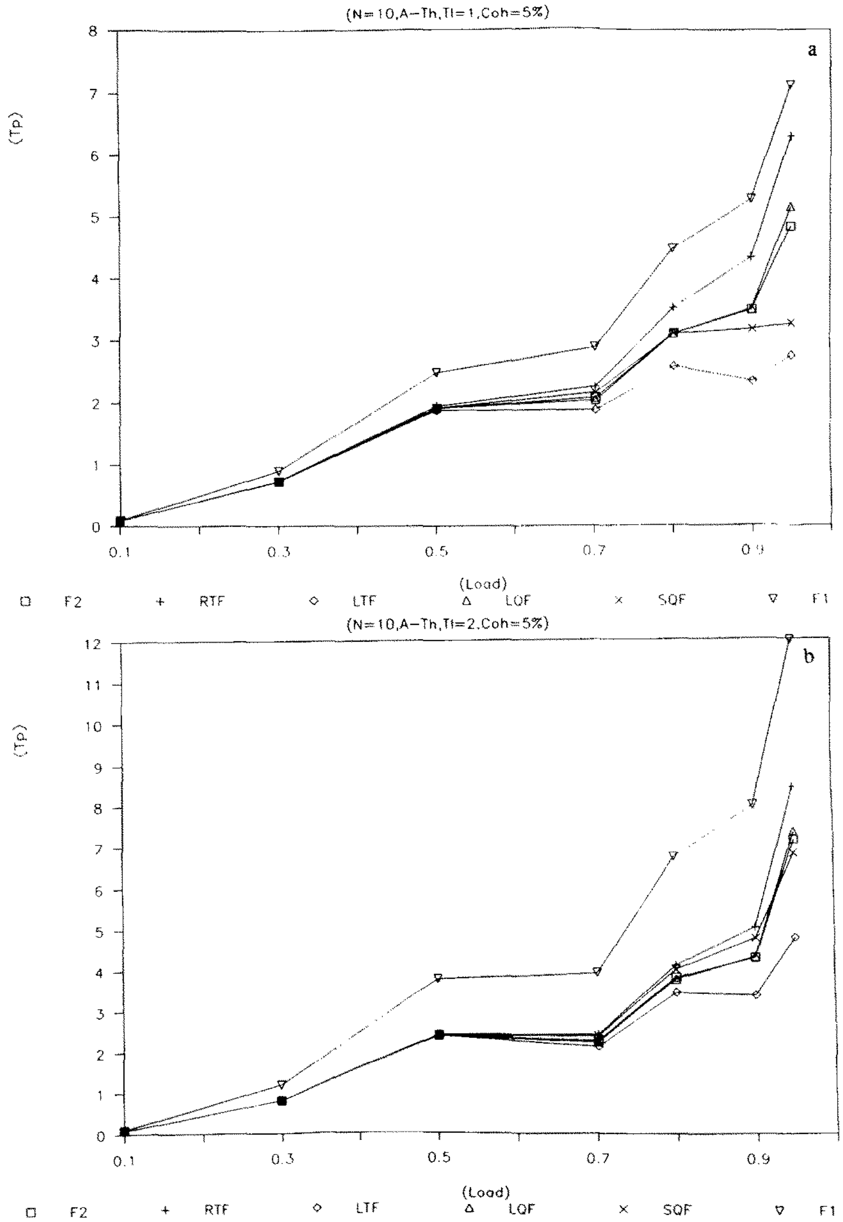

Figure 12. Throughput comparison. a, $N = 10$, A-Th, Tl = 1, Coh = 5%. b, $N = 10$, A-Th, Tl = 2, Coh = 5%.

this figure, we can observe that the network throughput of the F1 discipline is always higher than those of the partial disciplines. At low and medium system loads ($\rho \leq 0.5$), the partial disciplines have almost the same throughput. The larger the network throughput, the higher the probability of successful task transfers will be; the larger the throughput, the better the performance will be. This phenomenon can be observed in Figures 11a and 12a. Exceptions are the SQF and the LTF policies at extreme high-load conditions. The throughput comparison at Tl = 2 is shown in Figure 12b. We find that the effect of the Tl on the F1 discipline is more pronounced than on the partial disciplines.

The reason why the SQF results in the worst performance at $\rho \geq 0.9$ and Tl = 1 can be described as follows: The SQF discipline assigns higher priority to the minority tasks in the system, which results in a large queueing delay for the majority tasks and significant performance deterioration. Thus, at extreme high-load conditions and Tl = 1, the performance of SQF is worse than that of LTF. This phenomenon is more pronounced at $\rho = 0.95$, in which SQF results in a turnaround time of about 50 units and LTF results in a turnaround time of about 9.6 units. However, at large Tl, the effects of load balancing (large γ) will improve the performance of SQF significantly, this leads to the situation in which the performance of SQF is slightly superior to LTF.

4.2.3 High threshold at extreme high-system-load conditions. As mentioned above, the SQF policy almost results in the worst performance in the partial disciplines because it assigns higher priority to the minority tasks and results in a large queueing delay for the majority tasks. However, if the queue length difference between the local and remote tasks is small (under high-Th environments), then the SQF policy will perform well because of the reduced turnaround time for minority tasks and the insignificant queueing delay for the majority tasks. This description is verified in Figure 13a. We focus the experiments on extreme high-load conditions (ρ = 0.95). We observe that the SQF policy results in the best performance when Th $\geq$ 18. Because high Th are for systems with higher Coh, we can image that the SQF policy will perform well in high-overhead systems. Figure 13b shows this phenomenon. We

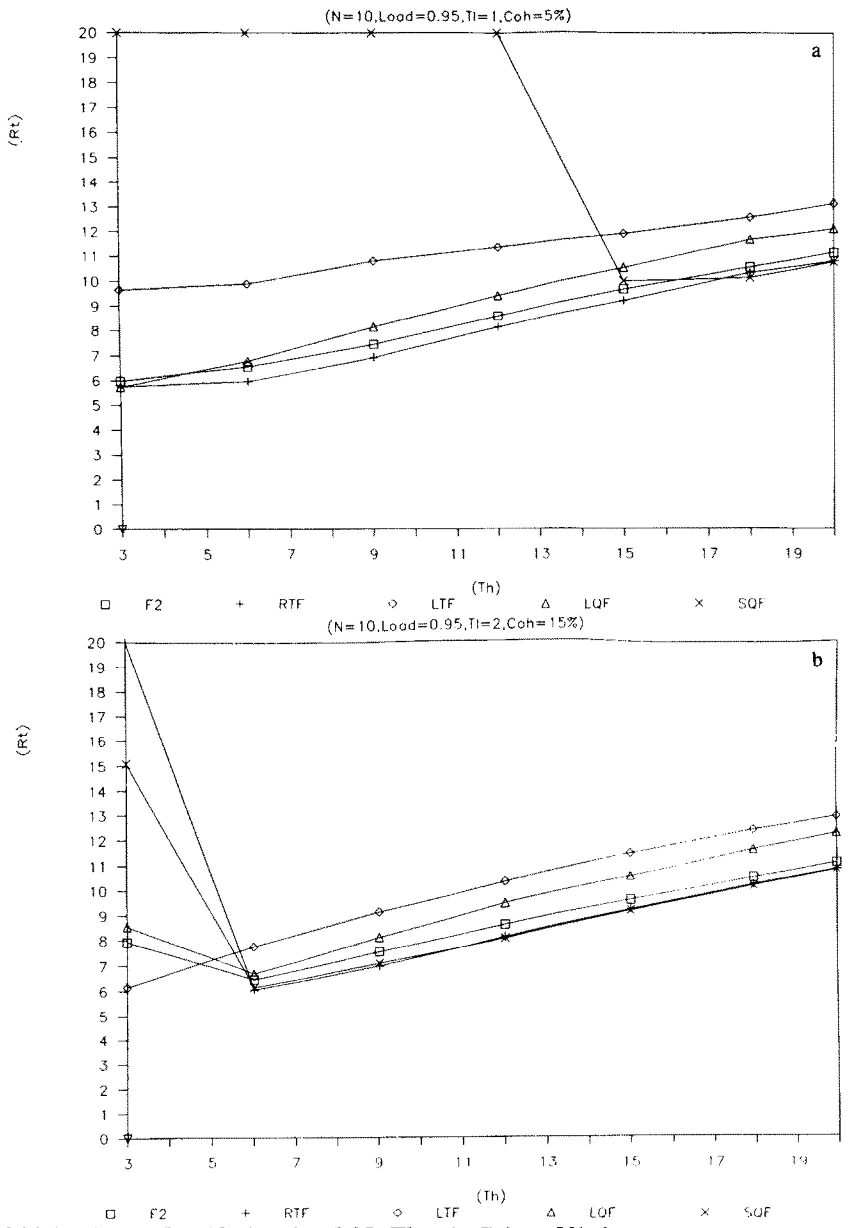

Figure 13. Effects of high Th. a, N = 10, Load = 0.95, Tl = 1, Coh = 5%. b, N = 10, Load = 0.95, Tl = 2, Coh = 15%.

also observe that under high-overhead and low-Th (Th = 3) environments, the RTF policy results in the worst performance and the LTF policy has the best performance. Based on the observation mentioned above, we can conclude that for partial disciplines at low Coh the RTF policy is appropriate; the SQF policy is suitable for systems with higher overheads.

4.3 Accuracy of the Decomposition Approximation

The mathematical queueing analysis described in section 3 is valid under the assumption that all nodes in the system are stochastically independent. The effect of the remainder of the system on an individual node is represented by an arrival process of the transferred tasks from the other nodes in the system. We also assume that this is a Poisson process. These assumptions lead to a decomposition approximation model in which each node in the system can be analyzed in isolation. Here we investigate the accuracy of this approximation. Let Sim and Num be the results from simulations and numerical solutions of the analytic model, respectively. The deviation (δ) is defined as $\delta(\%) = |\text{Sim} - \text{Num}/\text{Sim}| \times 100\%$.

Figure 14 shows the deviations of the F1, RTF, and LQF policies versus system load when $N = 10$ and $N = 20$, where F10, R10, and L10 depict the curves of F1, RTF, and LQF with $N = 10$, respectively. From this figure we observe that there is a very good consistence; the deviation is $< 2\%$ between the analytic model and the simulation when $\rho \leq 0.5$. However, the higher the system load, the larger the deviation will be. At high system loads, for instance, at $\rho = 0.9$ and $N = 10$, the deviation is $> 13\%$. These errors came from the effects of the decomposition approximation and the assumption that the transferred task process is a Poisson process. However, these effects are reduced when N is large. For instance, at $\rho = 0.9$ and $N = 20$, the deviation is $< 8\%$. Based on observations from various simulation experiences, this is because the larger the system size, the more the transferred task process (γ) is similar to a Poisson process. This leads to a situation in which the approximation is reduced for large systems. Thus, we can conjecture that the approximation error of the analytic model will approach 0 when the system size, N, approaches infinity.

The analytic queueing analysis described in section 3 is limited to Tl = 1 cases. Based on our experiments, the execution time of the simulation under extreme high-load conditions with $\rho = 0.9$ and Tl = 1 on a SUN/4 machine with 16 MIPS capability using PAWS language more than two hours. However, the same experiment, following the mathematical approach, running on an IBM-PC (80386 + 80397) with 4 MIPS capability using MATLAB language obtains a result of about 10 seconds. This is a significant difference that can be easily observed. Thus, if the analytic approach can be extended to Tl > 1 cases, a pronounced benefit will

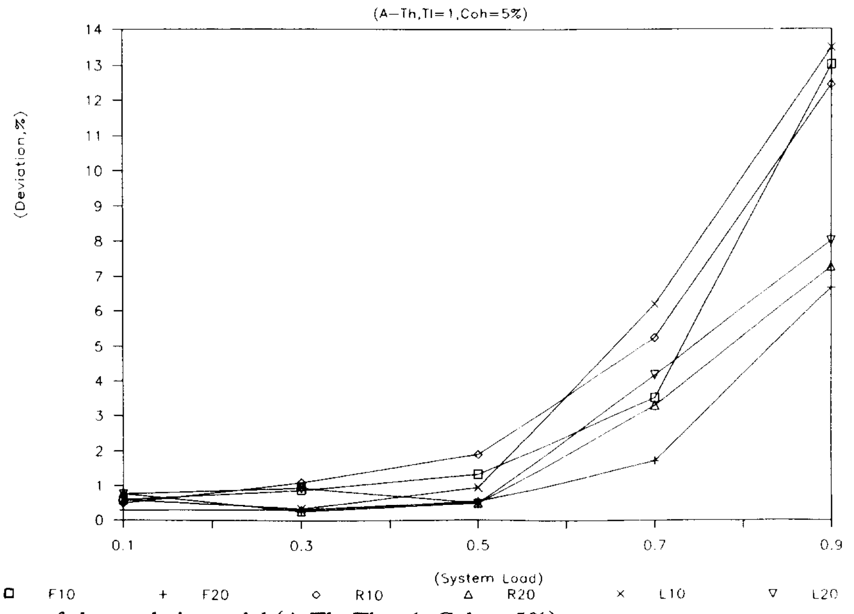

Figure 14. Development of the analytic model (A-Th, Tl = 1, Coh = 5%).

be obtained. For Tl > 1 conditions, the Markov chain approaches infinity two dimensionally, as depicted in Figure 4, some parameters, p and q, are too complicated to determine analytically. The mathematical analysis with further approximation might be a feasible approach. This interesting research area requires further investigation.

5. CONCLUSIONS

This study was concerned with the performance analysis of a simple load-balancing algorithm with different service disciplines. The six disciplines we investigated were F1, LTF, RTF, LQF, SQF, and F2. The analysis of the policies was carried out using two approaches, i.e., simulation and mathematical method.

The mathematical modeling of the Markov process of the entire system appeared to be computationally intractable. Thus, we made some assumptions and applied a decomposition technique to solve the simplified Markov process using the Z transform and matrix-geometric solution techniques.

The policies were tested over a large range of parameter values. Some salient observations are as follows.

- An extremely simple adaptive load-balancing algorithm that only collects local status information yields dramatic performance improvement over no load balancing.
- The optimal Th of the policy, under a realistic value of Coh, is 1 for $\rho \leq 0.5$, 2 for $0.5 < \rho < 0.9$, and 3 for $\rho \geq 0.9$.
- The performance of the load-balancing algorithms is insensitive to the effect of the Tl at low and medium-sized system loads. The improvement in performance caused by this effect will approach saturation when the Tl is greater than some specific values.
- The Coh is an important parameter that will affect the selection of Th and Tl. For low-overhead systems, low Th and high Tl are appropriate; high Th and low Tl are suitable for systems with higher overheads.
- Network traffic resulting from task transfers is a function of the system load, Th, and Tl.
- The F1 discipline performs better than the partial disciplines under a wide range of system loads and various Tl conditions.
- For partial disciplines, the RTF policy results in the best performance under low-overhead environments. However, the SQF policy performs very well for systems with higher overheads.
- The deviation of analytic results caused by decomposition approximation is always acceptable at low and medium-sized system loads. At high system loads, $N = 10$, the deviation might be $> 13\%$. However, it would be reduced to an acceptable value for large systems.

APPENDIX A

Here we give the closed-form representation of E[Q] of the F1 discipline. From the Markov chain in Figure 2, we can show that

$$P_i = \begin{cases} (\rho + \rho_1)^i P_0 & \text{for } i \leq \text{Th}, \\ (\rho + \rho_1)^{\text{Th}} \rho_1^{i-\text{Th}} P_0, & \text{for } i \geq \text{Th}, \end{cases} \tag{A.1}$$

where P_i denotes the probability that the queue length of the node is i. From the conservation of probabilities, $\Sigma_{i=0}^{\infty} P_i = 1$, we can solve for P_0, which give us

$$P_0 = \left[\frac{1 - (\rho + \rho_1)^{\text{Th}}}{1 - \rho - \rho_1} + \frac{(\rho + \rho_1)^{\text{Th}}}{1 - \rho_1} \right]^{-1}. \tag{A.2}$$

The mean number of tasks in the node is

$$\begin{aligned} E[Q] &= \sum_{i=1}^{\infty} iP_i \\ &= \left\{ (\rho + \rho_1) \left[\frac{1 - (\rho + \rho_1)^{\text{Th}}}{(1 - \rho - \rho_1)^2} \right. \right. \\ &\quad \left. - \frac{\text{Th}(\rho + \rho_1)^{\text{Th}}}{1 - \rho - \rho_1} \right] \\ &\quad \left. + \rho_1 (\rho + \rho_1)^{\text{Th}} \left[\frac{\text{Th}}{1 - \rho_1} + \frac{1}{(1 - \rho_1)^2} \right] \right\} P_0. \end{aligned}$$

The recursive substitution technique, which is used to compute the mean transferred task arrival/departure rate, is described as follows:

1. Set γ to an initial value.
2. Compute the probability distribution using equations A.1 and A.2.
3. Compute $\gamma_1 = \lambda \text{Prob}[Q \geq \text{Th}]$.
4. If $|\gamma_1 - \gamma| < \epsilon$, where ϵ is an arbitrary small constant, stop iteration. Otherwise, set $\gamma := \gamma_1$ and go to step 2.

The iterative scheme described above is based on the fact that Prob[Q > Th] is a nondecreasing function of γ. This is because the node queue length increases with an increasing γ for Tl = 1. Hence, the Prob[$Q >$ Th] also increases with an increasing γ. Numerical experiences indicate that regardless of the initial value of γ and how small ϵ is, the iteration always converges to a fixed point.

APPENDIX B

Here we derive the closed-form expression of E[LQ] and E[RQ] of the LTF policy using the Z transform technique. From the Markov chain in Figure 3.a, we can write down the equilibrium equations as follows:

$$\begin{aligned}
(\rho+\rho_1)P(0,0) &= P(0,1)+P(1,0),\\
(1+\rho+\rho_1)P(1,0) &= \rho P(0,0)+P(2,0),\\
(1+\rho+\rho_1)P(2,0) &= \rho P(1,0)+P(3,0),\\
(1+\rho_1)P(3,0) &= \rho P(2,0).
\end{aligned} \tag{B.1}$$

$$\begin{aligned}
(1+\rho+\rho_1)P(0,i) &= \rho_1 P(0,i-1)\\
&\quad +P(0,i+1)+P(1,i), i\geq 1,\\
(1+\rho+\rho_1)P(1,i) &= \rho_1 P(1,i-1)\\
&\quad +\rho P(0,i)+P(2,i), i\geq 1,\\
(1+\rho+\rho_1)P(2,i) &= \rho_1 P(2,i-1)\\
&\quad +\rho P(1,i)+P(3,i), i\geq 1,\\
(1+\rho_1)P(3,i) &= \rho_1 P(3,i-1)\\
&\quad +\rho P(2,i), i\geq 1.
\end{aligned} \tag{B.2}$$

Define the following Z transformations as

$$\begin{aligned}
I_0(z) &= \sum_{i=0}^{\infty} P(0,i)z^i,\\
I_1(z) &= \sum_{i=0}^{\infty} P(1,i)z^i,\\
I_2(z) &= \sum_{i=0}^{\infty} P(2,i)z^i,\\
I_3(z) &= \sum_{i=0}^{\infty} P(3,i)z^i.
\end{aligned} \tag{B.3}$$

From equations B.1, B.2, and B.3, and using some algebra, we obtain

$$\begin{aligned}
&(1+\rho+\rho_1)I_0(z)\\
&\quad = \rho_1 z I_0(z)+\frac{1}{z}I_0(z)\\
&\qquad + I_1(z)+\left(1-\frac{1}{z}\right)P(0,0),\\
&(1+\rho+\rho_1)I_1(z)\\
&\quad = \rho_1 z I_1(z)+\rho I_0(z)+I_2(z),\\
&(1+\rho+\rho_1)I_2(z)\\
&\quad = \rho_1 z I_2(z)+\rho I_1(z)+I_3(z),\\
&(1+\rho_1)I_3(z)\\
&\quad = \rho_1 z I_3(z)+\rho I_2(z).
\end{aligned} \tag{B.4}$$

From equations B.4, let $z = 1$. Using the fact that $I_0(1)+I_1(1)+I_2(1)+I_3(1) = 1$, we obtain

$$\begin{aligned}
I_1(1) &= \rho I_0(1),\\
I_2(1) &= \rho^2 I_0(1),\\
I_3(1) &= \rho^3 I_0(1),\\
I_0(1) &= \frac{1}{1+\rho+\rho^2+\rho^3}.
\end{aligned} \tag{B.5}$$

Therefore, the mean local task queue length is obtained from

$$\begin{aligned}
\mathrm{E[LQ]} &= I_1(1)+2I_2(1)+3I_3(1)\\
&= \frac{\rho+2\rho^2+3\rho^3}{1+\rho+\rho^2+\rho^3}.
\end{aligned}$$

From equation B.4, take derivative and let $z = 1$, we obtain

$$\begin{aligned}
I_3'(1) &= \rho I_2'(1)+\rho_1 I_3(1),\\
I_2'(1) &= \rho I_1'(1)+\rho_1[I_3(1)+I_2(1)],\\
I_1'(1) &= \rho I_0'(1)+\rho_1[I_3(1)+I_2(1)+I_1(1)],
\end{aligned} \tag{B.6}$$

where the prime (′) denotes the derivative with respect to its argument. Since $I_0'(1) = \sum_{i=1}^{\infty} iP(0,i)$, applying the fact that $P(0,i+1) = \rho_1 \sum_{j=0}^{3} P(j,i)$, for $i \geq 0$, and substituting them into the summation, we have

$$I_0'(1) = \rho_1(1+\mathrm{E[RQ]}). \tag{B.7}$$

From equations B.5, B.6, and B.7, after some algebraic manipulations, we finally obtain the mean remote task queue length:

$$\begin{aligned}
\mathrm{E[RQ]} &= I_0'(1)+I_1'(1)+I_2'(1)+I_3'(1)\\
&= \frac{\rho_1(\rho^6+3\rho^5+6\rho^4+10\rho^3+6\rho^2+3\rho+1)}{(1+\rho+\rho^2+\rho^3)[1-\rho_1(1+\rho+\rho^2+\rho^3)]}.
\end{aligned}$$

The mean remote task arrival rate is

$$\gamma = \lambda \mathrm{Prob}[\mathrm{LQ} = \mathrm{Th}_1]$$

$$= \lambda I_3(1) = \frac{\lambda \rho^3}{1+\rho+\rho^2+\rho^3}.$$

APPENDIX C

Here we describe the analytic queueing analysis of the LQF policy (Tl = 1 and $\mathrm{Th}_1 = 3$) in detail. Analysis of the RTF and the SQF policies can be performed by the same procedures as the LQF policy and is omitted.

Let $X_i = [P(0,i), P(1,i), P(2,i), P(3,i)]$ be the probability vector that the node has i remote tasks and $\overline{X} = [X_0, X_1, X_2, \ldots]$. From the Markov chain in Figure 3c, we can write down the equilibrium equations that lead to the matrix equation $\overline{X} * G = 0$, which describes the behavior of the system at equilibrium, where G is the infinitesimal generator of the Markov chain and $*$ is the operation of matrix multiplication. G has the structure of a block tridiagonal matrix of the form

$$G = \begin{bmatrix} B_{00} & B_{01} & & & & & \\ B_{10} & B_{11} & B_{12} & & & & \\ & B_{21} & B_{22} & B_{23} & & & \\ & & B_{32} & B_{33} & A_0 & & \\ & & & B_{43} & A_1 & A_2 & \\ & & & \cdot & A_2 & A_1 & \cdot \\ & & & & \cdot & A_2 & \cdot \end{bmatrix}.$$

We define the components of internal matrices of the infinitesimal generator G at the end of this appendix. Consider the following nonlinear matrix equation

$$A_0 + R * A_1 + R^2 * A_2 = 0 \qquad \text{(C.1)}$$

such that R is its nonnegative solution. We can show that R is an upper triangular matrix. Let $R = [r_{ij}]$, where

$$r_{ij} = 0, \forall\, i > j,$$

$$r_{11} = r_{22} = r_{33}$$

$$= \frac{\left[(\lambda+\gamma+\mu) - (\lambda+\gamma+\mu)^2 - 4\gamma\mu\right]^{0.5}}{2\mu},$$

$$r_{44} = \frac{\gamma}{\mu},$$

$$r_{12} = r_{23} = \frac{\lambda r_{11}}{(\lambda+\gamma+\mu) - \mu(r_{11}+r_{22})},$$

$$r_{34} = \frac{\lambda r_{33}}{(\gamma+\mu) - \mu(r_{33}+r_{44})},$$

$$r_{13} = \frac{r_{12}(\lambda+\mu r_{23})}{(\lambda+\gamma+\mu) - \mu(r_{11}+r_{33})},$$

$$r_{24} = \frac{r_{23}(\lambda+\mu r_{34})}{(\gamma+\mu) - \mu(r_{22}+r_{44})}, \text{ and}$$

$$r_{14} = \frac{\lambda r_{13} + \mu(r_{12}r_{24} + r_{13}r_{34})}{(\gamma+\mu) - \mu(r_{11}+r_{44})}.$$

Thus, the diagonal elements of R can be described explicitly in terms of the parameters of the Markov process. Once the diagonal elements are determined, the elements above the diagonal are computed from the value of the diagonal elements. The following theorem has been proven by Neuts [12].

Theorem. The Markov chain, with infinitesimal generator G, is positive recurrent if and only if $SP(R) < 1$, all the eigenvalues of R lie inside the unit disk, and the stochastic matrix $B[R]$, defined below, has a positive left invariant vector, eigenvector, $[X_0, X_1, X_2, X_3]$. Normalizing the eigenvector $[X_0, X_1, X_2, X_3]$ by $(X_0 + X_1 + X_2) * e + X_3 * (I - R)^{-1} * e = 1$, where $e = [1,1,1,1]^T$, T denotes the transpose of a vector, and I is the identity matrix of size 4, then the invariant probability vector X of G is given by $X_i = X_3 * R^{i-3}$, for $i \geq 3$.

In the case of the LQF policy, because R is a upper triangular matrix, its eigenvalues are its diagonal elements. It is clear that $SP(R) < 1$ if $\gamma < \mu$. However, for Tl = 1 cases, the value of γ is always smaller than that of the λ. The matrix $B[R]$, given by

$$B[R] = \begin{bmatrix} B_{00} & B_{01} & 0 & 0 \\ B_{10} & B_{11} & B_{12} & 0 \\ 0 & B_{21} & B_{22} & B_{23} \\ 0 & 0 & B_{32} & B_{33} + R * B_{43} \end{bmatrix}$$

is an irreducible, aperiodic matrix. The vector $[X_0, X_1, X_2, X_3]$ is the left eigenvector of $B[R]$. Therefore, the two conditions stated in the above theorem are satisfied. We now assume that all the values of all parameters are known. First, we calculate the components of the R matrix. The boundary conditions are determined by solving a system of linear equations $[X_0, X_1, X_2, X_3] * B[R] = 0$, and the remainder probability vectors are obtained from

$X_i = X_3 * R^{i-3}$, for $i \geq 3$. Thus, the performance measures are

$$E[\mathrm{LQ}] = \sum_{i=0}^{\infty} X_i * e_1 = (X_0 + X_1 + X_2) * e_1 + X_3 * (I - R)^{-1} * e_1, \quad \text{(C.2)}$$

$$E[\mathrm{RQ}] = \sum_{i=0}^{\infty} iX_i * e = (X_1 + 2X_2) * e + 3X_3 * (I - R)^{-1} * e + X_4 * (I - R)^{-2} * e, \quad \text{(C.3)}$$

where $e_1 = [0, 1, 2, 3]^T$.

The components of internal matrices of the infinitesimal generator G are listed as follows:

$$B_{00} = \begin{bmatrix} -\lambda-\gamma & \lambda & 0 & 0 \\ \mu & -\lambda-\gamma-\mu & \lambda & 0 \\ 0 & \mu & -\lambda-\gamma-\mu & \lambda \\ 0 & 0 & \mu & -\gamma-\mu \end{bmatrix},$$

$$B_{10} = \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{11} = \begin{bmatrix} -\lambda-\gamma-\mu & \lambda & 0 & 0 \\ \mu & -\lambda-\gamma-\mu & \lambda & 0 \\ 0 & \mu & -\lambda-\gamma-\mu & \lambda \\ 0 & 0 & \mu & -\gamma-\mu \end{bmatrix},$$

$$B_{21} = \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{22} = \begin{bmatrix} -\lambda-\gamma-\mu & \lambda & 0 & 0 \\ 0 & -\lambda-\gamma-\mu & \lambda & 0 \\ 0 & \mu & -\lambda-\gamma-\mu & \lambda \\ 0 & 0 & \mu & -\gamma-\mu \end{bmatrix},$$

$$B_{32} = \begin{bmatrix} \mu & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B_{33} = \begin{bmatrix} -\lambda-\gamma-\mu & \lambda & 0 & 0 \\ 0 & -\lambda-\gamma-\mu & \lambda & 0 \\ 0 & 0 & -\lambda-\gamma-\mu & \lambda \\ 0 & 0 & \mu & -\gamma-\mu \end{bmatrix},$$

$$A_1 = \begin{bmatrix} -\lambda-\gamma-\mu & \lambda & 0 & 0 \\ 0 & -\lambda-\gamma-\mu & \lambda & 0 \\ 0 & 0 & -\lambda-\gamma-\mu & \lambda \\ 0 & 0 & 0 & -\gamma-\mu \end{bmatrix}$$

$B_{01} = B_{12} = B_{23} = A_0 = \gamma * I$, $B_{43} = A_2 = \mu * I$.

REFERENCES

1. M. Livney and M. Melman, Load Balancing in Homogeneous Broadcast Distributed Systems, *Perform. Eval. Rev.* 11, 47–55 (1982).
2. H. S. Stone, Multiprocessor Scheduling with the Aid of Network Flow Algorithms, *IEEE Trans. Software Eng.* SE-3, 85–93 (1977).
3. H. S. Stone, Critical Load Factors in Two Processor Distributed Systems, *IEEE Trans. Software Eng.* SE-4, 254–258 (1978).
4. S. H. Bokhari, Dual Processor Scheduling with Dynamic Reassignment, *IEEE Trans. Software Eng.* SE-5, 341-349 (1979).
5. A. N. Tantawi and D. Towsley, Optimal Static Load Balancing in Distributed Computer Systems, *J. ACM* 32, 445–465 (1985).
6. D. L. Eager E. D. Lazowska, and J. Zahorjan, A Comparison of Receiver-Initiated and Sender-Initiated Adaptive Load Sharing, *Perform. Eval.* 6, 53–68 (1986).
7. Y. Wang and R. Morris, Load Sharing in Distributed Systems, *IEEE Trans. Comput.* C-34, 204–217 (1985).
8. M. N. Lionel, C. W. Xu, and T. B. Gendreau, A Distributed Drafting Algorithm for Load Balancing, *IEEE Trans. Software Eng.* SE-11, 1153–1161 (1985).
9. L. Kleinrock, *Queueing Systems, Vol. 2: Computer Applications*, Wiley, New York, 1976.
10. J. D. C. Little, A Proof of Queueing Formula L = λW, *Oper. Res.* 9, 383–387 (1961).
11. L. Kleinrock, *Queueing Systems, Vol. 1: Theory*, Wiley, New York, 1975.
12. M. F. Neuts, *Matrix Geometric Solutions in Stochastic Models: An Algorithmic Approach*, Johns Hopkins University Press, 1981.
13. P. Heidelberger and S. S. Lavenberg, Computer Performance Evaluation Methodology, *IEEE Trans. Comput.* C-33, 1195–1220 (1984).
14. *PAWS 3.0 User's Manual*, Scientific and Engineering Software, Inc., Austin, Texas, 1987.
15. N. S. Matloff, Probability Modeling and Computer Simulation, PWS-KENT, Boston, 1988.