Certain Performance Aspects of Optimal Load Balancing in Distributed Computer Systems

Doctoral Program in Engineering

University of Tsukuba

March 2004

Said Fathy El-Zoghdy

Certain Performance Aspects of Optimal Load Balancing in Distributed Computer Systems

March 2004

Said Fathy El-Zoghdy

A dissertation submitted in partial fulfillment of requirements
for the degree of Doctor of Philosophy in Engineering

Electronics and Information Sciences

Doctoral Program in Engineering

University of Tsukuba

To my family

Contents

List of Tables

List of Figures

Acknowledgements

First of all, I owe heartfelt thanks to my advisor, Professor Hisao Kameda, who provided kind and heartful support to my study, guided my research and led me toward the completion of this thesis. Without his strict, but kind and generous instruction, this work could not have been completed.

I would like to express special gratitude to Associate Professor Jie Li who had many useful conversations with me, provided many guidance and encouragement.

I also express sincere thanks to Professor Yoshihiko Ebihara, Professor Hiroyuki Kitagawa and Professor Koichi Wada of the Institute of Information Sciences and Electronics at the University of Tsukuba, for their valuable comments and discussions.

Many thanks go to all my colleagues and researchers in the Operating System and Distributed/Parallel Processing Laboratory at the University of Tsukuba for their helpful inspiration and hospitality.

I offer my regret to my children, Yasmin and Mohammed, for taken much time for this work which might have been spent with them and my thanks for their indulgence.

Finally, I dedicate this thesis to my wife, Manal, to my father and to my mother. I deeply appreciate their unending patience, encouragement and understanding, without which I could hardly have completed this work.

Abstract

A distributed computer system is considered to be a collection of autonomous computers (nodes) located at possibly different sites and connected by a communication network. Through the communication network, resources of the system can be shared by users at different locations. Performance enhancement is one of the most important issues in distributed systems. The performance of a distributed computer system can often be improved to an acceptable level by redistributing the workload among nodes. The problem of load redistribution in distributed computer systems is called *load balancing*. Load balancing policies may be either *static* or *dynamic*.

Static load balancing policies use only the statistical information on the system (e.g., the average behavior of the system) in making load balancing decisions. On the other hand, dynamic load balancing policies attempt to dynamically balance the workload reflecting the current system state and are therefore thought to be able to further improve the system performance.

Generally, the purpose of load balancing policies either static or dynamic is to improve the performance of the system by redistributing the workload among nodes. We can choose between several distinct objectives for performance optimization in many systems including communication networks, distributed computer systems, transportation flow networks, etc. Among them, we have the following three typical objectives or optima:

1. *The overall optimum*, where all jobs are regarded to belong to one group that has only one decision maker. The decision maker seeks to optimize a certain overall and single performance measure like the total cost or the overall mean response time over all the jobs.

2. *The individual optimum*, where each of infinitely many jobs (or the user of each) optimizes its own cost (e.g., its own expected response time) independently of the others.

3. *The class optimum*, where infinitely many jobs are classified into a finite number of classes or groups, each of which has its own decision maker and is regarded as one player or user. Each decision maker optimizes non-cooperatively its own cost (e.g., the expected response time) over only the jobs of its own class.

In this thesis, we use these three performance aspects (objectives or optima) with both static and dynamic load balancing policies to optimize the performance of the following two distributed computer systems. The first system consists of two types of service facilities, a Mainframe node Q_{MF} and an unlimited number of Personal Computer nodes Q_{PC}, both of which are connected by a communication network. We call this system model an *MF-PC network model*. The second system consists of a set of heterogeneous nodes (host computers or processors) connected in an arbitrary fashion by a communication network.

First, on the MF-PC network model, a comparison between the performance of a static overall optimal load balancing policy (SOOLBP) and a dynamic overall optimal load balancing policy (DOOLBP) is performed. We considered the $[L, q]$ threshold rule as a DOOLBP. Truly optimal solutions of both SOOLBP and DOOLBP have been characterized. The overheads due to the two policies are assumed to be negligible. For the DOOLBP (i.e., [L,q] threshold rule), a numerical algorithm for obtaining the optimal values of the threshold parameters L and q is proposed. Analytically, it is proved that the minimum value of the overall system mean response time is obtained by the DOOLBP ([L,q] threshold rule) with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. Also, we analytically proved the existence and uniqueness of optimal solution of the other threshold parameter L. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the overall system mean response time using the SOOLBP and the DOOLBP are studied through numerical experimentation. The results show that, in the model examined, the overall mean response time is improved by the DOOLBP over that of the one at most about 30% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The maximum improvement ratio is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase.

Second, on the MF-PC network model, a comparison between the performance of a static individually optimal load balancing policy (SIOLBP) and a dynamic individually optimal load balancing policy (DIOLBP) is performed. The $[L, q]$ threshold rule is considered as a DIOLBP. Truly optimal solutions of both SIOLBP and DIOLBP have been characterized. The overheads due to the two policies are assumed to be negligible. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the mean job response time using the SIOLBP and the DIOLBP are studied through numerical experimentation. The results show that the DIOLBP outperforms the SIOLBP in the overall mean response time, at most about 48% in the range of parameter values examined. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model under the SIOLBP and the DIOLBP. We found that, there is a difference between the ratio that a job arriving at the system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase. Through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have effect in satisfying the equilibrium in between the two system facilities. And also, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ and it approaches μ/θ. Additionally, several interesting phenomena are also observed.

Third, in a distributed computer system that consists of a set of heterogeneous nodes connected with a communication means, we presented a number of numerical examples around the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for static load balancing. Three different types of communication means (A), (B) and (C) are considered. Based on the system parameter setting, three types of symmetries *(overall symmetry, individual symmetry and complete symmetry)* are defined. From the numerical examples, it is observed that in class optimum, the worst-case degree of the paradox (WCDP) is largest (i.e., the worst performance is obtained) in the complete symmetry case where the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

Chapter 1

Introduction

1.1 Overview

One of the main advantages of distributed computer systems over stand-alone systems is the potential for resource sharing, to provide the users with a rich collection of resources that are usually unavailable or highly contended for in stand-alone systems. It is frequently observed that, in a computing environment with a number of nodes (host computers) connected by communications network, the nodes are often loaded very differently. Such imbalances in system load suggest that performance can be improved by transferring jobs from the heavily loaded nodes to the lightly loaded ones. This form of computing power sharing, with the purpose of improving the performance of a distributed computer system by redistributing the workload among the available nodes, is commonly called *load balancing*. Load balancing may be either *static* or *dynamic*.

Static load balancing policies [29, 31, 32, 33, 44, 38, 39, 40, 41, 48, 50] use only the statistical information on the system (e.g., the average behavior of the system) in making load-balancing decisions, and their principal advantage is lower overhead cost needed to execute them and their simplicity in implementation and mathematical tractability. They do not, however, adapt to fluctuations in workload. Under a situation where the system workload is statistically balanced, some computers may be heavily loaded at a given instant (hence suffering from performance degradation), while others are idle or lightly loaded.

On the other hand, dynamic load balancing policies [8, 31, 32, 40, 41, 65, 68, 85, 97, 98] attempt to dynamically balance the workload reflecting the current system state and are therefore thought to be able to further improve the system performance. Thus, it would be thought that, compared to static ones, dynamic load balancing policies are better able to respond to system changes and to avoid those states that result in poor performance. However, this is not always the case. In [97, 98] it have been shown through simulation that when overheads are non-negligibly high at heavy system loads, static load balancing policies can provide performance more stable and better than that provided by some dynamic load balancing policies. Obviously, the disadvantages of dynamic load balancing policies is that these policies are more complex than their static counterparts, in the sense that they require information on the runtime load and activities of state collection.

The purpose of load balancing policies either static or dynamic is to improve the performance of the system by redistributing the workload among nodes. We can choose between several distinct objectives for performance optimization in many systems including communication networks, distributed computer systems, transportation flow networks, etc. Among them, we have the following three typical objectives or optima:

1. *The overall optimum*, where all jobs are regarded to belong to one group that has only one decision maker. The decision maker seeks to optimize a certain overall and single performance measure like the total cost or the overall mean response time over all the jobs. We call an optimal load balancing policy whereby the overall mean response time is minimized the *overall optimal policy*. By the *overall optimization*

problem we mean the problem of obtaining the load balancing decision that achieves the objective of the overall optimal policy. In the literature, the solution of the overall optimization problem is referred to as system optimum, overall optimum, cooperative optimum or social optimum. In this thesis, we shall refer to it as the *overall optimum*.

2. *The individual optimum*, where each of infinitely many jobs (or the user of each) optimizes its own cost (e.g., its own expected response time) independently of the others. In this optimized situation, each job cannot expect any further benefit by changing its own decision. It is also assumed that the decision of a single job has a negligible impact on the performance of other jobs. We call an optimal load balancing policy whereby every job strives to optimize (minimize) its own mean response time independently of the other jobs the *individually optimal policy*. By the *individual optimization* problem we mean the problem of obtaining the load balancing decision that achieves the objective of the individually optimal policy. In the literature, the solution of the individual optimization problem is referred to as an individual optimum, Wardrop equilibrium, or user optimum. In this thesis, we shall refer to it as the *individual optimum*.

3. *The class optimum*, where infinitely many jobs are classified into a finite number ($N > 1$) of classes or groups, each of which has its own decision maker and is regarded as one player or user. Each decision maker optimizes non-cooperatively its own cost (e.g., the expected response time) over only the jobs of its own class. The decision of a single decision maker of a class has a non-negligible impact on the performance of other classes. In this optimized situation, each of a finite number of classes or players cannot receive any further benefit by changing its decision. We call the load balancing policy that has the previous description the *class optimal*

policy. By the *class optimization* problem we mean the problem of obtaining the load balancing decision that achieves the objective of the class optimal policy. In the literature, the solution of the class optimization problem is referred to as the class optimum, or Nash equilibrium. In this thesis, we shall refer to it as the *class optimum*.

In this thesis, we use these three performance aspects (objectives or optima) with both static and dynamic load balancing policies to optimize the performance of the following two distributed computer systems. The first system consists of two types of service facilities, a Mainframe node Q_{MF} and an unlimited number of Personal Computer nodes Q_{PC}, both of which are connected by a communication network. We call this system model an *MF-PC network model*. The second system consists of a set of heterogeneous nodes (host computers or processors) connected in an arbitrary fashion by a communication network.

First, on the MF-PC network model, a comparison between the performance of a static overall optimal load balancing policy (SOOLBP) and a dynamic overall optimal load balancing policy (DOOLBP) is performed [32, 39, 40]. The $[L, q]$ threshold rule is considered as a DOOLBP. Truly optimal solutions of both SOOLBP and DOOLBP have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, we do not take account of the difference in the overheads due to the two policies. For the DOOLBP ([L,q] threshold rule), a numerical algorithm for obtaining the optimal values of threshold parameters L and q is proposed. Analytically, it is proved that the minimum value of the overall system mean response time is obtained by the DOOLBP ([L,q] threshold rule) with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. Also, we analytically proved the existence and uniqueness of optimal solution for the other threshold parameter L. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1 and thus we have only two independent parameters λ and μ. The effects of changing these two parameters (λ and μ) on the overall system mean response time using the SOOLBP and DOOLBP are studied through numerical experimentation. The results show that, in the model examined, the overall system mean response time is improved by the DOOLBP over that of the SOOLBP at most about 30% in the range of parameter values examined. And, the maximum improvement ratio is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase.

Second, on the MF-PC network model, a comparison between the performance of a static individually optimal load balancing policy (SIOLBP) and a dynamic individually optimal load balancing policy (DIOLBP) is performed [31]. The $[L, q]$ threshold rule is considered as a DIOLBP. Truly optimal solutions of both SIOLBP and DIOLBP have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, we do not take account of the difference in the overheads due to the two policies. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1 and thus we have only two independent parameters λ and μ. The effects of changing these two parameters (λ and μ) on the mean job response time using the SIOLBP and DIOLBP are studied through numerical experimentation. The results show that the DIOLBP outperforms the SIOLBP in the overall mean response time, at most about 48% in the range of parameter values examined. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model under the SIOLBP and the DIOLBP. We found that, there is a difference between the ratio that a job arriving at the system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase. Through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have effect in satisfying the equilibrium in between the two system facilities. And also, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ and it approaches μ/θ. Additionally, several interesting phenomena are also observed.

Third, on a distributed computer system that consists of a set of heterogeneous nodes connected with a communication means, we presented a number of numerical examples around the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for static load balancing [29, 30, 33]. Three different types of communication means (A), (B) and (C) are considered. Based on the system parameter setting, three types of symmetries *(overall symmetry, individual symmetry and complete symmetry)* are defined. From the numerical examples, it is observed that in class optimum, the worst-case degree of the paradox (WCDP) is largest (i.e., the worst performance is obtained) in the complete symmetry case where the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

1.2 Methodology

The research methodology applied throughout this thesis is mathematical modelling. The programs for the considered models are implemented using the Microsoft Visual C++ version 6 on windows platform.

1.3 Thesis Outline

This thesis is organized as follows.

Chapter 2 presents a survey of the previous and the current studies on static and dynamic load balancing and Braess paradox in distributed computer systems.

Chapter 3 presents a comparison between the performance of a static overall optimal load balancing policy and a dynamic overall optimal load balancing policy on the MF-PC network model.

Chapter 4 presents a comparison between the performance of a static individually optimal load balancing policy and a dynamic individually optimal load balancing policy on the MF-PC network model.

Chapter 5 presents some numerical examples around the Braess-like paradoxes for non-cooperative static load balancing in a heterogeneous distributed computer system.

Chapter 6 concludes this thesis and describes the author's plans for future work.

Appendix A derives the overall system mean response time of a job arriving at the MF-PC network model with the $[L, q]$ threshold rule, $E\left[W_{[L,q]}\right]$.

Chapter 2

Background

A distributed computer system is considered to be a collection of autonomous nodes (host computers) located at possibly different sites and connected by a communication network. Through the communication network, resources of the system can be shared by users at different locations. However, a fundamental problem arises in making effective use of the total computing power of a distributed computing system. It is often the case that a certain node has very few tasks to handle at a given time, while another node has many. It is desirable to spread the total workload of the distributed computer system over all of its nodes. This avoids under utilization of power; further, it decreases response time for work introduced at more heavily loaded nodes. This form of computing power sharing, with the purpose of improving the performance of a distributed system by redistributing the workload among the available nodes, is commonly called *load balancing*. The purpose of load balancing is to improve the performance of the system by redistributing the workload among nodes, thus increasing processing capacity of the system without having to obtain additional or faster computer hardware.

Another method for improving the performance of a distributed computer system is upgrading the system by adding additional or faster computer hardware aiming to increase the total processing capacity of the system. In other words, we can think that the total processing capacity of a system will increase when the capacity of a part of the system increases and so we expect improvements in performance objectives accordingly in that case. The famous *Braess Paradox* tells us that this is not always the case; i.e., adding capacity to the system may sometimes lead to the degradation in the benefits of all users in an individual optimum.

This chapter presents a survey of the previous and the current studies on load balancing and Braess Paradox in distributed computer systems.

2.1 Load Balancing: A survey

Recent years have been witness to an increasing use of distributed computing system. This may be attributed to two main factors: growth of the Internet, and low cost solution of end-user computing devices. Many processes are distributed due to the inherent nature of tasks involved with them. Besides, scale of economy is often possible due to the use of clusters of less powerful computers instead of a central computer of significantly high power. However, a distributed solution can yield the true advantage only if it is possible to distribute works evenly among the available computers (nodes of the system). In other words, when load on the computers in a distributed environment has significant variance of workloads, high performance can be achieved by redistributing loads. The task of redistributing the loads on the computers is called *load balancing*.

Load balancing can be considered for two different types of systems: the multiprocessors, and the distributed computer systems. It is difficult to define these terms precisely because they have been used very imprecisely in the literature. We define these two terms by describing the most important characteristics of each. A *multiprocessor* is any computer

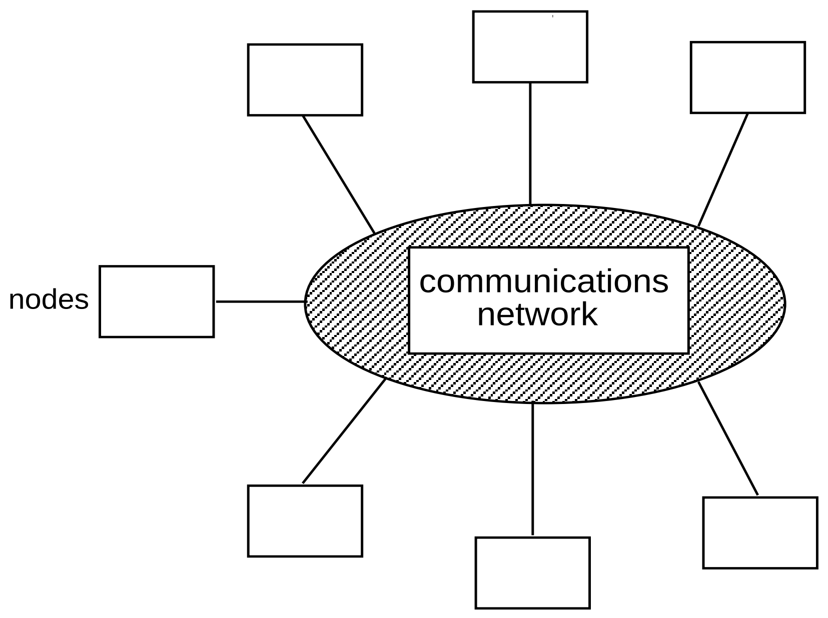

Figure 2.1: A distributed computer system

system of two or more processors that communicate via shared memory. A *distributed computer system* is any interconnection system of two or more computers (it is assumed that each computer has its own private memory). The interconnection structure must permit communication between any two computers (but not via shared memory). A number of studies for multiprocessor systems have been reported [11, 17, 18, 35, 61, 63, 99].

This section focuses on the related load balancing studies in distributed computer systems. Many papers that deal with load balancing algorithms model the distributed computer system being analyzed as a system that consists of a set of nodes connected in an arbitrary fashion by a communications network as illustrated in Figure 2.1. Through the communication network, resources (e.g., processors, computer servers, etc.) of the system can be shared by users at different locations.

From the user's point of view this set of resources acts like a *single virtual system*. As he submits a job for execution he does not and should not consider either the internal structure or the instantaneous load of the system. It is the duty of the system's load balancing algorithm to control the assignment of resources to jobs and to route the jobs according to these assignments.

A load balancing policy chooses the resources that should be used to run a job in order to improve a given performance measure. Load balancing problems are similar to determining an optimum routing policy for communications networks and an optimum traffic assignment policy for transportation networks, but there are some significant differences. In the routing and traffic assignment problem, a set of source-destination pairs, the traffic for each pair and cost constraints are specified. In the load balancing problem, there is no notion of source-destination traffic. Instead, there are collections of one or more resources which can perform a certain type of work and which we might call functionally equivalent subsystems. During execution, a job can choose (or be assigned) to access resources in a particular subsystem to obtain a certain type of service. Usually, the routing of jobs to the subsystem is not an issue. In some systems, jobs are grouped into classes and, for each class, resources are classified as either *local* or *remote*. If the load balancing algorithm chooses to execute a job at a remote resource, a penalty is paid (e.g., extra processing is needed) to transfer the job from its *local* node to the *remote* node. An important property of a load balancing policy is fairness of service, i.e., the system should operate in such a way that all jobs, regardless of their class, should be provided with specified *acceptable* levels of performance. Load balancing policies may be either *static* or *dynamic*.

2.1.1 Static Load Balancing

Static load balancing policies [24, 29, 31, 32, 33, 38, 39, 40, 41, 44, 48, 50, 58, 69, 83] use only the statistical information on the system (e.g., the average behavior of the system) in making load-balancing decisions, and their principal advantage is lower overhead cost needed to execute them and their simplicity in implementation and their mathematical tractability. They do not, however, adapt to fluctuations in workload. Under a situation where the system workload is statistically balanced, some computers may be heavily loaded at a given instant (hence suffering from performance degradation), while others are idle or lightly loaded. Static load balancing policies are useful for system sizing (e.g., allocation of resources, identification of bottlenecks, sensitivity studies, etc.). The results of optimal static load balancing may also help us design the system and make a parametric adjustment to improve the system performance [48, 50].

Static load balancing policies may be either *deterministic* (e.g., transfer all jobs originating at node A to node B) or *probabilistic* (e.g., transfer half of the jobs originating at node A to node B, and process the other half locally). The following paragraphs briefly describe some of the previous studies of static load balancing in distributed computer systems.

Tantawi and Towsley [74] studied a single job class model of a distributed computer system that consists of a set of heterogeneous host computers connected by a single channel communications network. In this model, nodes are represented by a number of resources, and different nodes may have different configurations and resources with different processing rates. Jobs arrive at each node according to a Poisson process with possibly different rates for each node. The model is required to be a product form queuing network. They considered an optimal static load balancing policy which determines the optimal load at each node so as to minimize the overall system mean job response time, and derived an algorithm (called a single-point algorithm) that determines the optimal load at each node for given system parameters. Ross and Yao [83] considered a more general problem consisting of dedicated and generic jobs. Dedicated jobs can be processed only on specified nodes, while generic jobs can be processed on any node in the system. And also they dealt with scheduling decision at each node. The authors have noted that the problem is separable over local scheduling decisions, and suggested a solution procedure based on this finding. They also showed that given an allocation of the jobs on the nodes, the task of scheduling can be solved as a polymatroid optimization problem. Mondal [69] considered the same model of Ross and Yao [83] with the same assumptions and his results only changes the allocation of the jobs on the nodes.

Kim and Kameda [15] considered the same model as Tantawi and Towsley [74] under the same assumptions and devised another single-point algorithm that seems more easily understandable and more straightforward than that of Tantawi and Towsley. They compared the performance of their algorithm with that of Tantawi and Towsley.

Also, Tantawi and Towsley [73] studied a distributed computer system that consists of a set of heterogeneous host computers (nodes) interconnected by a star network and they proposed a static load balancing algorithm that determines the optimal load at each node for given system parameters, so as to minimize the overall system mean job response time. On the basis of Tantawi and Towsley's work, Kim and Kameda [15] proposed an improved static load balancing algorithm for a distributed computer system with star network configuration. In Tantawi and Towsley's model [73], however, there is only one-way traffic from the external nodes to the central node in the sense that jobs can be forwarded for remote processing only from the external nodes to the central node. As an extension of this work, Li and Kameda [49] proposed an algorithm for optimal static load balancing in star network configurations with two-way traffic and then in [47, 48], they proposed an algorithm for optimal static load balancing in tree hierarchy network configurations.

Kameda and Zhang [46] studied the uniqueness of solutions in optimal static load balancing of open BCMP queuing networks. They obtained the linear relations that characterize the set of the optimal solutions. Thus the solution is unique if and only if the set of the optimal solutions reduces to a single point.

The models presented above deal only with single job class environment. In [13, 14, 16], Kim and Kameda extended the Tanatwi and Towsely single job class model [74] to multiple job class environment with almost the same assumptions of Tanatwi and Towsely and they proposed an optimal static load balancing algorithm for multiple job classes. As a generalization, Li and Kameda [50] proposed an optimal static load balancing algorithm in a multi-class jobs distributed/parallel computer system with general network configurations.

There are some significant differences between the problem of load balancing and that of routing for communications networks and traffic assignment for transportation networks as explained in section 2.1. In spite of the significant differences, the well known algorithms for flow assignment, the flow deviation (FD) algorithm [22, 62] and the Dafermos algorithm for traffic assignment [21, 71] can be applied to load balancing problems easily. Kim and Kameda [13] applied the two algorithms to load balancing problems and compared the performance of the two algorithms with the performance of their proposed load balancing algorithm for multi-class jobs. Also, Li and Kameda [50] applied the FD algorithm [22, 62] to load balancing problems and compared it's performance with the performance of their proposed load balancing algorithm for a multi-class jobs distributed/parallel computer system with general network configurations.

2.1.2 Dynamic Load Balancing

Dynamic load balancing policies [8, 25, 31, 32, 39, 40, 41, 57, 58, 59, 65, 68, 75, 84, 85, 88] attempt to dynamically balance the workload reflecting the current system state and are therefore thought to be able to further improve the system performance. Thus, it would be thought that, compared to static ones, dynamic load balancing policies are better able to respond to system changes and to avoid those states that result in poor performance. Obviously, the disadvantages of dynamic load balancing policies is that these policies are more complex than their static counterparts, in the sense that they require information on the runtime load and activities of state collection. Studies on dynamic load balancing have been usually limited to specific models that assume either that all the nodes in the system are identical or that the overheads involved in load balancing are negligible [8, 25, 31, 32, 39, 40, 41, 85].

Dynamic load balancing policies may be either preemptive or non-preemptive. A preemptive load balancing policy [28, 90, 96] allows load balancing to occur whenever the imbalance appears in the workloads among nodes. If a job that should be migrated to a new node is in the course of execution, its execution will be continued at the new node. On the other hand, a non-preemptive load balancing policy [25, 31, 32, 39, 40, 65, 68, 85, 98] assigns a newly arriving job to what appears at that moment to be the best node. Once the job execution begins, it is not moved even though its run-time characteristics, or the run-time characteristics of any other jobs, is changed after assigning the job in such a way as to cause the nodes to become much unbalanced. Since in most systems the service demands of jobs are not known before starting execution, with initial assignment jobs are assigned to nodes in ignorance of these demands. An initial distribution of jobs cross nodes that appears balanced will therefore become unbalanced as shorter jobs complete and leave behind an uneven distribution of longer jobs. Migration allows such imbalances to be corrected. To migrate a job in execution, however, is much complex and is accompanied with much overhead caused by gathering and transferring the state of the job, resulting in performance degradation.

This section focuses only on non-preemptive load balancing policies. A non-preemptive load balancing policy typically has three components:

1. A transfer policy that determines whether a job is processed locally or remotely.

2. A location policy that determines the node (server or processor) to which a job, selected for remote execution, should be sent.

3. An information policy that determines the amount of load information made available to the location policy and what load information should be collected and how this information is obtained.

A large number of the transfer policies proposed are *threshold* policies [8, 31, 32, 39, 40, 41, 59, 65, 68, 85, 98]. Typically, transfer policies use some kind of load index threshold to determine whether the node is heavily loaded or not (e.g. CPU queue length, CPU utilization, etc.). When this load index threshold is exceeded the load balancing condition is satisfied and the transferring mechanism is initiated. Location policy at a node determines the allocation of a job and takes the action of the transfer if the job is determined to be processed remotely. An information policy may be based on a *time-driven* or *event-driven*. In a time-driven approach, a node periodically announces its load information to other nodes or issues a request-for-bid message to other nodes to collect their load information.

Periodic policies do not adapt their activity to the system state. The overheads due to periodic information announcement or collection at hight system loads continue to increase the system load and thus worsen the situation. In an event-driven approach, on the other hand, a node does not announce its load information or issue a request-for-bid message for negotiation until its load changes. The information on the load state or the request-for-bid message at a node can be broadcasted to all other nodes, or only to a subset of the nodes or a single node. Since overhead and delay due to state information manipulation have strong effects on the performance of dynamic load balancing policies and can not usually be negligible, many researchers studied the effects of the amounts of the load state information on the performance of dynamic load balancing policies and they proposed many techniques to minimize the overheads cased by the state information manipulation [56, 60, 65, 68, 81]. Also, the effects of occasionally poor load balancing decisions and the potential for instability in dynamic load balancing because of the inherent inaccuracy of system state information have been studied in [65, 68].

Load balancing policies can be classified as *centralized* or *decentralized*. In centralized policies [8, 41, 54, 68, 88, 95], it may be considered as a system with only one load balancing decision maker. Arriving jobs to the system are sent to this load balancing decision maker, which distributes jobs to different processing nodes. The centralized policies has the advantages of easy information collection about job arrivals and departures and the natural implementation employing the server-client model of distributed processing. The major disadvantages of the centralized policies is the possible performance and reliability bottleneck due to the possible heavy load on the centralized job load balancing decision maker [95]. For this reason, the centralized approaches are not appropriate for large-scale systems. Furthermore, failure of the load balancing decision maker will make the load balancing inoperable. It appears that this policy is closely related to the *overall optimal policy* in that there is only one load balancing decision maker and it makes all the load balancing decisions.

The decentralized policies, on the other hand, delegates job distribution decisions to individual nodes. Usually each node accepts the local job arrivals and makes decisions to send them to other nodes based on its own partial information on the system load distribution. It appears that this policy is closely related to the *individually optimal policy* in that each job (or the user of each) optimizes its own cost (e.g., its own expected mean response time), independently of the others. The decentralized load balancing is widely used to handle the imperfect system load information [8, 41, 51, 52, 54, 60, 68, 95].

Decentralized load balancing policies can be broadly characterized as *sender-initiated*, *receiver-initiated*, and *symmetrically-initiated*. In sender-initiated policies [8, 41, 54, 60, 76, 81], congested nodes attempt to transfer jobs to lightly loaded ones. In the receiver-initiated policies [8, 41, 54, 60, 76], lightly loaded nodes search for congested nodes from which jobs may be transferred. Many policies have been analyzed, which combine the desired features of both sender and receiver-initiated techniques, and are called *symmetrically-initiated* [36, 54, 56]. They seek to find suitable receivers when senders wish to send jobs, and to find suitable senders when receivers wish to acquire jobs. Efficient symmetrical policies (e.g. [55]) behave as sender-initiated under low and mediate load conditions, and as receiver-initiated under heavy load conditions, following the corresponding result of Eager, Lazowska, and Zahorjan [60]. The following paragraphs briefly describe some of the previous studies of the dynamic load balancing in distributed computer systems.

Eager, Lazowska, and Zahorjan [59, 60] provide an analytic study of dynamic load balancing policies. They showed that the sender-initiated policy performs better at low to moderate system loads and the receiver-initiated policy performs better at hight system loads. They have also shown that the overhead associated with state information collection and maintenance under the distributed policy can be reduced substantially by probing only a few randomly selected nodes about their system state as opposed to all nodes in the system. Shivaratri and Krueger [36] have proposed and evaluated, using simulation, two location policies that combine the good features of the sender-initiated and receiver-initiated location policies. Schaar, Efe, Delcambre and Bhuyan [70] studied the impact of the communication delay on the performance of some dynamic load balancing policies.

Hac, and Jin [1] have implemented a receiver-initiated algorithm and evaluated its performance under three workload types: CPU-intensive, IO-intensive, and mixed workloads. They compared the performance of their load balancing policy with that when no load balancing is employed. They found that, for all the three types of workload, load balancing is beneficial. Unfortunately, they did not compare the performance of various load balancing policies that have been proposed in the literature. Also, in [2], they studied sender initiated and receiver initiated load balancing strategies. In these strategies, the system load is balanced in terms of the number of active processes on each host. A migration factor is considered, defined as the ratio of the mean transfer time to the response time of a process executed locally. If the migration factor is less than or equal to one, the process is declared as migrant, otherwise no action is taken. Their study is limited to independent applications.

Dikshit, Tripathi, and Jalote [78] have implemented both sender-initiated and receiver-initiated policies on a five node system connected by a 10Mb/s communication network. As a part of their study they have conducted an experiment on the impact of service time variance, but the coefficient of variation is less than or equal to 1 (taken from exponential and uniform distributions).

Dandamudi [76] evaluated the performance of three node scheduling policies: First-Come/First Served (FCFS), Shortest Job First (SJF), Round Robin (RR), combined with the sender-initiated and receiver-initiated load balancing. Furthermore, he looked at the impact of variance in the interarrival times and in the job service times. Dasgupta, Majumber, and Bhattacharya [77] proposed one of the newer dynamic, symmetrical, distributed, and efficient algorithms, called the Variable Threshold (V_THR) algorithm. They used it for dynamic load balancing on a shared BUS architecture, which monitors the threshold for the starting of load balancing, to dynamically adapt itself to the limited bandwidth of the shared BUS architecture. Antonis, Garofalakis, Mourtos, and Spirakis [54] proposed a dynamic, distributed hierarchical scheme, called the Virtual Tree Algorithm (VTA), which creates and uses a virtual binary tree structure over the actual network topology. It introduces the basic concept of conjugate nodes in multiple levels in the tree. Their algorithm needs remote information only for the transfer policy, and no additional information for the location policy. They proved that the proposed virtual construction can keep the exchanging messages to a number comparable to those of the previous efficient algorithms. And they compared the performance of their algorithm (VTA) with that of the V_THR algorithm that is proposed by Dasgupta, Majumber, and Bhattacharya [77].

Deng, Liu, Long, and Xiao [95] measured the information efficiency of a load balancing policy by the competitive ratio of the solution (for each load distribution) of a load balancing policy to the optimal solution (for the same load distribution) assuming that nodes have complete information about the load distribution over the network. They showed that when jobs have different sizes, even with preemptive scheduling, the load balancing policy is

NP-complete. When the jobs are of the same size, they gave a polynomial algorithm, using network-flow techniques, which extends to approximate solutions for jobs of different sizes. They also applied this benchmark solution for three network topologies: completely connected graphs, rings, and hierarchical complete k-ary trees. Stefano, Bello, and Mirabella [20] assess job allocation on heterogeneous computer networks. They argue that the use of minimum global information can contribute to improve the performance of a load balancing policy to a significant degree. The performance of random allocation policy is compared with two partially global job allocation policies. (1) Threshold policy selects a node at random and enquires if it has exceeded its load threshold. If it has not, the job is transferred to it. (2) Shortest policy selects a group of nodes randomly, acquires the load information on each and makes the allocation decision accordingly. As a conclusion the results show that even partial global information provides important performance improvement.

Mitzenmacher [68] studied the effect of occasionally poor load balancing decisions and the potential for instability in dynamic load balancing because of the inherent inaccuracy of system state information. Also, Dahlin [65] studied the same problem and he proposed load interpretation strategies that interpret system load information based on its age. Through simulation, he examined several simple algorithms that use such load interpretation strategies under a range of workloads. Bozyigit [64] presented a new dynamic load balancing scheme, called DYLOBA, where both the current system load and the load to be exerted by the application are equally important. The target system chosen is a general purpose network of workstations. The approach utilizes the past execution statistics of the applications. In this sense, information on the run time system load and resource requirement of the applications, averaged over past executions, is integrated.

Hui and Chanson [12] presented a hydrodynamic framework for solving the dynamic load balancing problem on a network of heterogeneous computers. In this approach, each processor is viewed as a liquid cylinder where the cross-sectional area corresponds to the capacity of the processor, the communication links are modelled as liquid channels between the cylinders, the workload is represented as liquid, and the load balancing algorithm describes the flow of the liquid. It is proven that all algorithms under this framework converges geometrically to the state of equilibrium, in which the heights of the liquid columns are the same in all the cylinders.

Altman and Shimkin [25] studied the effect of projected load buildup on individual user decisions and consequently on the system performance, in shared facility. Assuming that the users are symmetric, they have shown the existence of a unique equilibrium point, and how this equilibrium emerges as a result of simple learning scenario. Karatza and Hilzer [58] studied the effects of load balancing on the performance of a heterogeneous distributed computer system, where half of the total processors have double speed of the others. They considered two job classes. Programs of the first class are dedicated to fast processors, while second class programs are generic in the sense that they can be al-located to any processor. Their objective was to find a policy that results in good overall performance while maintaining the fairness of individual job classes. Through simulation, they examined and compared the processor performance under a variety of workloads. Their results show that the performance of the best method depends on system workload.

Tiemeyer and Wong [90] presented a distributed, dynamic load balancing algorithm for fully-connected distributed computing systems. In this work, they described a method through which the communication protocol can be tailored to the capabilities of the system's individual processors. Also, they described modifications designed to make the scheme fault tolerant. These modifications handle those cases in which one or more processors are considered nonfunctional. Watts, and Taylor [53] proposed a practical, comprehensive approach to dynamic load balancing that has been applied to nontrivial applications. Incorporated into the approach are a new diffusion algorithm, which offers a good trade-off between total work transfer and run time, and a task selection mechanism, which allows task size and communication costs to guide task movement.

Mirchandaney, Towsley, and Stankovic [85] studied the performance characteristics of simple load balancing algorithms for heterogeneous distributed systems. They assumed that a non-negligible delays are encountered in transferring jobs from one node to another and in gathering remote state information. They analyzed the effect of these delays on the performance of two threshold-based algorithms. Also, they formulated queueing theoretic models for each of the algorithms operating in heterogeneous systems under the assumption that the job arrival process at each node in Poisson and the service times and job transfer times are exponentially distributed. They solved these models using Matrix-Geometric solution technique. And they used these models to study the effects of different parameters and algorithm variations on the mean job response time: e.g., the effect of varying the thresholds, the impact of changing the probe limit, the impact of biasing the probing, and the optimal response times over a large range of loads and delays.

We found a very few number of works that considered the problem of comparing between the performance of static and dynamic load balancing policies. The following paragraphs briefly describe these studies.

Iqbal, Saltz, and Bokhari [4] studied the problem of uniformly distributing the load of a parallel program over a multiprocessor system. In this work, they described and analyzed four policies for load balancing. And, they compared the performance of these policies on a set of problems whose structure permits the use of the four policies. The considered four policies are (1) the optimal static assignment algorithm which is guaranteed to yield the best static solution, (2) the static binary dissection method which is very fast but sub-optimal, (3) the greedy algorithm, a static fully polynomial time approximation scheme, which estimates the optimal solution to arbitrary accuracy and (4) the predictive dynamic load balancing heuristic which uses information on the precedence relationships within the program. Through simulation, they showed that the dynamic policy outperforms any of the static methods, and the overhead incurred by the dynamic heuristic (4) is reduced considerably if it is started off with a static assignment provided by either (1), (2), or (3).

In [41, 97, 98], the authors compared through simulation the performance of two dynamic and two static load balancing policies in a heterogeneous distributed computer system model. They assumed that all the nodes in the system have the same function but possibly different capacities, and the overheads and the delays for both job transfer and system state-information exchange are non-negligible. Their simulation results show that both dynamic and static policies improve performance dramatically, and that the performance provided by the static policies is not much inferior to that provided by the dynamic policies. They also showed that when overheads are non-negligibly high at heavy system loads, static policies can provide performance more stable and better than that provided by the considered dynamic policies.

In the previous studies, the comparison between the performance of the static and dynamic policies is done through simulation. To the best of our knowledge, there is no work that compares analytically between the performance of static and dynamic load balancing policies in a distributed computer system model. For this reason in [32, 39, 40], we analytically compare between the performance of a static overall optimal load balancing policy (SOOLBP) and a dynamic overall optimal load balancing policy (DOOLBP) in a distributed computer system that consists of two types of service facilities, a Mainframe node Q_{MF} and an unlimited number of Personal Computer nodes Q_{PC}, both of which are connected by a communication network. Truly optimal solutions of both SOOLBP and DOOLBP have been characterized. The overheads due to the two policies are assumed to be negligible. The $[L, q]$ threshold rule is considered as a DOOLBP. A numerical algorithm for obtaining the optimal values of the threshold parameters L and q is proposed. Analytically, it is proved that the minimum value of the overall system mean response time is obtained by the DOOLBP with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. Also, we analytically proved the existence and uniqueness of optimal solution of the other threshold parameter L. That is, we need to choose only the proper value of L with q fixed to be 0 in finding the set of parameter values of the threshold rule that gives the minimum value for the overall system mean response time. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the overall system mean response time using the SOOLBP and DOOLBP are studied through numerical experimentation. The results show that, in the model examined, the overall system mean response time is improved by the DOOLBP over that of the SOOLBP at most about 30% in the range of parameter values examined while the overheads due to the two policies are not taken into account. And, the maximum improvement ratio is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase.

Also, in [31], we analytically compare between the performance of a static individually optimal load balancing policy (SIOLBP) and a dynamic individually optimal load balancing policy (DIOLBP) on the same model that is considered in [32, 39, 40]. The overheads due to the two policies are assumed to be negligible. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the mean job response time using the SIOLBP and the DIOLBP are studied through numerical experimentation. The results show that the DIOLBP outperforms the SIOLBP in the overall mean response time, at most about 48% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model under the SIOLBP and the DIOLBP. We found that, there is a difference between the ratio that a job arriving at the system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase. Through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have effect in satisfying the equilibrium in between the two system facilities. And also, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ and it approaches μ/θ. Additionally, several interesting phenomena are also observed.

2.2 Braress Paradox: A survey

Intuitively, we can think that the total processing capacity of a system will increase when the capacity of a part of the system increases, and so we expect improvements in performance objectives accordingly in that case. The famous Braess paradox tells us that this is not always the case; i.e., increased capacity of a part of the system may sometimes lead to the degradation in the benefits of all users in an individual optimum [10, 19, 27]. The Braess Paradox attracted the attention of researchers in many fields such as Arora and Sen [72] in the field of Software Multi-Agent Systems, Roughgarden and Tardos [91] in the Theory of Computing, Cohen and Kelly [27], Kelly [80] and Cohen and Jeffries [26] in queueing networks, Kelly [79] and Bean, Kelly and Taylor [34] in loss networks and Kameda *et al* [38, 42, 45] in distributed computational systems. The following paragraphs briefly describe some of the previous studies related to this topic.

Braess [19] discovered a deterministic mathematical model of a congested network such that, paradoxically, when a link (path) is added and each user seeks his best possible path, at the new equilibrium, the mean response time for all users is higher than before. At equilibrium, independently self-seeking users are unable to ignore that added capacity that ends up increasing their response time.

Clavert [9] supposed a Poisson stream of arriving users to a distributed processing system and they have a dynamic load balancing policy which gives them the quickest path. He analytically showed an example where increasing the processing capacity of a server in the considered model can lead to increasing the mean response time in equilibrium.

Cohen and Kelly [27] reported the first example of Braess's paradox in a mathematical model of a queueing network. They investigated Braess's paradox in the setting where the users (arrivals) have knowledge only of mean queue lengths of the network servers that is they used a static load balancing policy.

Cohen and Jeffries [26] reported some examples of single-server queueing networks in which adding servers or increasing the processing capacity of existing servers leads to degrading the network performance. Kameda [37] used a static load balancing policy to study the problem of estimating the worst case ratio of performance degradation caused by adding capacity for the sharing of jobs between nodes in networks generalized from what were studied by Cohen, Kelly and Jeffries [26, 27] in comparison with the networks of the same topology as the original Braess network [19]. In his work, the measure of performance degradation considered is the ratio of the mean response time for each user of a network after adding capacity to that before adding capacity, which means that the network has performance degradation if the measure is greater than one. And he showed that a value of the measure is less than 2 for every general Braess network and the worst case is obtained in a symmetric reduced Cohen-Kelly network.

The famous Braess paradox tells us that increased capacity of a part of the system may sometimes lead to the degradation in the benefits of all users in an individual optimum [10, 19, 27]. As it is known that the *class optimum* converges to the *individual optimum* as the number of classes becomes large [3], we can expect that, in the class optimum, a similar type of paradox occurs (with large number of classes), i.e., increased capacity of a part of the system may lead to the degradation in the benefits of all classes in a class optimum, whenever it occurs for the individual optimum. We call it the *Braess-like paradox*. Indeed in [5], Korilis *et al.* found some examples wherein the Braess-like paradox appears in a class optimum where all user classes are identical in the same topology for which the original Braess Paradox (for the individual optimum) was in fact obtained. Furthermore in [6], he also obtained a sufficient condition under which the Braess Paradox should not occur in a more general model that has one source-destination pair and identical user classes.

In a model that has asymmetric classes; i.e., classes are not identical, Kameda *et al.* [38] have obtained, however, numerical examples where a paradox similar to Braesss appears in the class optimum but does not occur in the individual optimum in the same environment. These cases look quite strange if we note that such a paradox should never occur in the overall optimum and if we regard the class optimum as an intermediate between the overall optimum and the individual optimum. Later on, in [43] he also showed that the worst-case degree of the paradox (WCDP) may increase without bound in class optimum where the values of parameters of all classes are identical and also it has been shown that this strange behavior (i.e., the WCDP may increase without bound) does not occur for the overall and individual optimum, in the same setting of the system parameters. To the best of our knowledge, [43] is the first paper that reported such a case where the WCDP can increase without bound. In [29, 30, 33], we studied the dependence of the WCDP on the system parameter setting through a number of numerical examples around the Braess-like paradox in a distributed computer system. Each node in the system has, at its disposition, a communication means, which it may use to forward to other nodes an arbitrary portion of its job arrival stream. We considered three different types of communication means (A), (B) and (C). Based on the system parameter setting, we defined three different types of symmetries: *overall symmetry, individual symmetry and complete symmetry*. From the numerical examples, it is observed that in the class optimum, the WCDP is largest in the complete symmetry case when the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

Chapter 3

A Comparative Study of Static and Dynamic Overall Optimal Load Balancing Policies in a Mainframe – Personal Computer Network Model

3.1 Introduction

As technology has quickly and relentlessly advanced in the field of computer hardware, distributed computer systems have become increasingly popular. A distributed computer system is considered to be a collection of autonomous computers (nodes) located at possibly different sites and connected by a communication network. Through the communication network, resources of the system can be shared by users at different locations. Distributed computer systems, such as networks of workstations or mirrored sites on the World Wide Web, face the problem of using their resources effectively. If some hosts lie idle while others are extremely busy, system performance may fall significantly. Performance enhancement is one of the most important issues in distributed systems. The performance of a distributed computer system can often be improved to an acceptable level by redistributing the workload among nodes. The problem of load redistribution in distributed computer systems is called *load balancing*. A number of load balancing policies have been proposed to improve the performance of distributed/parallel systems (e.g., to minimize the mean job response time, to maximize the processing capacity of the system) by efficiently utilizing the processing power of the entire system. Although a communication delay is incurred in transferring a job from one node to another, the performance of a distributed computer system can generally be improved by an effective load balancing policy [51, 52, 59, 86, 92]. Load balancing policies may be either *static* or *dynamic*.

Static load balancing policies [8, 15, 41, 74, 98] use only the statistical information on the system (e.g., the average behavior of the system) in making load-balancing decisions, and their principal advantage is lower overhead cost needed to execute them and their simplicity in implementation and their mathematical tractability. They do not, however, adapt to fluctuations in the workload. Under a situation where the system workload is statistically balanced, some computers may be heavily loaded at a given instant (hence suffering from performance degradation), while others are idle or lightly loaded.

On the other hand, dynamic load balancing policies [8, 41, 57, 59, 75, 84, 85, 88] attempt to dynamically balance the workload reflecting the current system state and are therefore thought to be able to further improve the system performance. Thus, it would be thought that, compared to static ones, dynamic load balancing policies are better able to respond to system changes and to avoid those states that result in poor performance. However, this is not always the case. In [97, 98] it have been shown through simulation that when overheads are non-negligibly high at heavy system loads, static load balancing policies can provide performance more stable and better than that provided by some dynamic load balancing policies. Obviously, the disadvantage of dynamic load balancing policies is that these policies are more complex than their static counterparts, in the sense that they require information on the runtime load and activities of state collection. The effect of occasionally poor load balancing decisions and the potential for instability in dynamic load balancing because of the inherent inaccuracy of system state information have been studied in [68].

Generally, the purpose of load balancing policies either static or dynamic is to improve the performance of the system by redistributing the workload among nodes. We can choose between several distinct objectives for performance optimization in many systems including communication networks, distributed computer systems, transportation flow networks, etc. Among them, we have three typical objectives or optima:

1. *The overall optimum*, where all jobs are regarded to belong to one group that has only one decision maker. The decision maker seeks to optimize a certain overall and single performance measure like the total cost or the overall mean response time (the expected value of the time length that starts when a job arrives at the system and ends when the job leaves the system after the processing of the job is completed) over all the jobs. We call an optimal load balancing policy whereby the overall mean response time is minimized the *overall optimal policy*. By the *overall optimization* problem we mean the problem of obtaining the load balancing decision that achieves the objective of the overall optimal policy. In the literature, the solution of the overall optimization problem is referred to as system optimum, overall optimum, cooperative optimum or social optimum. In this thesis, we shall refer to it as the *overall optimum*.
2. *The individual optimum*, where each of infinitely many jobs (or the user of each) optimizes its own cost (e.g., its own expected response time) independently of the others.

In this optimized situation, each job cannot expect any further benefit by changing its own decision. It is also assumed that the decision of a single job has a negligible impact on the performance of other jobs. We call an optimal load balancing policy whereby every job strives to optimize (minimize) its own mean response time independently of the other jobs the *individually optimal policy*. By the *individual optimization* problem we mean the problem of obtaining the load balancing decision that achieves the objective of the individually optimal policy. In the literature, the solution of the individual optimization problem is referred to as an individual optimum, Wardrop equilibrium, or user optimum. In this thesis, we shall refer to it as the *individual optimum*.

3. *The class optimum*, where infinitely many jobs are classified into a finite number ($N > 1$) of classes or groups, each of which has its own decision maker and is regarded as one player or user. Each decision maker optimizes non-cooperatively its own cost (e.g., the expected response time) over only the jobs of its own class. The decision of a single decision maker of a class has a non-negligible impact on the performance of other classes. In this optimized situation, each of a finite number of classes or players cannot receive any further benefit by changing its decision. We call the load balancing policy that has the previous description the *class optimal policy*. By the *class optimization* problem we mean the problem of obtaining the load balancing decision that achieves the objective of the class optimal policy. In the literature, the solution of the class optimization problem is referred to as the class optimum, or Nash equilibrium. In this thesis, we shall refer to it as the *class optimum*.

Note that the class optimum is reduced to the overall optimum when the number of classes reduces to 1 ($N = 1$) and approaches the individual optimum when the number of classes becomes infinitely many ($N \rightarrow \infty$) [3].

In this chapter, we propose *a static and a dynamic overall optimal load balancing policies* whereby job scheduling is determined so as to minimize the overall system mean response time of jobs in the whole system. That is, the goal of the two policies is system wide optimization. Our ultimate goal is to examine to what extent the dynamic overall optimal load balancing policy outperforms the static one by an exhaustive numerical investigation on a model for which both policies are analytically studied. Optimal static load balancing policies have been analytically studied in a variety of models for distributed computer systems [15, 48, 73, 74, 98]. On the other hand, as far as we know, optimal dynamic load balancing policies have been studied only in very specific models: one is that of using an M/M/*m* queueing model [59], and another is what is analytically studied in [25]. The latter is the model studied here that consists of a Mainframe node Q_{MF} and an unlimited number of Personal Computer nodes Q_{PC}. The $[L, q]$ threshold rule is considered as a dynamic overall optimal load balancing policy. In this rule, a job arriving at the Q_{PC} node is forwarded to the Q_{MF} node with probability 1 if the number of jobs staying at the Q_{MF} node is less than L, with probability q if the number equals L, and otherwise is processed by the Q_{PC} node. The model allows us to have exhaustive numerical investigation to gain insight into the problem. The objective of both policies is to minimize the overall system mean response time. The performance of these two policies is compared on the considered model where truly optimal solutions of both static and dynamic overall optimal load balancing policies have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, we do not take account of the difference in the overheads due to the two load balancing policies. The model we consider here is analytically studied in [25] and is motivated in part by the following scenario.

Potential computer users, each requiring the use of a computer to execute a given job, arrive sequentially at a computer facility. As it has been stated in [25], the model studied here is relevant to a wide range of application areas that involve shared service such as computing and telecommunications. We mention here two relevant applications in the latter area:

1. Consider a situation where users can communicate with each other either through a Local Area Network (LAN) or through the public network, e.g., by connecting to the telephone network via a modem, which is typically slower. However, the throughput available to each user on the LAN decreases as the total workload increases. This is specially the case in LANs where a single channel should be shared between all users, e.g., the Fiber Distributed Data Interface (FDDI). The LAN can thus be approximated by a processor sharing queue, whereas the public network can be viewed as assigning a private server to each session.

2. Consider a non-real time application, such as data transfer, on an Asynchronous Transfer Mode (ATM) network. ATM networks support both guaranteed services as well as best-effort services.

    (a) Guaranteed services are Constant Bit Rate (CBR), in which a fixed amount of bandwidth is assigned to a session, and Variable Bit Rate (VBR), in which some average and peak bit-rates are assigned to a session.

    (b) Best-effort services are Available Bit Rate (ABR) and Unspecified Bit Rate (UBR); in both cases, some available bandwidth is shared among the connections that use these services.

At a session level, ABR and UBR services can be approximated by a processor sharing queue, whereas CBR and VBR services can be approximated by a single server, dedicated for one session. For more information about the real applications to the MF-PC network model see [25].

While there have been some studies of performance comparison between dynamic and static load balancing policies in more sophisticated models where overheads are considered [41, 98], the truly optimal dynamic policy is not accurately obtained in contrast to the model considered here. The results obtained here show that, in the model examined, the dynamic overall optimal load balancing policy outperforms the static one in the overall system mean response time, at most about 30% in the range of parameter values examined while the overheads due to the two policies are not taken into account. Another remarkable finding is that the minimum value of the overall system mean response time is achieved by the $[L, q]$ threshold rule with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. Also, we proved the existence and uniqueness of the optimal solution of the other threshold parameter L. That is, we need to choose only the proper value of L with q fixed to be 0 in finding the set of parameter values of the threshold rule that gives the minimum value for the overall system mean response time.

3.2 Model Description and Assumptions

We consider a distributed computer system that consists of two types of service facilities, a Mainframe node Q_{MF} and unlimited number of Personal Computer nodes Q_{PC}, both of which are connected in an arbitrary fashion by a communication network as shown in Figure 3.1. We call this system model an *MF-PC network model*. We assume that the expected communication delay between the Q_{MF} node and the Q_{PC} node is negligible.

Jobs arrive at the system according to a time-invariant Poisson process, i.e. inter-arrival times of jobs are independent, identically and exponentially distributed with mean $1/\lambda$. Simultaneous arrivals are excluded. A job arriving at the system may be processed either by the Q_{MF} node or by the Q_{PC} node according to load balancing policies. We assume that the service rate at Q_{MF} is μ and that its service discipline is first-come-first-served (FCFS), or processor sharing whereby the service rate for each job equals $v(n) = \mu/n$ when the number of jobs in the Q_{MF} node is n. The Q_{PC} node offers a fixed expected service time θ^{-1}. In the Q_{PC}, service starts immediately upon admission, and thus the mean response time is identical to the service time. We assume that at both Q_{MF} and Q_{PC}, service times are independent, identically and exponentially distributed.

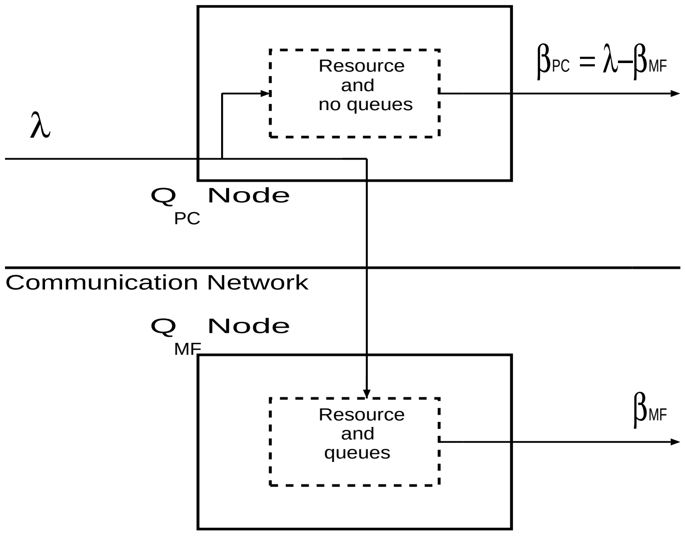

Figure 3.1: A model of an MF-PC network system

3.3 Two Optimal Load Balancing Policies

In the following two subsections, we present a static overall optimal and a dynamic overall load balancing policies and their solutions.

3.3.1 Static Overall Optimal Load Balancing Policy (SOOLBP)

It is very important to design and implement computer systems that have good performance. We may have, however, several performance measures in evaluating the performance of computer systems. To evaluate and optimize the performance of a computer system, we are forced to use one out of several performance measures. It seems that the most common performance measure is the overall mean response time, which is defined to be the expected value of the time length that starts when a job arrives at the system (i.e., an arbitrary node) and ends when the job leaves the system after the processing of the job is completed. In this section, we consider a static overall optimal load balancing policy that determines the optimal load at each node so as to minimize the overall system mean response time in our system model.

In this policy, the decision of transferring a job from one node to another does not depend on the state of the system, and hence is *static* in nature. Also, we assume that a job transferred from one node to another receives its service there, and is not further transferred.

We use the following notation:

- β_{MF}: Job processing rate (load) at the Q_{MF} node.

- $F_{MF}(\beta_{MF})$: Expected delay of a job processed at the Q_{MF} node.

With the considered model we have:

$$F_{MF}(\beta_{MF}) = \begin{cases} \dfrac{1}{\mu - \beta_{MF}} & \text{if } \beta_{MF} < \mu; \\ \infty & \text{otherwise.} \end{cases} \tag{3.1}$$

The problem of minimizing the overall system mean response time is expressed as

$$\min D(\beta_{MF}) = \frac{1}{\lambda}[\beta_{MF} F_{MF}(\beta_{MF}) + (\lambda - \beta_{MF})\theta^{-1}] \tag{3.2}$$

with respect to β_{MF} such that $0 \le \beta_{MF} \le \lambda$.
Define β_0 $(0 \le \beta_0 < \mu)$ such that $\dfrac{\mu}{(\mu - \beta_0)^2} = \theta^{-1}$.

The optimal load at the Q_{MF} node (β_{MF}) is given as follows:

$$\beta_{MF} = \begin{cases} \beta_0 & \text{if } \beta_0 < \lambda; \\ \lambda & \text{if } \lambda \le \beta_0. \end{cases} \tag{3.3}$$

3.3.2 Dynamic Overall Optimal Load Balancing Policy (DOOLBP)

By this policy, each arriving job may observe the current load in the Q_{MF} node, and then choose whether to join the shared mainframe or to remain at the Q_{PC} node. Also, the goal of this policy is to minimize the overall system mean response time.

A class of threshold load balancing policies have been shown to be useful when jobs are completely independent and consists of single threads of control. This situation is fairly common in networks of workstations. Such threshold policies contain control parameters (e.g. threshold values and transfer probability for every host), that require fine-tuning in order to yield optimal or near optimal performance. For more information about the use of the threshold load balancing policies see [25, 40, 59, 82, 87, 92, 94].

We use the $[L, q]$ threshold rule as a dynamic overall optimal load balancing policy. In this rule, an arriving job will go to the Q_{MF} node with probability of, respectively, 0, q, and

1, if the job finds that the Q_{MF} node has, more than, equal to, and less than, L jobs. We consider a formula $E\left[W_{[L,q]}\right]$ for the overall mean response time of the system with respect to $[L, q]$ threshold rule and minimize $E\left[W_{[L,q]}\right]$. The overall mean response time of a job arriving at the system with threshold $[L, q]$, $E\left[W_{[L,q]}\right]$, is obtained as follows:

$$E\left[W_{[L,q]}\right] = P\theta^{-1} + Q\lambda^{-1}, \tag{3.4}$$

where, if $\rho \neq 1$ (i.e. $\lambda \neq \mu$),

$$\begin{aligned} P &= P_0(1 - q + q\rho)\rho^L, \\ Q &= P_0\rho\frac{(-(L+1)\rho^L)(1-\rho) + (1-\rho^{L+1})}{(1-\rho)^2} \\ &\quad +(L+1)P_0 q\rho^{L+1}, \\ P_0 &= \frac{1-\rho}{1-\rho^{L+1}(1-q) - q\rho^{L+2}}, \end{aligned}$$

and if $\rho = 1$ (i.e. $\lambda = \mu$),

$$P = \frac{1}{L+1+q}, \quad Q = \frac{(L+1)(L+2q)}{2(L+1+q)},$$

where:

- P is the probability that an arriving job at the system goes to the Q_{PC} node.
- Q is the expected number of jobs (which includes the jobs in service) in the Q_{MF} node from state 0 to state $L + 1$ (see Figure 3.2).
- P_0 is the probability that the number of jobs in the Q_{MF} node is 0.

For the derivation of $E\left[W_{[L,q]}\right]$, see Appendix A.

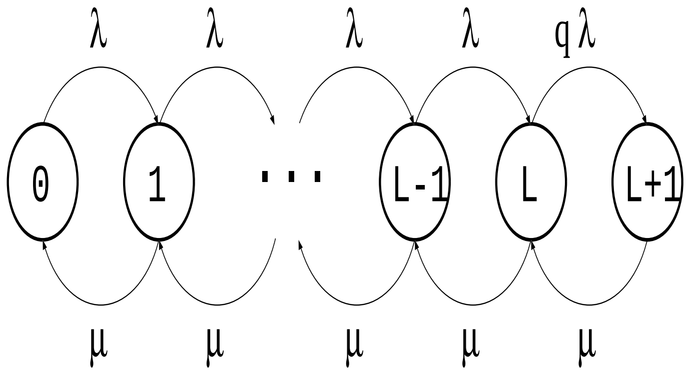

Figure 3.2: State transition diagram

Observing that the overall system mean response time does not depend on the service discipline in the Q_{MF} node (PS, FCFS, etc.), the problem reduces to that of a standard queueing control.

Proposition 3.3.1 *The overall system mean response time is minimized by the [L, q] threshold policy with the value of threshold parameter q=0 and the suitable selection of the other threshold parameter L.*

proof: Note that the $[L, 1]$ threshold policy is identical to the $[L + 1, 0]$ threshold policy. It is sufficient to show that, given λ, μ, θ and L, $E\left[W_{[L,q]}\right]$ is monotonically non-decreasing or non-increasing in $q \in [0, 1]$. That is, either $\frac{\partial}{\partial q}E\left[W_{[L,q]}\right] \geq 0$ for all $q \in [0, 1]$, $\frac{\partial}{\partial q}E\left[W_{[L,q]}\right] \leq 0$ for all $q \in [0, 1]$, or $\frac{\partial}{\partial q}E\left[W_{[L,q]}\right] = 0$ for all $q \in [0, 1]$.

It can be shown as follows. Given λ, μ, θ and L, we have the following two distinct cases:

- Case (1): $\rho = 1$ (i.e. $\lambda = \mu$) and $q \in [0, 1)$
- Case (2): $\rho \neq 1$ (i.e. $\lambda \neq \mu$) and $q \in [0, 1)$

Case (1): $\rho = 1$ (i.e. $\lambda = \mu$) and $q \in [0, 1]$

$$E\left[W_{[L,q]}\right] = \frac{1}{(L+1+q)}\theta^{-1} + \frac{(L+1)(L+2q)}{2(L+1+q)}\lambda^{-1}, \tag{3.5}$$

$$\frac{\partial E}{\partial q} = \frac{-2\lambda + (2+3L+L^2)\theta}{2\lambda\theta(L+1+q)^2}. \tag{3.6}$$

Case (2): $\rho \neq 1$ (i.e. $\lambda \neq \mu$) and $q \in [0, 1]$

$$E\left[W_{[L,q]}\right] = P\theta^{-1} + Q\lambda^{-1}, \tag{3.7}$$

where

$$P_0 = \frac{1-\rho}{1-\rho^{L+1}(1-q) - q\rho^{L+2}}, \tag{3.8}$$

$$\begin{aligned} Q &= P_0\rho\frac{(-(L+1)\rho^L)(1-\rho) + (1-\rho^{L+1})}{(1-\rho)^2} \\ &\quad + (L+1)P_0 q\rho^{L+1}, \end{aligned} \tag{3.9}$$

$$P = P_0(1-q+q\rho)\rho^L. \tag{3.10}$$

Hence,

$$\frac{\partial E}{\partial q} = \frac{\rho^L(C_1 - C_2)}{\lambda\theta(1+\rho^{L+1}(q-1) - q\rho^{L+2})^2}, \tag{3.11}$$

where

$$C_1 = \theta\rho(1+L-2\rho-L\rho+\rho^{L+2}), \tag{3.12}$$

$$C_2 = \lambda(\rho-1)^2. \tag{3.13}$$

In both of the above two cases, the numerators of (3.6) and (3.11) are independent of q whereas the denominators depend on q and remain positive for all $q \in [0, 1]$. We therefore see that $E\left[W_{[L,q]}\right]$ is either monotonically non-increasing or non-decreasing in $q \in [0, 1]$, given λ, μ, θ and L.

Proposition 3.3.2 *Given λ, μ, and θ, there exists $\hat{L}$ such that $E\left[W_{[L,0]}\right] - E\left[W_{[L-1,0]}\right] < 0$ for $L \leq \hat{L}$ and $E\left[W_{[L,0]}\right] - E\left[W_{[L-1,0]}\right] > 0$ for $L > \hat{L}$.*

That is, the response time function decreases in L for $0 \leq L \leq \hat{L}$ and increases in L for $L > \hat{L}$.

proof: Given λ, μ, θ, and $q = 0$, we have the following two distinct cases:

- Case (1): $\rho = 1$ (i.e., $\lambda = \mu$)
- Case (2): $\rho \neq 1$ (i.e., $\lambda \neq \mu$)

Case (1): $\rho = 1$ (i.e. $\lambda = \mu$)

$$E\left[W_{[L,q]}\right] = \frac{1}{(L+1)}\theta^{-1} + \frac{L}{2}\lambda^{-1}, \tag{3.14}$$

$$\frac{\partial E}{\partial L} = \frac{1}{2\lambda} - \frac{1}{(L+1)^2\theta}, \tag{3.15}$$

$$\frac{\partial^2 E}{\partial L^2} = \frac{2}{(L+1)^3\theta}, \tag{3.16}$$

Thus, $\frac{\partial^2 E}{\partial L^2} \geq 0$ for all values of $L \geq 0$, which means that, the response time function $E\left[W_{[L,q]}\right]$ is convex and hence, it has only one minimum point.

Case (2): $\rho \neq 1$ (i.e. $\lambda \neq \mu$)

$$\begin{aligned} E\left[W_{[L,q]}\right] &= \frac{1}{\mu}\left[\frac{\rho^L}{(1-\rho^{L+1})}\left[(1-\rho)\frac{\mu}{\theta} - (L+1)\right]\right. \\ &\quad \left. + \frac{1}{(1-\rho)}\right] \end{aligned} \tag{3.17}$$

$$\frac{\partial E}{\partial L} = \frac{\rho^L(\theta(\rho^{L+1}-1) - ((L+1)\theta + (\rho-1)\mu)\log\rho)}{\mu\theta(\rho^{L+1}-1)^2}. \tag{3.18}$$

Since $\frac{\rho^L}{\mu\theta(\rho^{L+1}-1)^2} > 0$, from the above equation, it is seen that the sign of $\frac{\partial E}{\partial L}$ depends on the value of $\Delta(L) = \theta(\rho^{L+1} - 1) - [(L + 1)\theta + (\rho - 1)\mu] \log(\rho)$. Then, $\frac{d}{dL}\Delta(L) = \theta(\rho^{L+1} - 1) \log(\rho)$.

Note that $\Delta(-1) < 0$ and since in this case, $\rho \neq 1$, then we have the following two distinct cases:

- Case (1): $\rho > 1$, then by noting that $\log\rho > 0$ and $\rho^{L+1} - 1 > 0$ for $L > -1$, $\frac{d}{dL}\Delta(L) > 0$ for $L > -1$.
- Case (2): $\rho < 1$, then by noting that $\log\rho < 0$ and $\rho^{L+1} - 1 < 0$ for $L > -1$, $\frac{d}{dL}\Delta(L) > 0$ for $L > -1$.

This proves that $\Delta(L)$ is increasing in L for $L > -1$. Therefore, there exists a unique value $\check{L}$ of L such that $\Delta(\check{L}) = 0$. Thus $E\left[W_{[L,0]}\right]$ decreases with L for $L < \check{L}$ and increases with L for $L > \check{L}$. Note that $[i]$ denotes the largest integer that is not greater than i. Set $\hat{L} = [\check{L}]$ for $\check{L} = [\check{L}]$. For $\check{L} > [\check{L}]$, set $\hat{L} = [\check{L}]$, if $E\left[W_{[\check{L},0]}\right] \leq E\left[W_{[\check{L}+1,0]}\right]$, and $\hat{L} = [\check{L}] + 1$, otherwise. Then, $E\left[W_{[L,0]}\right]$ decreases with L for $L \leq \hat{L}$ and $E\left[W_{[L,0]}\right]$ increases with L for $L > \hat{L}$.

By this proposition, we proved the existence and uniqueness of the optimal solution of the other threshold parameter L. That is, we need to choose only the proper value of L with q fixed to be 0 in finding the set of parameter values of the threshold rule that gives the minimum value for the overall system mean response time.

From the above two propositions, we easily see that, given λ, μ, and θ with $q = 0$, the following algorithm gives the minimum value of the other threshold parameter L that minimizes the overall system mean response time of a job arriving at the MF-PC network model.

Starting from $L = 0$, while $E\left[W_{[L,0]}\right] \geq E\left[W_{[L+1,0]}\right]$, increase L by 1, and otherwise stop. Then the $[L, 0]$ threshold policy brings the minimum value of the overall system mean response time $E\left[W_{[L,0]}\right]$.

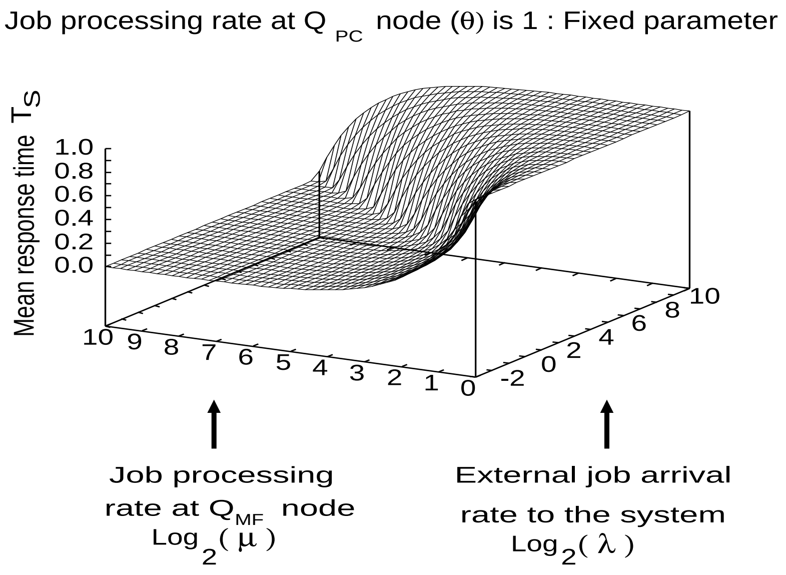

Figure 3.3: The overall system mean response time T_S by the SOOLBP for each combination of the values of λ and μ

3.4 Results and Discussion

Through a number of numerical examples, we estimate the overall system mean response time of the MF-PC network model, using a SOOLBP and a DOOLBP, for each combination of the values of job arrival rate λ to the system, job processing rate μ at the Q_{MF} node, and job processing rate θ at the Q_{PC} node. Since we have only three system parameters λ, μ and θ, we scale down θ to 1 and thus we have only two independent parameters. We denote by T_D and T_S, respectively, the overall system mean response times of the DOOLBP and

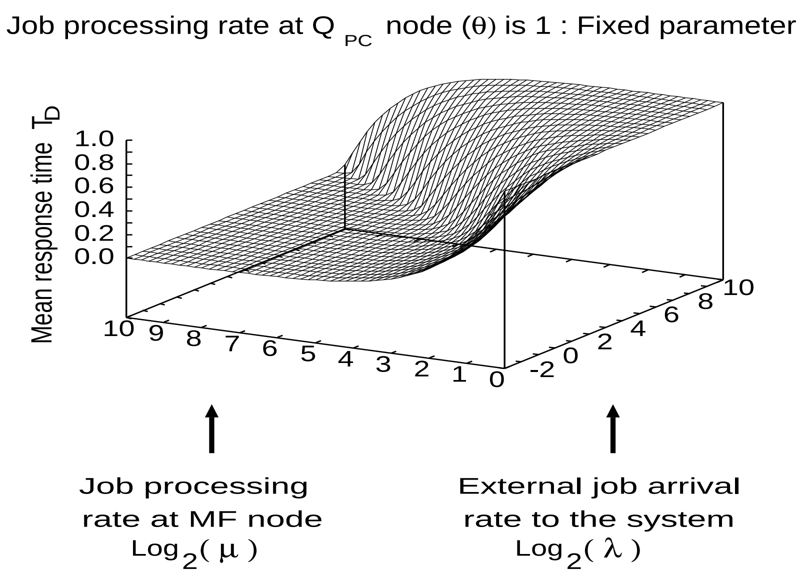

Figure 3.4: The overall system mean response time T_D by the DOOLBP for each combination of the values of λ and μ

SOOLBP.

Figures 3.3 and 3.4 show the overall mean response time of the system by the SOOLBP and DOOLBP, respectively, for various combinations of the values of the system parameters λ and μ. From these two figures, we can see that the overall system mean response time offered by the two considered load balancing policies is in between 0 and 1. This is because the Q_{PC} node offers a fixed expected service time θ^{-1} and we scaled down θ to 1. Also, form these two figures, it is easy to note that the overall system mean response time obtained by the DOOLBP is better than that of the static one specially when the arrival rate λ to the system approaches the processing rate μ of the Q_{MF} node. To estimate how much it is better, we define the improvement ratio in the overall system mean response time to be the ratio of the overall system mean response time of the DOOLBP over that of the SOOLBP as $\frac{T_S - T_D}{T_S}$.

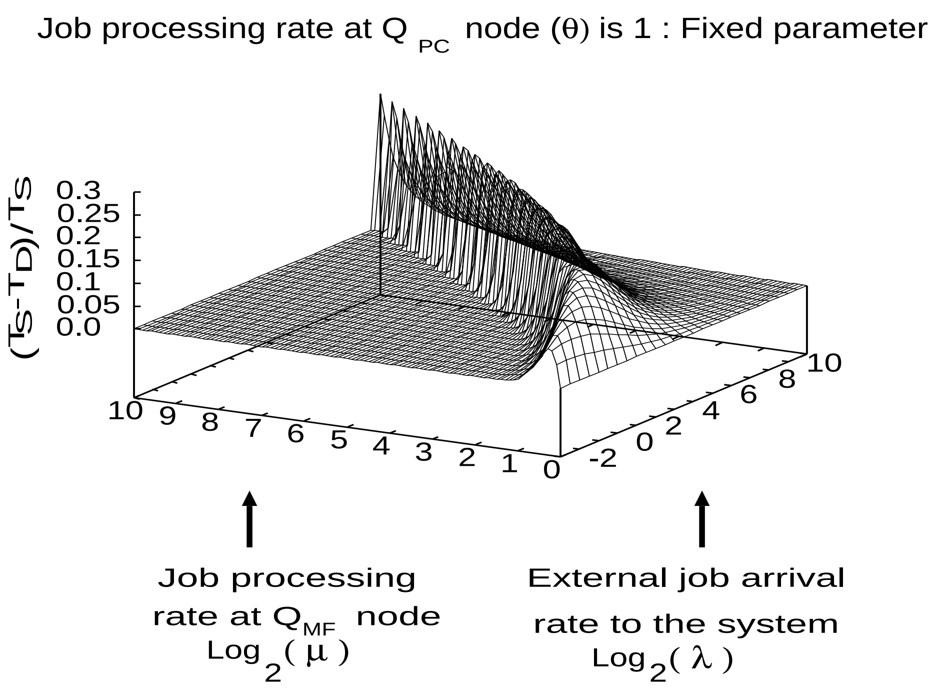

Figure 3.5: The improvement ratio in the overall system mean response time by the DOOLBP over the SOOLBP for each combination of the values of λ and μ

Figure 3.5 shows the improvement ratio in the overall system mean response time with respect to λ and μ. Figure 3.6 shows, for each given value of λ, the maximum improvement ratio in the overall system mean response time with respect to μ. The results naturally confirmed our forecast that the DOOLBP is more effective than the static one. From the two Figures 3.5 and 3.6, we can see that, in the model examined, the improvement ratio in the overall system mean response time by the DOOLBP over the SOOLBP is at most about 30% in the range of parameter values examined while overhead due to the two policies are not taken into account. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. Figure 3.7 shows the corresponding value of μ that gives the maximum improvement ratio in the overall system

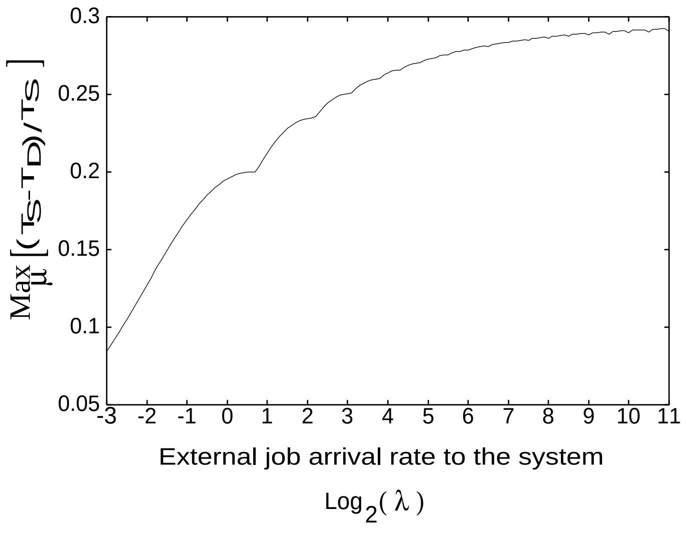

Figure 3.6: The maximum improvement ratio in the overall system mean response time (with respect to μ) by the DOOLBP over the SOOLBP for each value of λ

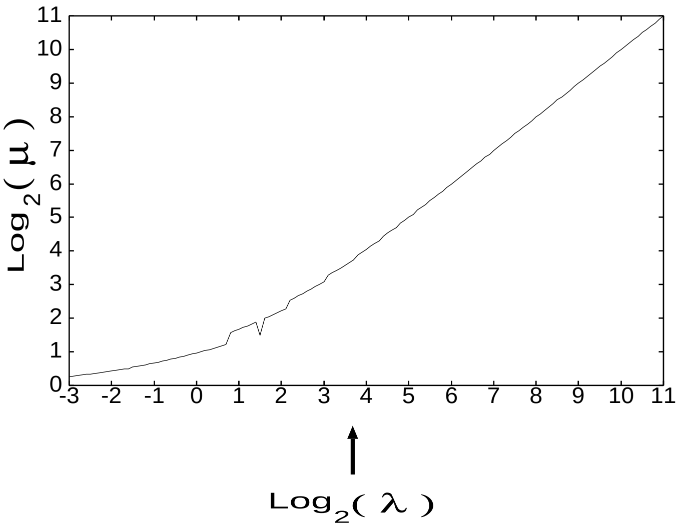

Figure 3.7: The value of μ that gives the maximum improvement ratio in the overall system mean response time by the DOOLBP over the SOOLBP for each value of λ

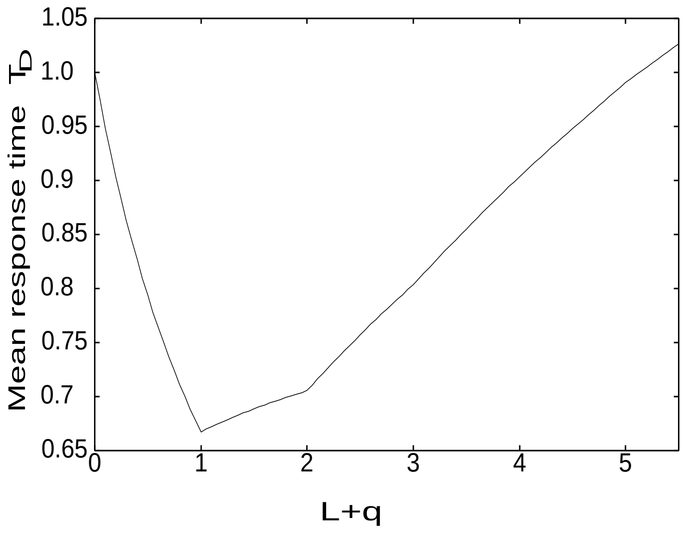

Figure 3.8: The overall system mean job response time by the DOOLBP for each combination of L and q for the case of $\lambda = 1.4142135$ and $\mu = 2.2028464$

mean response time for each given value of λ. From this figure, we see that the maximum improvement ratio in the overall system mean response time is achieved for the cases where $\lambda \sim \mu$ for rather large values of both.

Another remarkable observation is that if the $[L, q]$ threshold rule is used as the DOOLBP, the minimum value of the overall system mean response time is achieved by an $[L, 0]$ threshold rule, that is, the overall system mean response time can be minimized only by suitably selecting the threshold parameter L and the other threshold parameter q is not effective. Since L is an integer and q whose region is $[0, 1)$ (note that $[L, 1]$ is identical to $[L + 1, 0]$), superficially it might look that the DOOLBP (i.e., $[L, q]$ threshold rule) has a continuous parameter $L + q$ to control. The DOOLBP, however, has only the discrete parameter L as the effective parameter to control (see, e.g., Figure 3.8) whereas the SOOLBP has a continuous parameter β_{MF} to control. The two Figures, 3.5 and 3.6, show seemingly peculiar behaviors concerning the improvement ratio in the overall system mean response time as the values of system parameters change. This peculiarity is thought to come from the contrast between the continuity in the control variable β_{MF} for the SOOLBP and the discreteness in the threshold parameter L for the DOOLBP.

3.5 Conclusion

We have studied two optimal load balancing policies. One is a static overall optimal load balancing policy and the other is a dynamic overall optimal load balancing policy, for a distributed computer system consisting of a single-server central node (Q_{MF}) and an infinite-server satellite node (Q_{PC}) connected by a communication network. The aim of both policies is to minimize the overall system mean response time. By numerical examination, we have estimated the difference in the effects on the overall system mean response time between a dynamic overall optimal load balancing policy using the $[L, q]$ threshold rule and a static overall optimal load balancing policy. The results show that, in the model examined, the dynamic overall optimal load balancing policy outperforms the static one in the overall system mean response time, at most about 30% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. Another remarkable result is that, the minimum value of the overall system mean response time is achieved by the dynamic overall optimal load balancing policy ($[L, q]$ threshold rule) with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. That is, we need to choose only the proper value of L with q fixed to be 0 in finding the set of parameter values of the threshold rule that gives the minimum value for the overall system mean job response time.

Chapter 4

A Comparative Study of Static and Dynamic Individually Optimal Load Balancing Policies in a Mainframe – Personal Computer Network Model

4.1 Introduction

The growth of online Internet services during the past decade has increased the demand for scalable and dependable distributed computing systems. These systems face high quality-of-service requirements and concurrently serve many clients that transmit a large, often bursty, number of requests. A distributed computer system is considered to be a collection of autonomous computers (nodes) located at possibly different sites and connected by a communication network. Through the communication network, resources of the system can be shared by users at different locations. Performance enhancement is one of the most important issues in distributed systems. The performance of a distributed computer system can often be improved to an acceptable level simply by redistributing the load among the nodes. The problem of load redistribution in distributed systems is called *load balancing*

[93]. Load balancing for distributed systems has been an active research area for many years and hence as a result, a number of load balancing policies have been proposed to improve the performance of distributed/parallel systems (e.g., to minimize the mean response time of a job, to maximize the processing capacity of the system) by efficiently utilizing the processing power of the entire system. Although a communication delay is incurred in transferring a job from one node to another, the performance of a distributed computer system can generally be improved by an effective load balancing policy [51, 52, 59, 86, 92]. Load balancing policies may be either static or dynamic. For more information about static and dynamic load balancing policies, see section 2.1 in chapter 2.

Traditional computer networks were designed and operated with the overall (system-wide) optimization in mind. Accordingly, the actions of the network users were determined so as to optimize the overall network performance. Consequently, users would often find themselves sacrificing some of their own performance for the sake of the entire network. Recently, it has been recognized that the overall optimization may be an impractical paradigm for the control of the modern (high speed and large-scale) networking configurations. Indeed, control decisions in large-scale networks are often made by each user independently, according to its own individual performance objectives. Such networks are henceforth called *non-cooperative*. The *individual optimum* and the *class optimum* are considered to be two different ways to model the decision making in non-cooperative networks. The most common example of a non-cooperative network is the Internet. In the current Transmission Control Protocol, each user adjusts its transmission window the maximum number of unacknowledged packets that the user can have circulating in the network independently, based on some feedback information about the level of congestion in the network (detected as packet loss). Moreover, the Internet Protocol (both IPv4 and the current IPv6 Specification), for example, provides the option of source routing that enables the user to determine the path(s) its flow follows from source to destination [5, 6, 7, 91]. Also, this kind of non-cooperative resource sharing problems can be observed at higher network layers as well, such as the use of mirror database sites (FTP sites on the Internet providing a good example), World-Wide Web servers and similar systems.

In this chapter, we propose two load balancing policies. One is a static individually optimal load balancing policy (SIOLBP) and the other is a dynamic individually optimal load balancing policy (DIOLBP). In these policies, every job strives to optimize (minimize) its own mean response time independently of the other jobs. In this optimized situation, each job cannot expect any further benefit by changing its own decision. It is also assumed that the decision of a single job has a negligible impact on the performance of other jobs. According to the individually optimal policy, jobs are scheduled so that every job may feel that its own expected response time is minimized if it knows the expected node delay at each node. In other words, when the individually optimal policy is realized, the expected response time of a job cannot be improved further when the scheduling decisions for other jobs are fixed, and the system reaches an equilibrium. It appears that this policy is closely related to a completely decentralized scheme in that each job itself determines on the basis of the information of the mean node delay which node should process it. The performance of these two policies is compared in a distributed computer system where truly optimal solutions of the SIOLBP and the DIOLBP have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, we do not take account of the difference in the overheads due to the two load balancing policies. We focus on two main issues. The first issue is to examine to what extent the DIOLBP policy outperforms the static one by an exhaustive numerical investigation on a model for which both policies are analytically studied. The second issue concerns the examination of the job flow traffic in the proposed system model under the two load balancing policies.

While there have been some studies of performance comparison between dynamic and static load balancing policies in more sophisticated models where overheads are considered [41, 97, 98], the truly optimal dynamic policy is not accurately obtained in contrast to the model considered here.

The results obtained here show that, in the model examined, the DIOLBP outperforms the static one in the overall mean response time, at most about 48% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model under the SIOLBP and the DIOLBP. We found that, there is a difference between the ratio that a job arriving at the system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase.

4.2 Model Description and Assumptions

We consider a distributed computer system model as shown in Figure 4.1. The model consists of two types of service facilities, a Mainframe node Q_{MF} and an unlimited number of Personal Computer nodes Q_{PC}, both of which are connected by a communication network. We call this system model the *MF-PC network model*. This model is absolutely identical to the model which is considered throughout Chapter 3. We assume that the expected communication delay between the Q_{MF} node and the Q_{PC} node is negligible. Jobs arrive at the

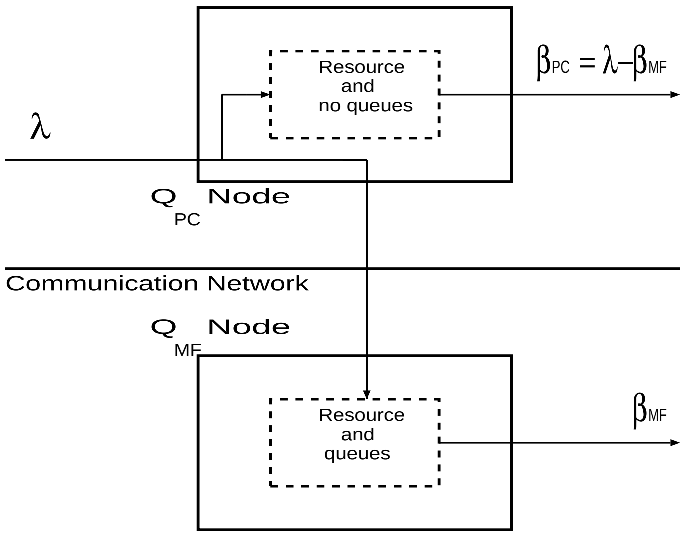

Figure 4.1: A model of an MF-PC network system

system according to a time-invariant Poisson process, i.e. inter-arrival times of jobs are independent, identically and exponentially distributed with mean $1/\lambda$. Simultaneous arrivals are excluded. A job arriving at the system may be processed either by the Q_{MF} node or by the Q_{PC} node according to load balancing policies. We assume that the service rate at Q_{MF} node is μ and that its service discipline is processor sharing whereby the service rate for each job equals $v(n) = \mu/n$ when the number of jobs in the Q_{MF} node is n. The Q_{PC} node offers a fixed expected service time θ^{-1}. We assume that at both Q_{MF} and Q_{PC}, service times are independent, identically and exponentially distributed.

The model we consider here is analytically studied in [25] and is motivated in part by the following scenario.

Potential computer users, each requiring the use of a computer to execute a given job, arrive sequentially at a computer facility. Each user upon arrival, may choose between the following two options: either connect to a central mainframe node Q_{MF}, which is normally serving many users in parallel; or use a personal computer node Q_{PC}. Each user (job) is interested in minimizing his own response time individually (i.e., independently of the other users in the system). For more information about the real applications of the model considered here, see section 3.1 in chapter 3.

4.3 Two Optimal Load Balancing Policies

In the following two subsections, we present static and dynamic individually optimal load balancing policies and their solutions.

4.3.1 Static Individually Optimal Load Balancing Policy (SIOLBP)

In this policy, the decision of transferring a job from one node to another does not depend on the state of the system, and hence is *static* in nature. Also, we assume that a job transferred from one node to another receives its service there, and is not further transferred. According to the individually optimal policy, jobs are scheduled so that every job may feel that its own expected response time is minimum if it knows the expected node delay at each node. In other words, when the individually optimal policy is realized, the expected response time of a job cannot be improved further when the scheduling decisions for other jobs are fixed, and the system reaches an equilibrium [41, 97, 98]. It appears that this policy is closely related to a completely decentralized scheme in that each job itself determines on the basis of the information of the mean node delay which node should process it.

We use the following notation:

- β_{MF}: Job processing rate (load) at the Q_{MF} node.
- β_{PC}: Job processing rate (load) at the Q_{PC} node.

- $F_{MF}(\beta_{MF})$: Expected delay of a job processed at the Q_{MF} node.

With the considered model we have:

$$F_{MF}(\beta_{MF}) = \begin{cases} \dfrac{1}{\mu - \beta_{MF}} & if\, \beta_{MF} < \mu, \\ \infty & \text{otherwise.} \end{cases}$$

The static individually optimal load balancing policy is formulated as the problem of minimizing the mean response time of each job which is expressed as

$$D(\beta_{MF}) = min\left\{F_{MF}(\beta_{MF}), \theta^{-1}\right\} \tag{4.1}$$

with respect to β_{MF} such that $0 \le \beta_{MF} \le \lambda$.

Definition 4.3.1 *β_{MF} is said to satisfy the equilibrium conditions for the static individually optimal policy if the following relations hold:*

$$F_{MF}(\beta_{MF}) > \theta^{-1}, \quad \beta_{MF} = 0, \tag{4.2}$$

$$F_{MF}(\beta_{MF}) < \theta^{-1}, \quad \beta_{MF} = \lambda, \tag{4.3}$$

$$F_{MF}(\beta_{MF}) = \theta^{-1}, \quad 0 \le \beta_{MF} \le \lambda, \tag{4.4}$$

subject to the total follow constraint

$$\beta_{MF} + \beta_{PC} = \lambda. \tag{4.5}$$

4.3.2 Dynamic Individually Optimal Load Balancing Policy (DIOLBP)

We assume that the users are self-optimizing, so that each wishes to minimize his own response time. By this policy, each arriving job may observe the current load in the Q_{MF} node, and then choose whether to join the shared mainframe or to remain at the Q_{PC} node.

A class of threshold load balancing policies have been shown to be useful when jobs are completely independent and consists of single threads of control. This situation is fairly common in networks of workstations. Such threshold policies contain control parameters (e.g. threshold values and transfer probability for every host), that require fine-tuning in order to yield optimal or near optimal performance. For more information about the use of the threshold load balancing policies see [25, 40, 59, 82, 87, 92, 94].

We use the $[L, q]$ threshold rule as a dynamic individually optimal load balancing policy. In this rule, an arriving job will go to the Q_{MF} node with the probability of, respectively, 0, q, and 1, if the arriving job finds that the Q_{MF} node has, more than, equal to, and less than, L jobs.

Given the system parameters λ, μ, and θ, we use the algorithm of Altman and Shimkin [25] to compute the optimal values L^* and q^* of the control parameters L and q that satisfy the equilibrium in between the two system facilities. This algorithm can be summarized as follows:

Equilibrium Threshold Algorithm

The equilibrium threshold $[L^*, q^*]$ is determined by the following procedure.

$L^* = min\left[L \geq 0 : V(L, [L, 1]^\infty) > \theta^{-1}\right]$

$If \;\; V(L^*, [L^*, 0]^\infty) \geq \theta^{-1}$, then the equilibrium threshold is $[L^*, 0]$.

$If \;\; V(L^*, [L^*, 0]^\infty) < \theta^{-1}$, then the equilibrium threshold is $[L^*, q^*]$, where $0 < q^* < 1$ is the unique solution of

$$V(L^*, [L^*, q^*]^\infty) = \theta^{-1}, \tag{4.6}$$

where:

- $V(n,[L,q]^{\infty})$ is the expected service time of an arriving job if it joins the Q_{MF} node at queue length n $(0 \leq n \leq L)$ while all subsequent jobs are using the threshold rule $[L,q]$.

First, we determine the service times $V(L,[L,q]^{\infty})$ and then, we obtain the solution q^* of equation 4.6 as follows:

Fix $[L,q]$ and define $V(n) := V(n,[L,q]^{\infty})$. Then $V(n)$, $0 \leq n \leq L$, is the solution of the following set of $L+1$ linear equations:

$$\begin{aligned}
V(n) &= \frac{1}{\alpha} + \frac{\mu n}{\alpha(n+1)}V(n-1) + \frac{\lambda}{\alpha}V(n+1), \qquad 0 \leq n \leq L-2, &(4.7)\\
V(L-1) &= \frac{1}{\alpha} + \frac{\mu(L-1)}{\alpha L}V(L-2) + \frac{q\lambda}{\alpha}V(L) + \frac{(1-q)\lambda}{\alpha}V(L-1), &(4.8)\\
V(L) &= \frac{1}{\mu} + \frac{L}{L+1}V(L-1), &(4.9)
\end{aligned}$$

where $\alpha = \lambda + \mu$. These equations follow from the memoryless property of the system, which implies that $V(n)$ equals the expected remaining service time of any user present at queue length $n+1$. Thus, $V(n)$ equals the expected time till the next transition (α^{-1} in the first equations), plus the expected remaining service time after that transition. These equations can obviously be solved numerically for each given $[L,q]$; however, in order to obtain the optimal threshold in closed form we use a more explicit solution as follows. By 4.7, $V(n)$ can be expressed as

$$V(n) = a(n)V(0) + b(n), \qquad 0 \leq n < L, \qquad (4.10)$$

where the coefficients $a(n)$ and $b(n)$ are obtained recursively by substituting 4.10 into 4.7, which yields

$$a(n+1) = \frac{1}{\lambda}\left[(\mu+\lambda)a(n) - \mu\frac{n}{n+1}a(n-1)\right], \qquad n \geq 1, \tag{4.11}$$

$$b(n+1) = \frac{1}{\lambda}\left[(\mu+\lambda)b(n) - \mu\frac{n}{n+1}a(n-1) - 1\right], \quad n \geq 1, \tag{4.12}$$

with initial values $a(0) = 1$, $a(1) = (\alpha/\lambda)$, $b(0) = 0$, $b(1) = -\lambda^{-1}$. Note that these coefficients do not depend on L and q. So, $V(0, [L, q])$ is obtained by substituting $V(L)$ from 4.9 into 4.8 and then substituting $V(L-2)$ and $V(L-1)$ from 4.10 as follows:

$$V(0) = \frac{\mu^{-1} - \frac{1}{L+1}b(L-1) - q^{-1}(b(L) - b(L-1))}{\frac{1}{L+1}a(L-1) + q^{-1}(a(L) - a(L-1)} \tag{4.13}$$

$V(n)$ can now be obtained from 4.10 to 4.13. Hence, the equilibrium threshold can be calculated. First, we compute L^* as follows:

$L^* = min\left[L \geq 0 : V(L, [L, 1]^\infty) > \theta^{-1}\right]$. If $V(L^*, [L^*, 0]^\infty) \geq \theta^{-1}$, then $q^* = 0$. Otherwise, $0 < q^* < 1$ is computed as the unique solution of 4.6. Using 4.9, 4.10, and 4.13, $V(L^*, [L^*, q]^\infty)$ can be expressed as a function of q whose solution is

$$q^* = C^{-1}\left[\frac{a(L^*) - a(L^*-1)}{a(L^*)}\left(\theta^{-1} - \mu^{-1} - \frac{L^*}{L^*+1}b(L^*-1)\right) + \frac{L^*}{L^*+1}\left(b(L^*) - b(L^*-1)\right)\right], \tag{4.14}$$

where $C \triangleq \mu^{-1} - \theta^{-1}/(L^*+1)$.

For more details about the computation of $V(n, [L, q]^\infty)$ and the optimality of this algorithm see [25].

After computing the optimal values L^* and q^* of the control parameters L and q that satisfy the equilibrium in between the two system facilities, the mean response time of a job arriving at the system with threshold $[L^*, q^*]$ is computed using equation 3.4.

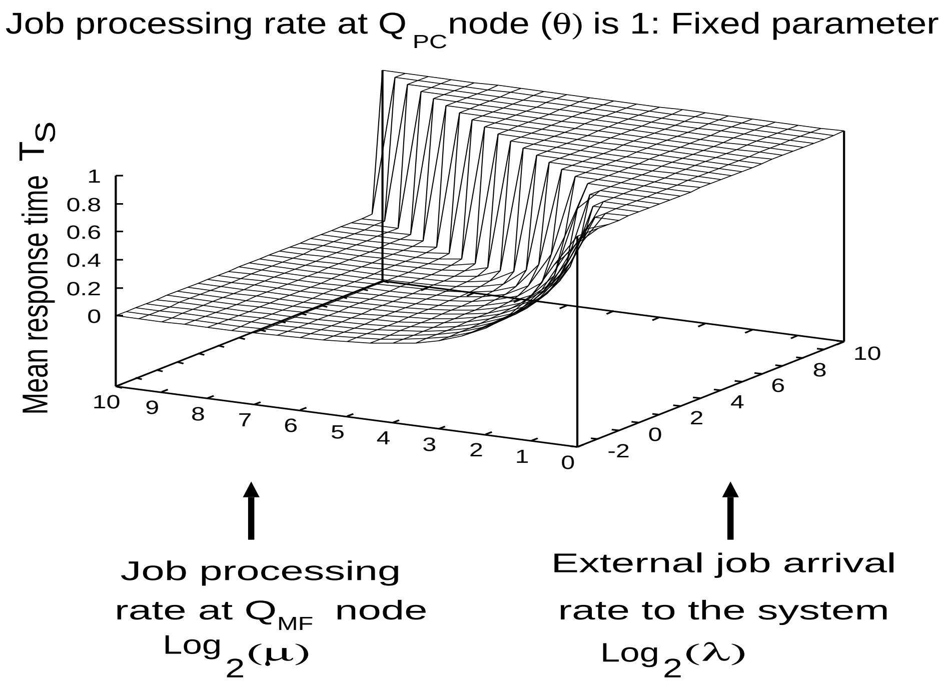

Figure 4.2: Mean job response time T_S by the SIOLBP for each combination of the values of λ and μ

4.4 Results and Discussion

Through a number of numerical examples, we estimate the mean response time of a job arriving at the MF-PC network model for each combination of the values of job arrival rate λ to the system, job processing rate μ at the Q_{MF} node, and job processing rate θ at the Q_{PC} node. Since we have only three system parameters λ, μ and θ. Without a loss of generality, we scaled down θ to 1 and thus we have only two independent parameters. We denote by T_S and T_D, respectively, the mean job response times with the SIOLBP and DIOLBP.

Figures 4.2 and 4.3 show that the mean response time of a job arriving at the system by the SIOLBP and DIOLBP, respectively, for various combinations of the values of λ and μ. From these two figures, we can see that the mean job response time offered by the two

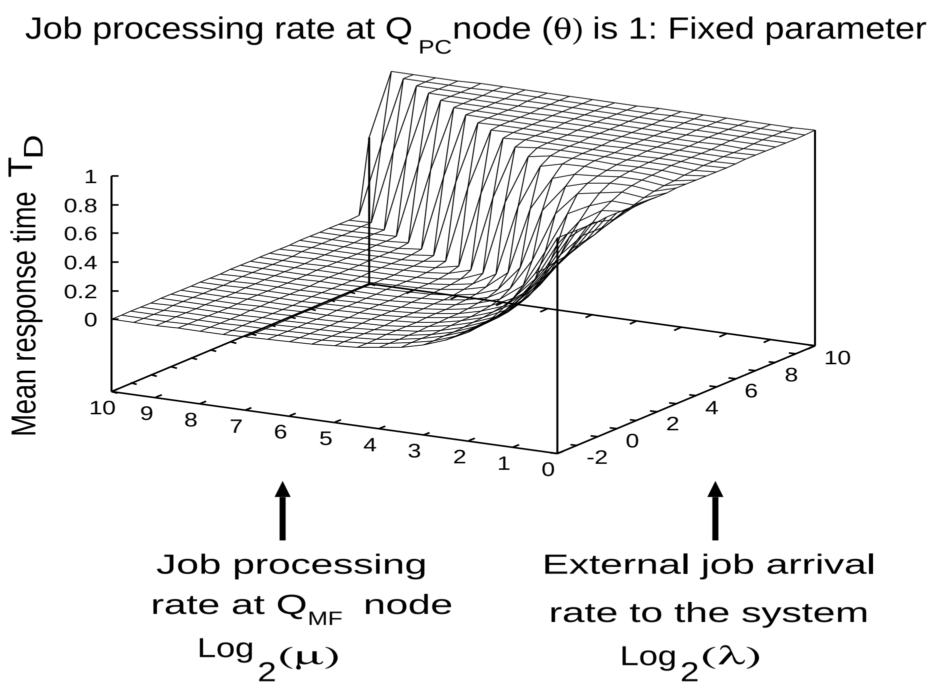

Figure 4.3: Mean job response time T_D by the DIOLBP for each combination of the values of λ and μ

considered load balancing policies is in between 0 and 1. This is because the Q_{PC} node offers a fixed service time θ^{-1} and we scale down θ to 1. Also, form these two figures, it is easy to note that the mean job response time obtained by the DIOLBP is better than that of the static one specially when the arrival rate λ to the system approaches the processing rate μ of the Q_{MF} node. To estimate how much it is better, we define the improvement ratio in the mean job response time to be the ratio of the mean job response time of the DIOLBP over that of the SIOLBP as $\frac{T_S - T_D}{T_S}$. Figure 4.4 shows the improvement ratio in the mean job response time with respect to λ and μ. From that figure, we can see that the mean response time is improved by the DIOLBP over that of the SIOLBP at most about 48% in the range of parameter values examined while the overheads due to the two policies are not

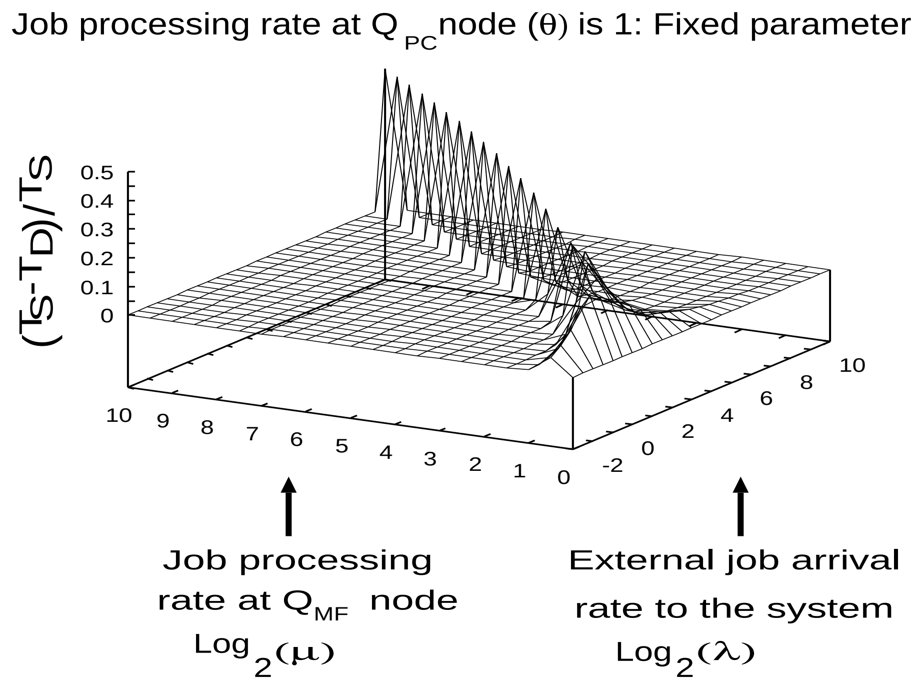

Figure 4.4: The improvement ratio in the mean job response time by DIOLBP over the SIOLBP for each combination of the values of λ and μ

taken into account. The maximum improvement ratio is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. The results naturally confirmed our forecast that the DIOLBP is more effective than the static one.

Since the Q_{PC} node offers a fixed expected service time (θ^{-1}). We computed the Q_{MF} node mean job response time by the SIOLBP (MFT_S) and the DIOLBP (MFT_D) to see the effect of these two policies on it. By the SIOLBP, the Q_{MF} node mean job response time is the same as the mean response time of a job arriving at the system (see Figure 4.2), this is because when the two system service facilities are used, the Q_{MF} node response time equals that of the Q_{PC} node (θ^{-1}). The Q_{MF} node mean job response time by the DIOLBP is shown in Figure 4.5. From that figure, it is noticed that the Q_{MF} node mean job response

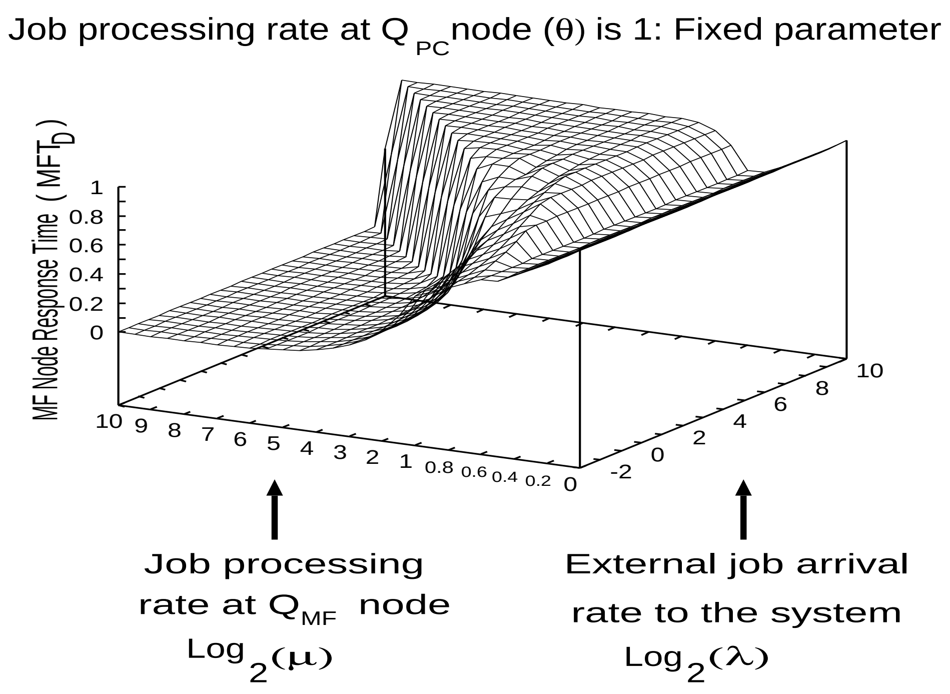

Figure 4.5: Mean job response time of the Q_{MF} node by the DIOLBP for each combination of the values of λ and μ

time decreases as μ decreases from 2θ to $\sqrt{2}\theta$ which is unusual and then starts to increase again as μ decreases from $\sqrt{2}\theta$ to θ. This is because, when $\mu = 2\theta$, the optimal values of the threshold parameters L and q are $L = 1$ and $q = 1$, which is equivalent to $L = 2$ and $q = 0$ and as $\mu \to \sqrt{2}\theta$, the value of q gradually decreases until it becomes zero at $\mu = \sqrt{2}\theta$. This explains why the Q_{MF} node mean job response time decreases as μ decreases from 2θ to $\sqrt{2}\theta$. When μ decreases from $\sqrt{2}\theta$ to θ the optimal values of the threshold parameters L and q are $L = 1$ and $q = 0$ (whatever the values of λ and μ). Which means that only on job could be in the Q_{MF} node and hence, it's expected response time is μ^{-1}. So that the Q_{MF} node mean job response time increases as μ decreases from $\sqrt{2}\theta$ to θ. This directly explains the peculiarity obtained in Figure 4.6 which presents the improvement ratio $(\frac{MFT_S - MFT_D}{MFT_S})$

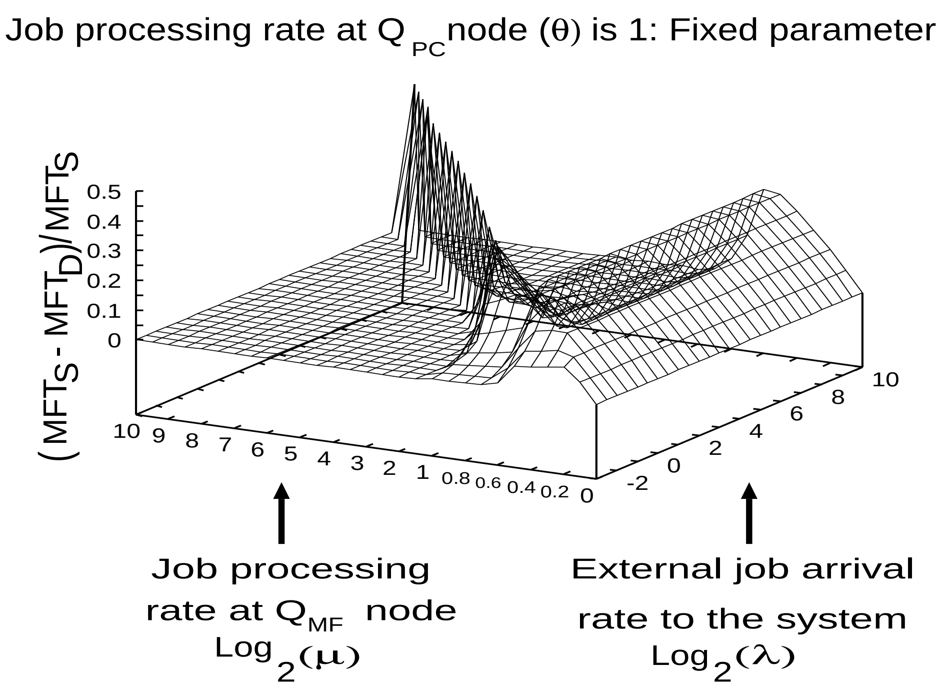

Figure 4.6: The improvement ratio in the Q_{MF} node mean job response time by the DIOLBP over the SIOLBP for each combination of the values of λ and μ

in the Q_{MF} node mean job response time.

To examine the job flow traffic in the proposed system model under the two load balancing policies, we compute the ratio that an arriving job at the system goes to Q_{MF} node under the SIOLBP and the DIOLBP, denoted by R_s and R_d, respectively. Figure 4.7 presents $|R_s - R_d|$, since in this figure, it is not easy to see exactly what going on and $|R_s - Rd| < 0.21$, we compute $\sqrt{|R_s - R_d|}$ which is a magnification for the difference between the two ratios for various combinations of the values of λ and μ as shown in Figure 4.8. From that figure one can notice that almost there is no difference between R_s and R_d when λ is significantly smaller than μ, the maximum difference between R_s and R_d is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase.

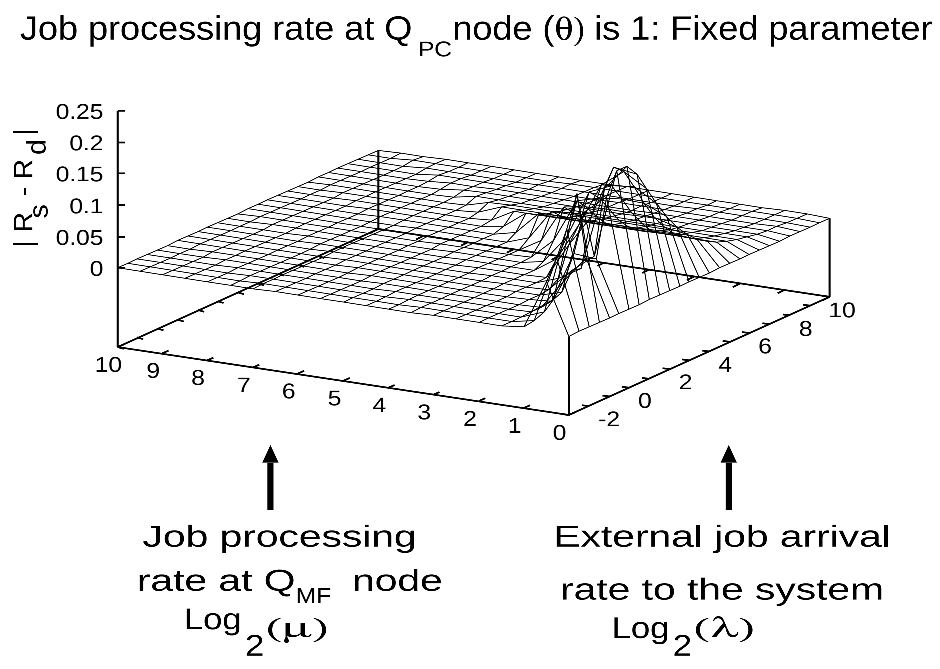

Figure 4.7: The absolute value of the difference between the ratio that an arriving job at the system goes to the Q_{MF} node under the SIOLBP and the DIOLBP (i.e., $|R_s - R_d|$) for each combination of the values of λ and μ

As exemplified by Figure 4.9, through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have a effect in satisfying the equilibrium in between the two system facilities. And, also as exemplified by Figure 4.9, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ and it approaches μ/θ.

4.5 Conclusion

We have studied two optimal load balancing policies, static individually and dynamic individually, for a system consisting of a single-server central node (Q_{MF}) and an infinite-server

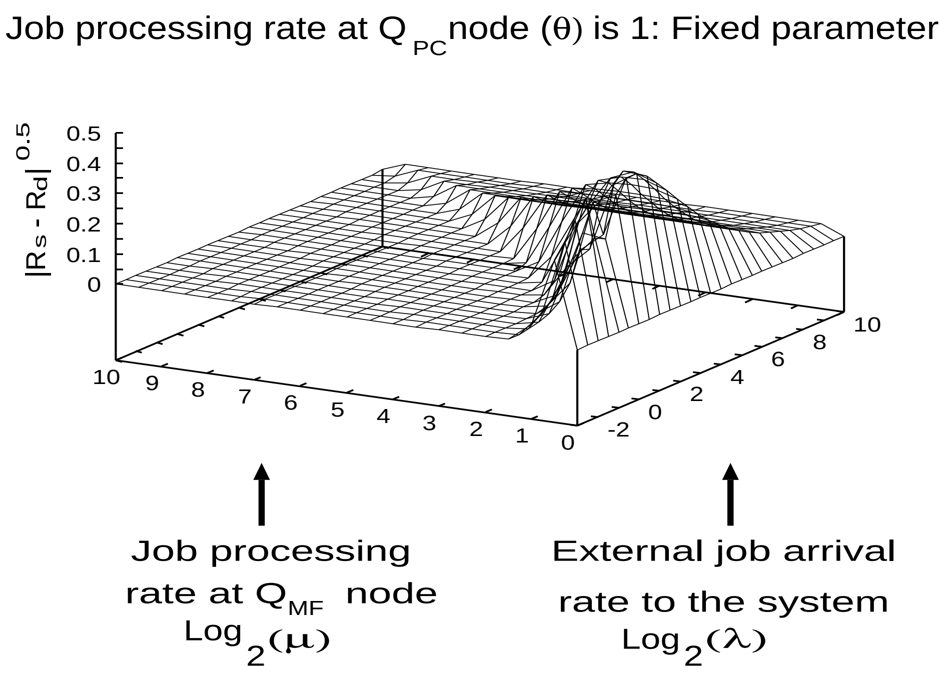

Figure 4.8: The square root of the absolute value of the difference between the ratio that an arriving job at the system goes to the Q_{MF} node under the SIOLBP and the DIOLBP (i.e., $\sqrt{|R_s - R_d|}$) for each combination of the values of λ and μ

satellite node (Q_{PC}) connected by a communication network. By numerical examination, we have estimated the rate of difference in the effects on the mean job response time between a SIOLBP and a DIOLBP using threshold $[L, q]$.

We have observed that the improvement ratio in the mean response time by the DIOLBP over the static one is at most about 48% in the model examined while the overheads due to the policies are not taken into account. The improvement ratio is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model by computing the ratio that an arriving job at the system goes to Q_{MF} node under the SIOLBP and the DIOLBP. The results show that, there is a difference between the ratio that a job arriving at the

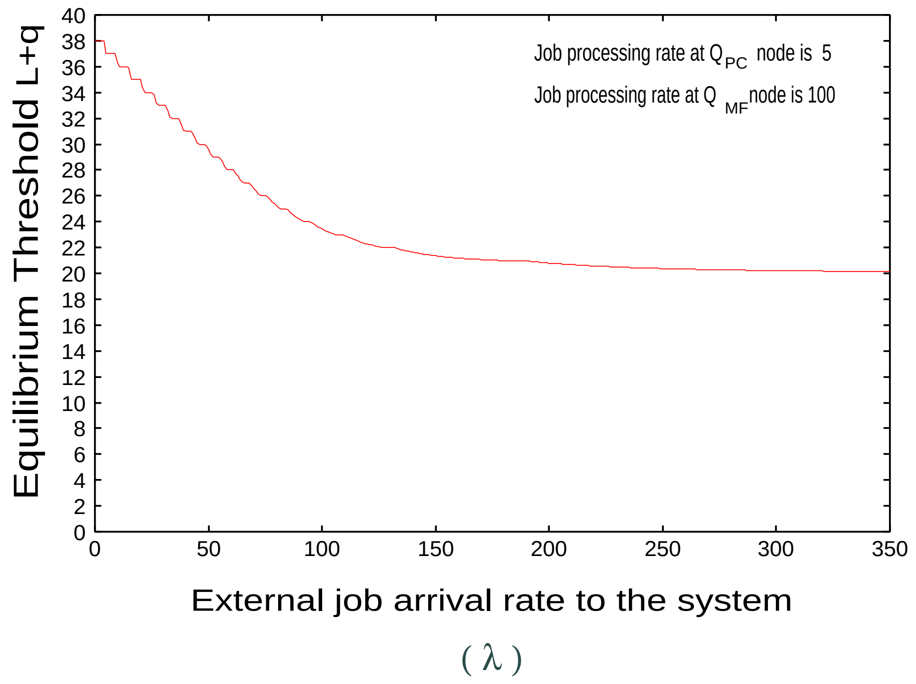

Figure 4.9: L+q as a function of the external job arrival rate λ to the system

system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase.

Through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have a effect in satisfying the equilibrium in between the two system facilities. And also, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ and it approaches μ/θ.

Chapter 5

Numerical Studies on a Paradox for Non-Cooperative Static Load Balancing in Distributed Computer Systems

5.1 Introduction

Tasks that require huge computations and process colossal quantities of data are now numerous and diverse. Such is the case of meteorological or climate prediction, computing the aerodynamic behavior of a new model of aircraft, deciphering the genome of a living organism or detecting the elementary particles produced by an accelerator, to name but a few. These tasks are also becoming increasingly ambitious, and thus more and more demanding in terms of computing power, data flow and memory capacity. How can computer infrastructure meet these continuously growing needs?

The performance of the hardware and software available in each computing center or to each individual user is rising very sharply. This trend is not however sufficient to meet the many challenges that face science, technology and industry. The computing power in hardware doubles every 18 months or so, on the average, whereas storage capacity doubles every 12 months and the performance of network connections doubles every 9 months.

Thus, the performance of computers improves less rapidly than that of networks. Therefore, a potentially revolutionary concept has been developing for six to seven years. The idea is to link geographically distant equipment together, especially via the Internet, to constitute a network that combines the computing power, storage capabilities and so forth of all its users. Each of these users will thus be able to use the sum of available resources in terms of computing power, memory, software and data, put in by all the other users of the network. This is the basic idea behind computing grids. It means that computer resources are simultaneously globalized and dematerialized [67]. The model considered in this chapter could be considered as a basic model for the *GRID* computing infrastructure.

The exponential growth of computer networking, in terms of number of users and components, traffic volume and diversity of service, demands massive upgrades of capacity in existing networks. Traditionally, capacity design methodologies have been developed with a single-class networking paradigm in mind. This approach overlooks the non-cooperative structure of modern (high speed and large-scale) networks and entails, as will be explained in the sequel, the danger of degraded performance when resources are added to a network, a phenomenon known as the *Braess Paradox*. The term non-cooperative is used to characterize networks operated according to a decentralized control paradigm, where control decision are made by each user independently, according to its own individual performance objectives. The *individual optimum* and the *class optimum* are considered to be two different ways to model the decision making in non-cooperative networks. The most common example of a non-cooperative network is the Internet. In the current Transmission Control Protocol, each user adjusts its transmission window the maximum number of unacknowledged packets that the user can have circulating in the network independently, based on some feedback information about the level of congestion in the network (detected as packet loss). Moreover, the Internet Protocol (both IPv4 and the current IPv6 Specification), for example, provides the option of source routing that enables the user to determine the path(s) its flow follows from source to destination [5, 6, 7, 91]. Also, this kind of non-cooperative resource sharing problems can be observed at higher network layers as well, such as the use of mirror database sites (FTP sites on the Internet providing a good example), World-Wide Web servers and similar systems. In fact, the problem extends beyond the realm of networking. Multiprocessor systems that are shared by noncooperative tasks provide yet another field for the potential application of the present study.

Intuitively, we can think that the total processing capacity of a system will increase when the capacity of a part of the system increases and so we expect improvements in performance objectives accordingly in that case. The famous Braess Paradox tells us that this is not always the case; i.e., adding capacity to the system may sometimes lead to the degradation in the benefits of all users in an individual optimum [10, 19, 26, 27, 42, 66, 89].

As it is known that the class optimum converges to the individual optimum as the number of classes becomes large [3], we can expect that, in the class optimum, a similar type of paradox occurs (with large number of classes), i.e., increased capacity of a part of the system may lead to the degradation in the benefits of all classes in a class optimum, whenever it occurs for the individual optimum. Indeed in [5], Korilis found some examples wherein the Braess-like paradox appears in a class optimum where all user classes are identical in the same topology for which the original Braess Paradox (for the individual optimum) was in fact obtained. Furthermore in [6], he also obtained a sufficient condition under which the Braess Paradox should not occur in a more general model that has one source-destination pair and identical user classes.

In [38], Kameda obtained, however, numerical examples where a paradox similar to

Braess's appears in the class optimum but does not occur in the individual optimum in the same environment. These cases look quite strange if we note that such a paradox should never occur in the overall optimum and if we regard the class optimum as an intermediate between the overall optimum and the individual optimum.

In [43], it has been shown that the worst-case degree of the paradox (WCDP) may increase without bound in class optimum where the values of parameters of all classes are identical and also it has been shown that this strange behavior (i.e., the WCDP may increase without bound) does not occur for the overall and individual optimum, in the same setting of the system parameters. To the best of our knowledge, [43] is the first paper that reported such a case where the WCDP can increase without bound. Based on [43] some questions arise like, under what conditions in the class optimum this strange behavior appears? If we slightly change the system parameter setting to represent asymmetric system model, what will happen to this strange behavior? Will it increase (decrease)? If it increases (decreases), will this increase (decrease) be rapid or slow? And finally, what will be the overall tendency of the WCDP? The algorithms used to obtain the optima and the equilibria are based on the algorithms given in [15, 41, 45, 50, 74].

In this chapter, we answer these questions through a number of numerical examples around the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for static load balancing. Each node in the system has, at its disposition, a communication means, which it may use to forward to other nodes an arbitrary portion of its job arrival stream. We considered three different types of communication means (A), (B) and (C). Based on the system parameter setting, three types of symmetries *(overall symmetry, individual symmetry and complete symmetry)* are defined. From the numerical examples, it is observed that in the class optimum, the WCDP is largest (i.e., the worst performance is obtained) in the complete symmetry case when the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

5.2 Model Description and Assumptions

We consider a distributed computer system that consists of m nodes (host computers or processors) connected with a communication means as shown in Figure 5.1. Nodes are numbered $1, 2, ..., m$. Each node consists of a single exponential server with service rate $\mu_i (i = 1, 2, ..., m)$. We classify jobs arriving at node i into class i, $i = 1, 2, ..., m$. Jobs arrive to node i according to a time-invariant Poisson process, with the average external arrival rate ϕ_i, out of which the rate x_{ii} of jobs are processed at node i. The rate x_{ij} of jobs is forwarded upon arrival through the communication means to another node j $(\neq i)$ to be processed there and the results of processing those jobs are returned back through the communication means to node i. We assume further that a transferred job from node i to node j ($\neq i$) receives its service at node j and is not transferred to other nodes *(i.e., each job is forwarded at most once)*. Then, it follows that $\sum_p x_{ip} = \phi_i$, $x_{ij} \geq 0$, $i, j = 1, 2, ..., m$. We denote the vector $(x_{i1}, x_{i2}, \cdots, x_{im})$ by $\mathbf{x}_i$ and the vector $(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m)$ by $\mathbf{x}$. Thus, $\mathbf{x} = (x_{11}, x_{12}, \cdots, x_{1m}, x_{21}, x_{22}, \cdots, x_{2m}, \cdots, x_{mm})$. Denote the set of $\mathbf{x}$'s that satisfy the constrains by $\mathbf{E}$ and the total arrival rate to the system by Φ, hence $\Phi = \sum_p \phi_p$. Each node has one decision maker, also numbered i, $i = 1, 2, \cdots, m$. Within these constrains, a set of values of x_{ij}, $(i, j = 1, 2, \cdots, m)$ are chosen to achieve the optimization. The load on node i is $\sum_q x_{qi}$ and is denoted by β_i. The expected processing (*including queueing*) time of a job that is proceeded at node i, is given by:

$$D_i(\beta_i) = \begin{cases} \dfrac{1}{\mu_i - \beta_i} & if\ \beta_i < \mu_i, \\ \infty & \text{otherwise.} \end{cases}$$

The expected communication (*including queueing*) time of forwarding a job arriving at node i to node j and sending it back after processing from node j to node i, $(i \neq j)$ is denoted by $G_{ij}(\mathbf{x})$. We refer to the length of time between the instant when a job arrives at a node and the instant when it leaves one of the nodes, after all processing and communication, if any, are over as *the response time* for the job. Thus the expected response time of a job that arrives at node i is given by:

$$T_i(\mathbf{x}) = \frac{1}{\phi_i} \sum_k x_{ik} T_{ik}(\mathbf{x}), \tag{5.1}$$

where

$$T_{ii}(\mathbf{x}) = D_i(\beta_i), \tag{5.2}$$

$$T_{ij}(\mathbf{x}) = D_j(\beta_j) + G_{ij}(\mathbf{x}), \text{ for } j \neq i \tag{5.3}$$

Then, the overall expected response time of a job that arrives at the system is given by:

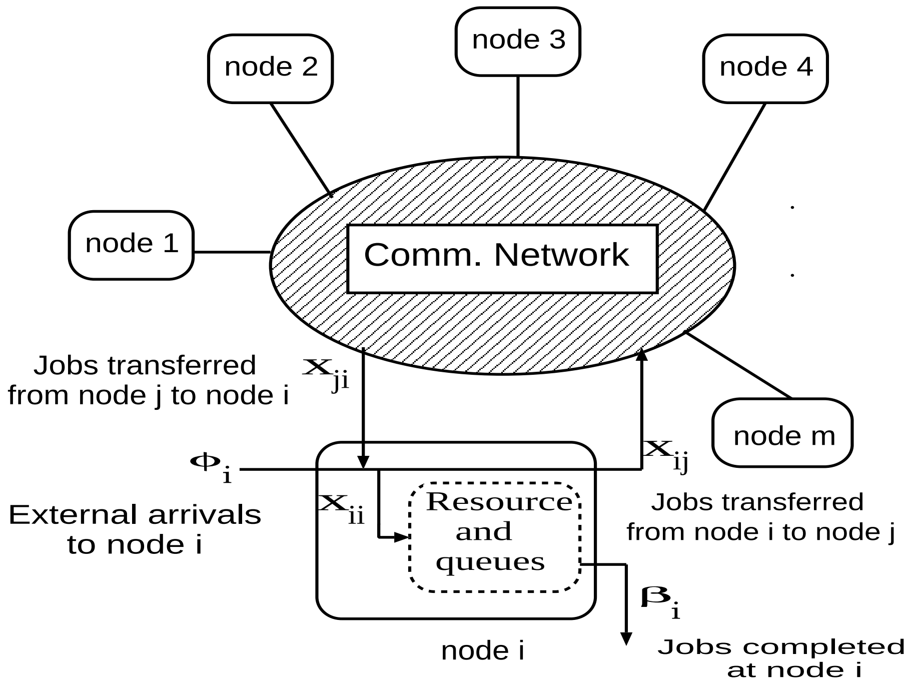

Figure 5.1: A distributed computer system

$$T(\mathbf{x}) = \frac{1}{\Phi} \sum_i \phi_i T_i(\mathbf{x}) \tag{5.4}$$

As we mentioned in chapter 1, section 1.1, in many systems including communication networks and distributed computer systems, we may have several distinct objectives for performance optimization. Among them, we have the following three typical objectives or optima:

1. *The overall optimum* is given by $\bar{\mathbf{x}}$ that satisfies the following,

$$T(\bar{\mathbf{x}}) = \min T(\mathbf{x}) \quad \textit{such that} \quad \mathbf{x} \in \mathbf{E}. \tag{5.5}$$

2. *The individual optimum* is given by $\hat{\mathbf{x}}$ that satisfies the following for all i,

$$T_i(\hat{\mathbf{x}}) = \min_p \{T_{ip}(\hat{\mathbf{x}})\}, \quad \textit{such that} \;\; \hat{\mathbf{x}} \in \mathbf{E}. \tag{5.6}$$

3. *The class optimum (or Nash equilibrium)* is given by $\tilde{\mathbf{x}}$ that satisfies the following for all i,

$$\tilde{T}_i = T_i(\tilde{\mathbf{x}}) = \min_{\mathbf{x}_i} T_i(\tilde{\mathbf{x}}_{-(i)}; \mathbf{x}_i), \;\; \textit{such that} \; (\tilde{\mathbf{x}}_{-(i)}; \mathbf{x}_i) \in \mathbf{E}, \tag{5.7}$$

where $(\tilde{\mathbf{x}}_{-(i)}; \mathbf{x}_i)$ denotes the mm-dimensional vector in which the elements corresponding to $\tilde{\mathbf{x}}_i$ have been replaced respectively by $\mathbf{x}_i$.

In [23, 97] it is shown that the three problems (5.5), (5.6) and (5.7) are equivalent to some variational inequalities. For the existence and uniqueness of those optima the reader is referred to [23, 24]. In [45], it has been shown that no mutual forwarding of jobs occurs in overall and individual optima. Consequently, *no paradox* occurs in overall and individual optima. In this chapter, we consider only the class optimum.

5.3 Communication Means

As to the communication means, we consider the following three types (A), (B) and (C).

(A) It consists of $m(m-1)$ two-way communication lines. The two-way communication line ij is used for forwarding of jobs that arrive at node i to node j ($\neq i$) and for sending back the processed results of these jobs to node i. Each communication line connecting node i to node j consists of s communication channels. Each communication channel is chosen randomly with probability $1/s$ and is modelled by a processor sharing server with service rate $1/t$; *i.e., the mean communication (without queueing) time is* t. Thus, the capacity of each communication channel is $1/t$. We assume that the expected communication (*including queueing*) time of a job arriving at node i and being processed at node j ($\neq i$) is expressed as

$$G_{ij}(\boldsymbol{x}) = \begin{cases} \dfrac{t}{1 - \dfrac{x_{ij}t}{s}} & \text{if } \dfrac{x_{ij}t}{s} < 1, \\ \infty & \text{otherwise.} \end{cases}$$

(B) It consists of a single communication line that is used commonly in forwarding and sending back of jobs that arrive at all nodes in the system. The assumption on the communication line is the same as in type (A) except that there is only one communication line which is used for forwarding and sending back jobs arriving at all nodes in the system. Thus, the expected communication (*including queueing*) time of a job arriving at node i and being processed at node j ($\neq i$) is expressed as

$$G_{ij}(\boldsymbol{x}) = \begin{cases} \dfrac{t}{1 - \dfrac{\lambda t}{s}} & \text{if } \dfrac{\lambda t}{s} < 1, \\ \infty & \text{otherwise,} \end{cases}$$

where $\lambda = \sum_{i=1}^{m} \sum_{k=1,(k\neq i)}^{m} x_{ik}$ is the communication traffic through the line (network traffic).

(C) It consists of a single or multiple communication line that has no queueing delay. Thus, the expected communication time of a job arriving at node i and being processed at node j ($\neq i$) is expressed as

$$G_{ij}(\mathbf{x}) = t.$$

5.4 Worst-Case Degree of the Paradox (WCDP)

For each set of data ϕ_i and μ_i, $i = 1, 2, \cdots, m$, we can find some value t^{∞} (depending upon the set of data) of the mean communication time such that the communication line is not used any more at equilibria if the mean communication time is larger than t^∞. For the set of data ϕ_i and μ_i, $i = 1, 2, \cdots, m$, we increase the communication time from 0 to t^∞. For each t we compute the class optimum (Nash equilibrium).

We focus our attention on the degradation that may occur as a result of increasing the communication capacity. To this aim we say that a *Braess-like paradox* occurs if the following holds:

$$\begin{aligned} &\delta_i(t_1, t_2) > 0 && for\ all\ i \\ &for\ some\ t_1, t_2 && such\ that\ 0 < t_1 < t_2, \end{aligned} \tag{5.8}$$

where $\delta_i(t_1, t_2) = \dfrac{\tilde{T}_i(t_1) - \tilde{T}_i(t_2)}{\tilde{T}_i(t_2)}$ and $\tilde{T}_i(t)$ denotes the mean response time for class i jobs, computed at the unique (Nash) equilibrium, when the mean communication time is t.

For simplicity, we only consider the case where $t_2 = t^\infty$ equivalently, the system has no communication means and we denote $\delta_i(t, t^\infty)$ by $\Delta_i(t)$. Denote $\boldsymbol{\phi}= (\phi_1, \phi_2, \cdots, \phi_m)$ and $\boldsymbol{\mu}= (\mu_1, \mu_2, \cdots, \mu_m)$. Thus, we define the worst-case degree of the paradox ($\Gamma(\boldsymbol{\mu}, \boldsymbol{\phi})$) as follows:

$$\Gamma(\boldsymbol{\mu}, \boldsymbol{\phi}) = \max_t\{\min_i\{\Delta_i(t)\}\}. \tag{5.9}$$

5.5 Types of Symmetries

Based on the system parameter setting, we define the following types of symmetries among nodes of the system.

5.5.1 Overall Symmetry

If the following condition holds

$$\frac{\mu_i}{(\mu_i - \phi_i)^2} = constant, for\ all\ i, \tag{5.10}$$

then according to [41, 45], there is no forwarding of jobs among nodes for any value of the communication channel capacity $1/t$, for the cases (A), (B) and (C) of the communication means, when the system is at the overall optimum. If condition (5.10) holds, in this case, we say that we have an overall symmetry property among nodes.

5.5.2 Individual Symmetry

If the following condition holds

$$\frac{1}{\mu_i - \phi_i} = constant, for\ all\ i, \tag{5.11}$$

it can be proved from definition (5.6) that at the individual optimum there is no forwarding of jobs among nodes for any value of the communication channel capacity $1/t$, for the cases (A), (B) and (C) of the communication means. If condition (5.11) holds, in this case, we say that we have an individual symmetry property among nodes.

5.5.3 Complete Symmetry

If both conditions (5.10) and (5.11) hold or equivalently if $\mu_1 = \mu_2 = \cdots = \mu_m$ and $\phi_1 = \phi_2 = \cdots = \phi_m$, then we say that we have a complete symmetry among nodes. In complete symmetry, according to [45], there is no forwarding of jobs both in the overall and individual optima.

5.5.4 No Symmetry

If the following condition holds

$$\frac{\phi_i}{\mu_i - \phi_i} = constant, \; for\ all\ i, \tag{5.12}$$

in this case, we say that there is no symmetry property among nodes.

5.6 Results and Discussion

We answer the questions raised in section 5.1 through a number of numerical examples around the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for static load balancing. These examples are classified according to the type of symmetries among nodes of the system as follows:

5.6.1 Complete Symmetry Maintained

In Figure 5.2 and table 5.1, with $m = 2$ (*i.e., the system has two nodes*), we show the effect of changing $\mu_i = \mu$ while keeping $\phi_i = \phi = 1$ (*i.e., the complete symmetry property is maintained*) on the WCDP using the communication means of type (C). As shown in Figure 5.2 and table 5.1, by using the communication means of type (C), the WCDP may increase without bound as the arrival rate ϕ approaches the processing rate μ.

Also, in the tables 5.2, 5.3 and 5.4 with $m = 2, 4, 8$ respectively and $s = 1$ (*i.e, every communication line has only one communication channel*), we show the effect of changing $\mu_i = \mu$ while keeping $\phi_i = \phi = 1$ (*i.e., the complete symmetry property is maintained*) and also, we show the effect of increasing the number m of nodes in the system on the

WCDP using the communication means of type (A) when the processing rate μ approaches the arrival rate ϕ. As shown in the tables 5.2, 5.3 and 5.4, by using the communication means of type (A), the WCDP converges to a certain limit as the arrival rate approaches the processing rate and it increases as the number m of nodes in the system increases. It is very important to note that, the results obtained when the communication means of type (B) is used show the same tendency to those of type (A).

Generally, from the previous results, we can say that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used.

$$\frac{\phi_1}{\mu_1 - \phi_1} + \frac{\phi_2}{\mu_2 - \phi_2} = 10000. \tag{5.13}$$

Note: *under condition (5.13), the obtained values of ϕ_i are very close to the corresponding values of μ_i.*

Table 5.1: The effect of changing μ while keeping $\phi = 1$ on the WCDP in complete symmetry with $m = 2$ using the communication means of types (C), when the processing rate approaches the arrival rate

μ	1.01	1.001	1.0001	1.00001
Γ(%)	1250.0	12500.0	125000.0	1249999.998

To see what will happen if the system parameter setting gradually departs the complete symmetry property, using the communication means of types (A) and (C), we examined a distributed computer system that consists of two servers (*i.e., m=2)* and $s = 1$. We show typical numerical examples, by changing the system parameters values for each of the three

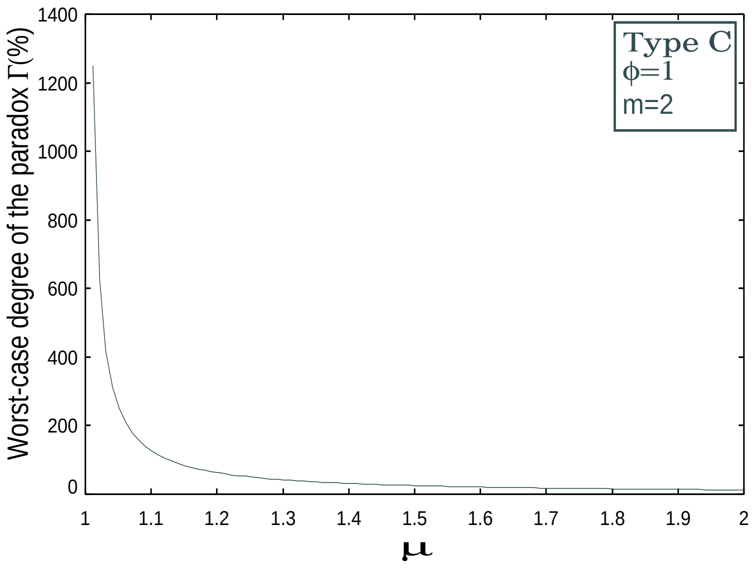

Figure 5.2: The WCDP (Γ) in complete symmetry given the values of μ and $\phi = 1$ with $m = 2$ using the communication means of type (C)

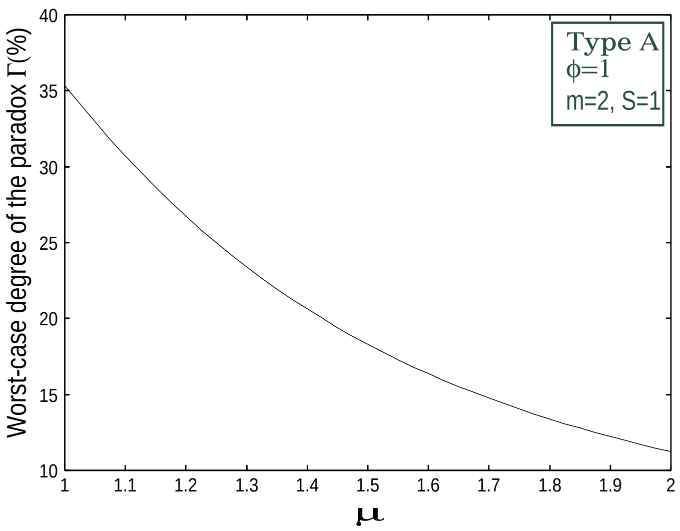

Figure 5.3: The WCDP (Γ) in complete symmetry given the values of μ and $\phi = 1$ with $m = 2$ and $s = 1$ using the communication means of type (A)

Table 5.2: The effect of changing μ while keeping $\phi = 1$ on the WCDP in complete symmetry with $m = 2$ and $s = 1$ using the communication means of types (A), when the processing rate approaches the arrival rate

μ	1.01	1.001	1.0001	1.00001
Γ(%)	34.8587	35.3052	35.3503	35.3510

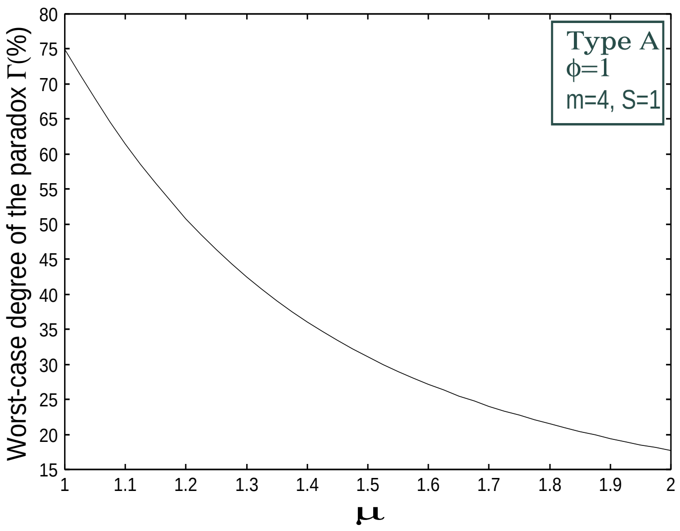

Figure 5.4: The WCDP (Γ) in complete symmetry given the values of μ and $\phi = 1$ with $m = 4$ and $s = 1$ using the communication means of type (A)

Table 5.3: The effect of changing μ while keeping $\phi = 1$ on the WCDP in complete symmetry with $m = 4$ and $s = 1$ using the communication means of types (A), when the processing rate approaches the arrival rate

μ	1.01	1.001	1.0001	1.00001
Γ(%)	73.5149	74.8501	74.9849	74.9851

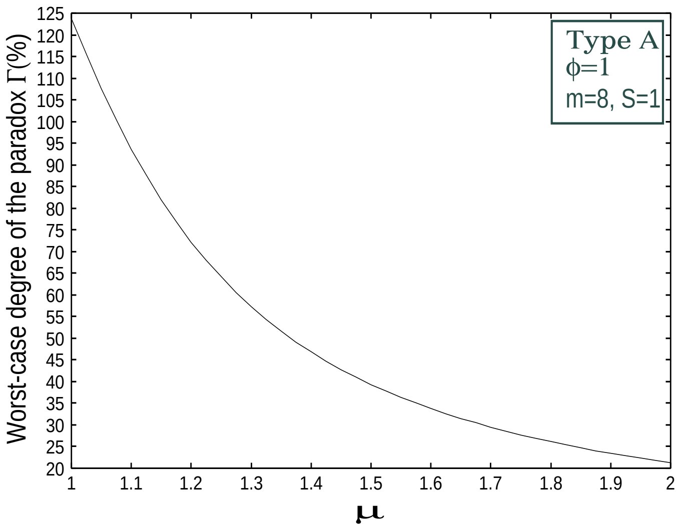

Figure 5.5: The WCDP (Γ) in complete symmetry given the values of μ and $\phi = 1$ with $m = 8$ and $s = 1$ using the communication means of type (A)

Table 5.4: The effect of changing μ while keeping $\phi = 1$ on the WCDP in complete symmetry with $m = 8$ and $s = 1$ using the communication means of types (A), when the processing rate approaches the arrival rate

μ	1.01	1.001	1.0001	1.00001
Γ(%)	120.2931	123.3942	123.7087	123.7124

directions: overall, individual and no symmetry property is maintained. In particular, we consider a family of systems for which condition (5.13) holds for the cases with overall, individual and no symmetry property is maintained. We have added condition (5.13) to be sure that ϕ_i is very close to μ_i to be able to see what will happen to the WCDP if the system parameter setting gradually departs the complete symmetry property.

5.6.2 Overall Symmetry Maintained

The two Figures 5.6 and 5.9 show how the WCDP depends on the combination of μ_1 and μ_2, with ϕ_1 and ϕ_2 given by condition (5.13) and $\frac{\mu_1}{(\mu_1 - \phi_1)^2} = \frac{\mu_2}{(\mu_2 - \phi_2)^2}$ (i.e., overall symmetry property is maintained) using the communication means of types (A) and (C) respectively.

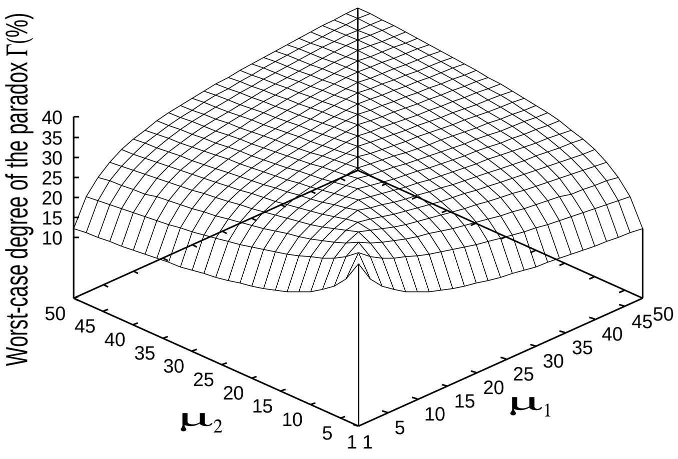

Figure 5.6: The effect of changing the system parameters while keeping the overall symmetry property among nodes on the WCDP using the communication means of type (A)

From the two Figures 5.6 and 5.9, it is observed that the WCDP gets it's maximum value when $\mu_1 = \mu_2$ and thus, $\phi_1 = \phi_2$ (i.e., in complete symmetry) and it increases as μ_1

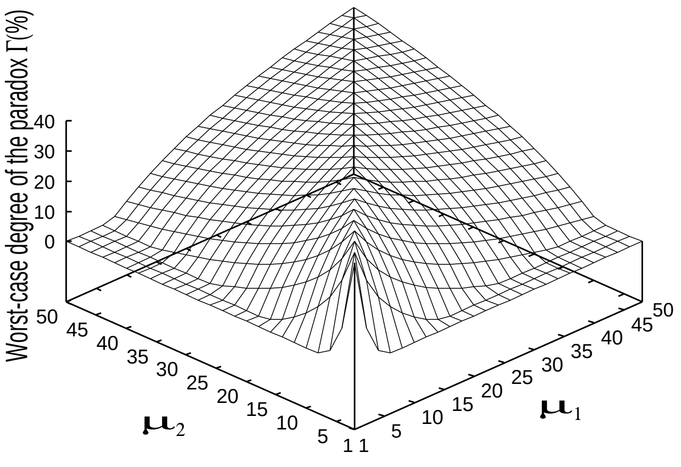

Figure 5.7: The effect of changing the system parameters while keeping the individual symmetry property among nodes on the WCDP using the communication means of type (A)

and μ_2 increase. It decreases slowly as the system parameter setting gradually departs the complete symmetry while keeping the overall symmetry property.

5.6.3 Individual Symmetry Maintained

The two Figures 5.7 and 5.10 show how the WCDP depends on the combination of μ_1 and μ_2, with ϕ_1 and ϕ_2 given by condition (5.13) and $\frac{1}{\mu_1 - \phi_1} = \frac{1}{\mu_2 - \phi_2}$ (i.e., the individual symmetry property is maintained) using the communication means of types (A) and (C) respectively.

From the two Figures 5.7 and 5.10, it is observed that the WCDP gets it's maximum value when $\mu_1 = \mu_2$ and thus, $\phi_1 = \phi_2$ (i.e., in complete symmetry) and it increases as μ_1 and μ_2 increase. It decreases a little bit more rapidly than that obtained in the overall symmetry case as the system parameter setting gradually departs the complete symmetry while keeping the individual symmetry property.

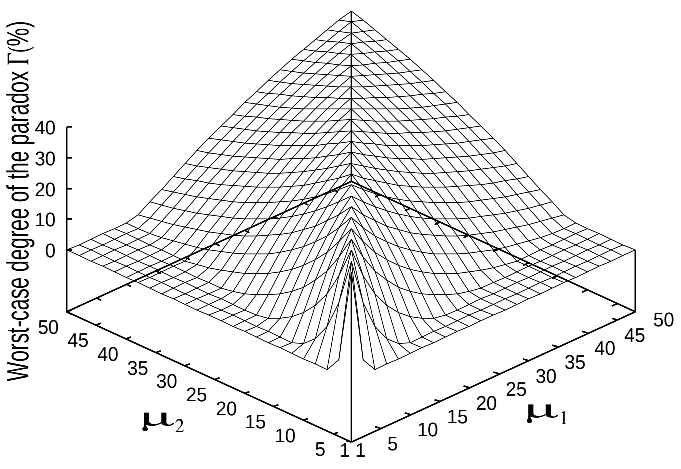

Figure 5.8: The effect of changing the system parameters without keeping any kind of symmetry among nodes on the WCDP using the communication means of type (A)

5.6.4 No Symmetry Maintained

The two Figures 5.8 and 5.11 show how the WCDP depends on the combination of μ_1 and μ_2, with ϕ_1 and ϕ_2 given by condition (5.13) and $\frac{\phi_1}{\mu_1 - \phi_1} = \frac{\phi_2}{\mu_2 - \phi_2}$ (i.e., no symmetry property is maintained) using the communication means of types (A) and (C) respectively.

From the two Figures 5.8 and 5.11, it is observed that the WCDP gets it's maximum value when $\mu_1 = \mu_2$ and thus, $\phi_1 = \phi_2$ (i.e., in complete symmetry) and it increases as μ_1 and μ_2 increase. It decreases more rapidly than that obtained in the individual and the overall symmetry cases as the system parameter setting gradually departs the complete symmetry without keeping any kind of symmetries.

Remark 5.6.1 *It is very important to note that, in this case, the system still has a kind of symmetry such that the expected queue length (the number of jobs that stay in each node) is identical when no communication means is available.*

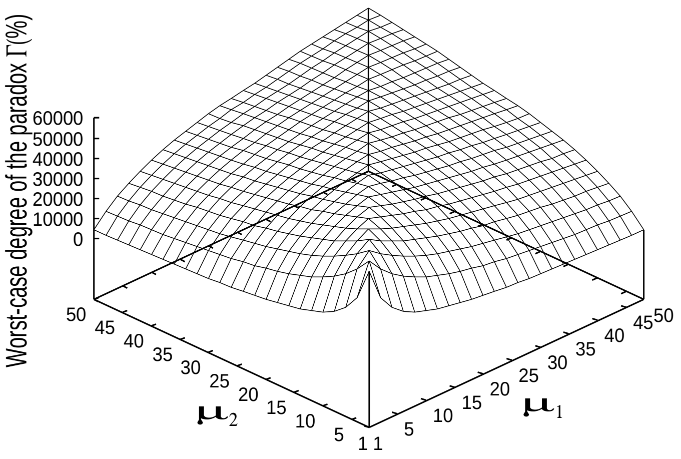

Figure 5.9: The effect of changing the system parameters while keeping the overall symmetry property among nodes on the WCDP using the communication means of type (C)

5.6.5 Complete Symmetry vs. No Symmetry

From the previous examples presented in sections 5.6.2, 5.6.3 and 5.6.4, it is observed that the WCDP is largest in complete symmetry where the arrival rate approaches the processing rate. But it is very important to note that in these cases (overall symmetry, individual symmetry and no symmetry), each case has a relation that correlates ϕ_i and μ_i which may imply a kind of symmetry between nodes. What will happen if we did not keep that relation? Will we find an asymmetric case where the WCDP is greater than that obtained in the complete symmetry case when the arrival rate approaches the processing rate? Or the obtained result will be ensured. And also, what will be the effect of increasing the number m of nodes in the system and the number s of channels in every communication line on the WCDP? We answer these questions through a number of numerical examples. We classify the numerical examples based on the type of the communication means as follows:

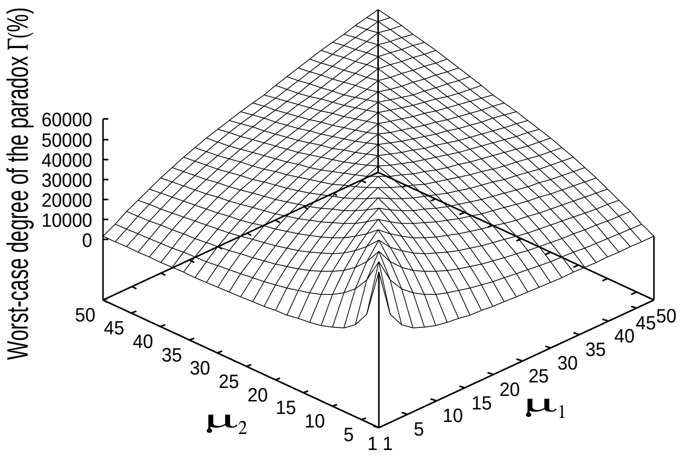

Figure 5.10: The effect of changing the system parameters while keeping the individual symmetry property among nodes on the WCDP using the communication means of type (C)

Communication means of type (A)

In Figures 5.12, 5.13 and 5.14, using the communication means of type (A), with $m = 2$ and $s = 1, 4$ and 100, respectively, that is, we have four independent parameters μ_1, ϕ_1, μ_2 and ϕ_2. Without a loss of generality, we scaled down ϕ_1 to 1 and thus we have only three independent parameters. We set them as follows, $\mu_i = 1(0.1)2, i = 1, 2$ *(i.e., the value of μ_i is varied from 1(jobs/sec) to 2(jobs/sec) in steps of 0.1(jobs/sec))* and $\phi_2 = 0.5(0.01) < \mu_2$. For each given value of μ_1, through a finer search, we compute the following $\max_{\mu_2,\phi_2} \Gamma$ and then we compare it with the WCDP that is obtained in the corresponding complete symmetry case. As shown in Figures 5.12, 5.13 and 5.14, it is observed that the WCDP increases and it converges to a certain limit as the arrival rate approaches the processing rate, and we did not find any asymmetric case where the WCDP is greater than what is obtained in the complete symmetry case where the arrival rate approaches the processing rate. Furthermore, it is observed that the WCDP increases as the number s of channels in

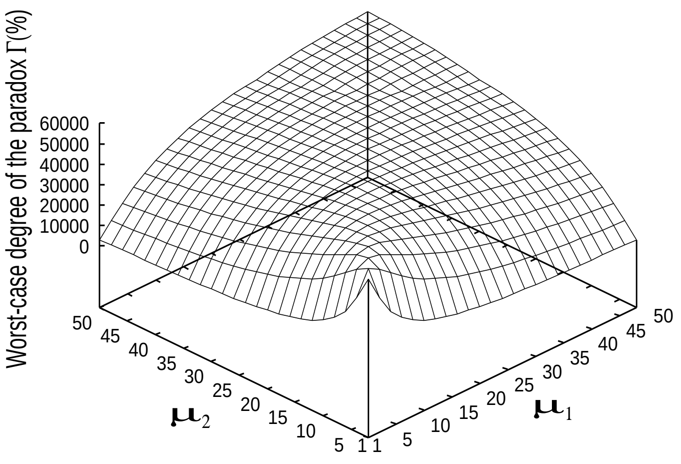

Figure 5.11: The effect of changing the system parameters without keeping any kind of symmetry among nodes on the WCDP using the communication means of type (C)

every communication line increases and when $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of what is obtained with the same system parameter setting but with $s = 1$.

In Figures 5.15, 5.16 and 5.17, using the communication means of type (A), with $m = 4$ and $s = 1, 4$ and 100, respectively, we observe the effect of the number m of nodes in the system on this phenomena. We have eight independent parameters $\mu_i, \phi_i, i = 1, \cdots, 4$. Again, without a loss of generality, we scaled down ϕ_1 to 1 and thus we have seven independent parameters. We set them as follows, $\mu_i = 1(0.1)2$, $i = 1, \cdots, 4$ and $\phi_i = 0.5(0.01) < \mu_i, i = 2, 3, 4$. For each given value of μ_1, through a finer search, we compute the following $\max_{\mu_i,\phi_i} \Gamma, i = 2, 3, 4$ and then we compare it with the WCDP that is obtained in the corresponding complete symmetry case. From Figures 5.12, 5.13, 5.14, 5.15, 5.16, and 5.17, it is observed that the WCDP increases as the number m of nodes in the system increases. Also, with $m = 4$ as shown in Figures 5.15, 5.16, and 5.17, it is observed that the WCDP increases and it converges to a certain limit as the arrival rate approaches the processing rate. Again we did not find any asymmetric case where the WCDP is greater than what is obtained in the complete symmetry case where the arrival rate approaches the processing rate, which ensures the results obtained earlier. It is also observed that the WCDP increases as the number s of channels in every communication line increases and when $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of what is obtained with the same system parameter setting but with $s = 1$.

Communication means of type (B)

The results obtained when the communication means of type (B) is used show the same tendency as what is obtained when the communication means of type (A) is used. For this reason, we only present a part of this results here. In Figures 5.18 and 5.19, with the communication means of type (B), $m = 4$ and $s = 1$ and 100, respectively, we show the results of the same experiments as that performed with type (A). Like the results obtained when the communication means of type (A) is used, it is observed that the WCDP increases and it converges to a certain limit as the arrival rate approaches the processing rate, and we have found that there is no asymmetric case where the WCDP is greater than what is obtained in the complete symmetry case where the arrival rate approaches the processing rate. Furthermore, it is observed that the WCDP increases as the number s of channels in every communication line increases and when $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of what is obtained with the same system parameter setting but with $s = 1$.

Communication means of type (C)

Figures 5.20 and 5.21 show the results of the same experiments using the communication means of type (C) with $m = 2$ and $m = 4$ respectively. We observe that as the arrival rate approaches the processing rate, the WCDP increases without bound and that of asymmetric cases approaches that of the symmetric cases. Thus, we observe that for any asymmetric case there exist a complete symmetric case that has the WCDP greater than that of the asymmetric case.

In each of the above cases, it has been seen that, in the class optimum, there exists no asymmetric case that has the WCDP greater than that of a complete symmetric case where the arrival rate approaches the processing rate, and therefore that, in the class optimum, the WCDP is largest in complete symmetry.

5.7 Conclusion

We have presented a number of numerical examples for the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for load balancing. From these examples, it is observed that in the class optimum, the WCDP is largest in the complete symmetry case where the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as any of the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

In conclusion, it seems that, in the systems of symmetrical nodes, adding means of job forwarding looks apparently ineffective and in some class optimum, adding means of job forwarding causes mutual job forwarding among nodes and bring about the paradox. If the results observed in this study hold generally, we think that more exhaustive research into these problems is worth pursuing in order to gain insight into the optimal design and QoS (quality of service) management of distributed computer systems, communication networks, etc.

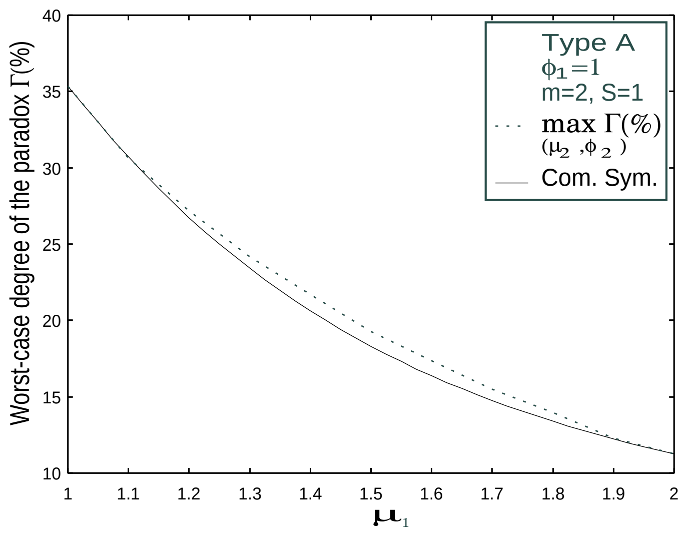

Figure 5.12: Comparison between the values of Γ that is obtained in complete symmetry with $m = 2, s = 1$ and $\max_{\mu_2,\phi_2} \Gamma$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

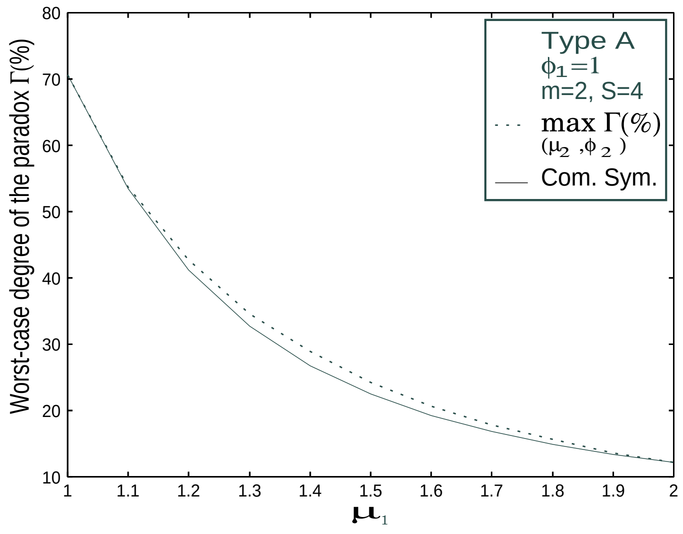

Figure 5.13: Comparison between the values of Γ that is obtained in complete symmetry with $m = 2, s = 4$ and $\max_{\mu_2,\phi_2} \Gamma$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

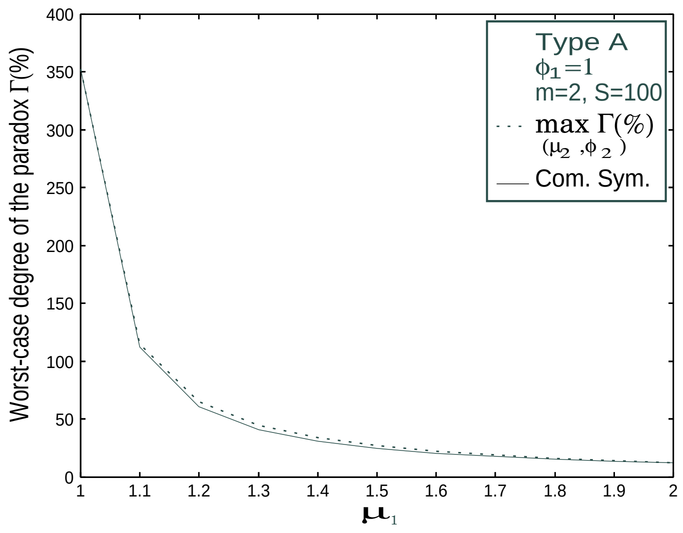

Figure 5.14: Comparison between the values of Γ that is obtained in complete symmetry with $m = 2, s = 100$ and $\max_{\mu_2,\phi_2} \Gamma$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

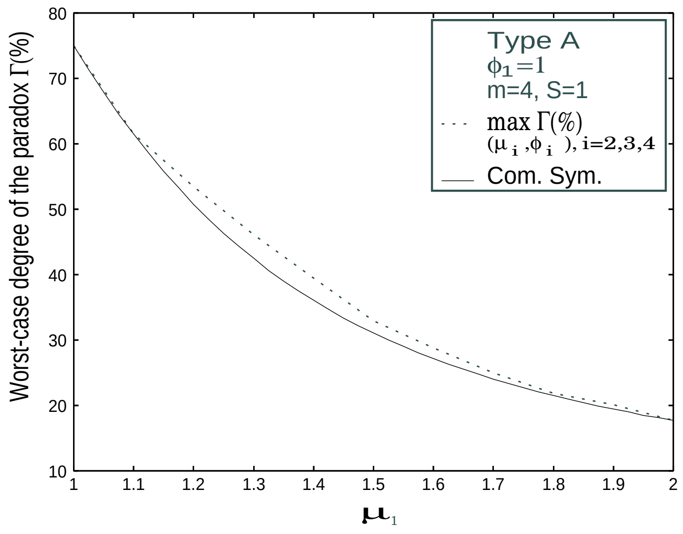

Figure 5.15: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4, s = 1$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

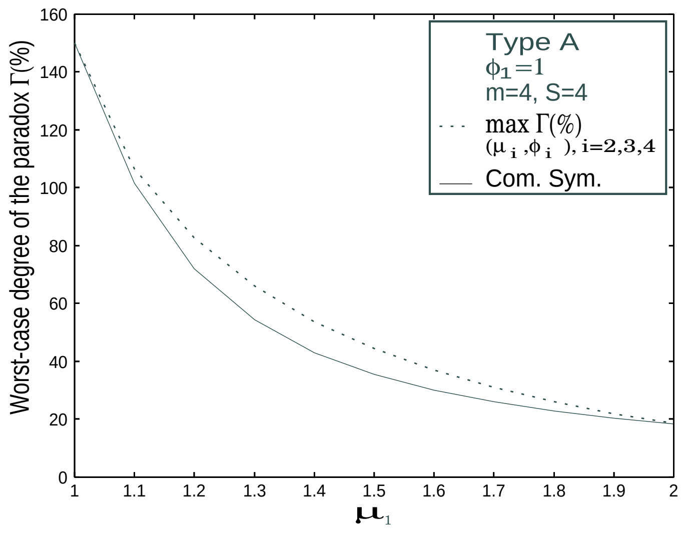

Figure 5.16: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4, s = 4$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

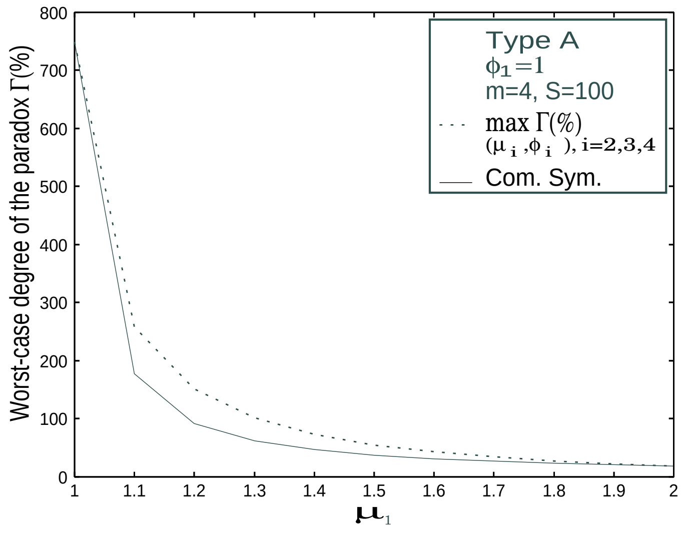

Figure 5.17: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4, s = 100$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (A)

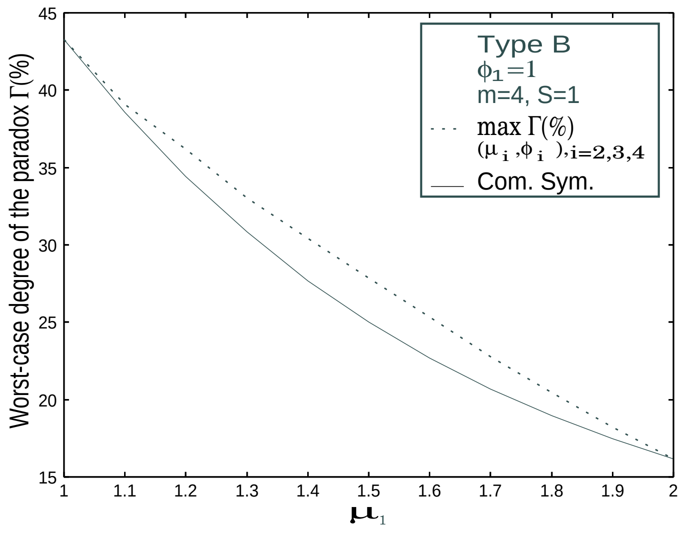

Figure 5.18: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4, s = 1$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (B)

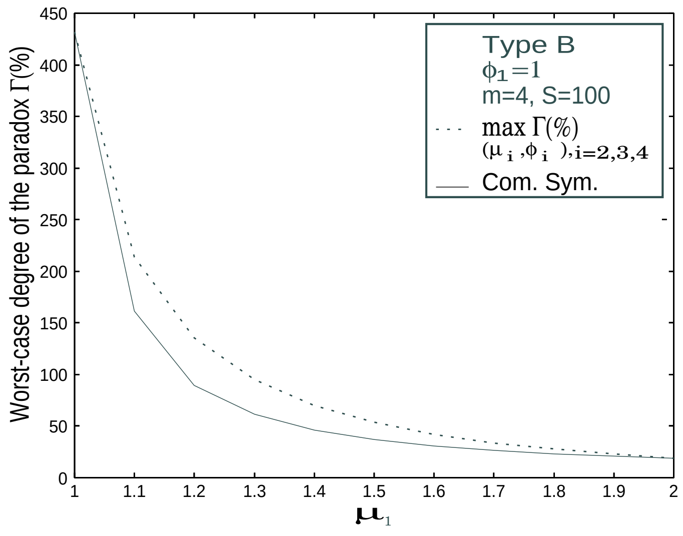

Figure 5.19: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4, s = 100$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (B)

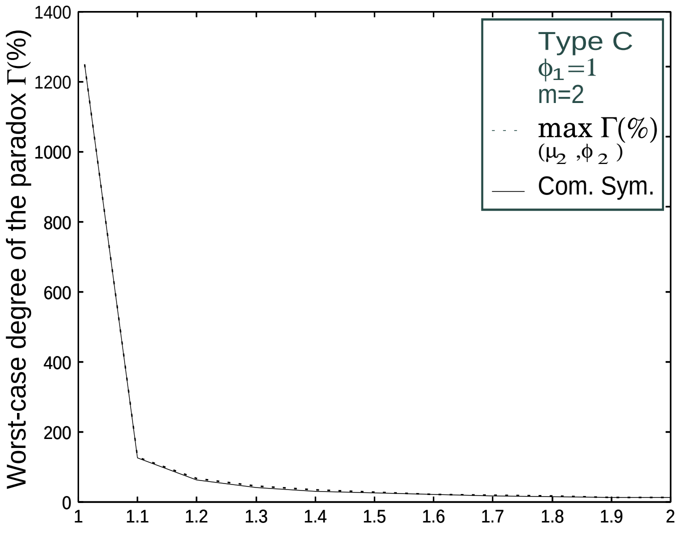

Figure 5.20: Comparison between the values of Γ that is obtained in complete symmetry with $m = 2$ and $\max_{\mu_2,\phi_2} \Gamma$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (C)

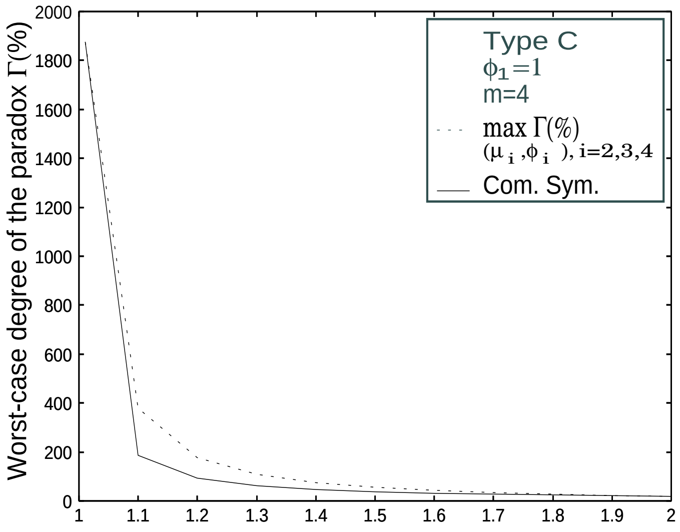

Figure 5.21: Comparison between the values of Γ that is obtained in complete symmetry with $m = 4$ and $\max_{\mu_i,\phi_i} \Gamma$, $(i = 2, 3, 4)$ for every given value of μ_1 and $\phi_1 = 1$ using the communication means of type (C)

Chapter 6

Conclusions and Future Work

One of the main advantages of distributed computer systems over the stand-alone systems is that distributed computer systems can share job processing in the event of overloads. Load balancing involves the distribution of jobs throughout a networked computer system, thus increasing throughput without having to obtain additional or faster computer hardware. Load balancing policies may be either static or dynamic.

In this thesis, both static and dynamic load balancing policies have been considered. Throughout our study, we used three performance aspects (objectives or optima) namely: overall optimum, individual optimum and class optimum with static and dynamic load balancing policies to optimize the performance of the considered distributed computer systems.

In Chapter 2, we present a brief survey for the previous and current load balancing studies related to our work.

In chapter 3, we propose a static and a dynamic overall optimal load balancing policies. The objective of both policies is to minimize the overall system mean response time. The performance of these two policies is compared on the MF-PC network model where truly optimal solutions of both static and dynamic load balancing policies have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, the overheads due to the two policies are assumed to be negligible. The $[L, q]$ threshold rule is used as a dynamic overall optimal load balancing policy. For this policy (i.e., [L,q] threshold rule), a numerical algorithm for obtaining the optimal values of L and q is proposed. Analytically, it is proved that the minimum value of the overall system mean response time is obtained by the dynamic overall optimal load balancing policy with the value of the threshold parameter $q = 0$ and the suitable selection of the other threshold parameter L. Also, we analytically proved the existence and uniqueness of optimal solution for the other threshold parameter L. That is, we need to choose only the proper value of L with q fixed to be 0 in finding the set of parameter values of the threshold rule that gives the minimum value for the overall system mean response time. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the overall system mean response time using the static overall optimal load balancing policy and the dynamic overall optimal load balancing policy are studied through numerical experimentation. The results show that, in the model examined, the overall system mean response time is improved by the dynamic overall optimal load balancing policy over that of the static overall optimal load balancing policy at most about 30% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The maximum improvement ratio is achieved for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase.

In chapter 4, we propose a static and a dynamic individually optimal load balancing policies. In these policies, every job strives to optimize (minimize) its own mean response time independently of the other jobs. The performance of these two policies is compared on the MF-PC network model where truly optimal solutions of both static and dynamic load balancing policies have been characterized. The analytical tractability of the model encourage us to perform such comparison analytically, for this reason, the overheads due to the two policies are assumed to be negligible. The $[L, q]$ threshold rule is used as a dynamic individually optimal load balancing policies. Three independent parameters are considered: job processing rate μ at the Q_{MF} node, job processing rate θ at the Q_{PC} node and job arrival rate λ to the system. Without a loss of generality, θ is scaled down to 1. The effects of changing the other two parameters (λ and μ) on the mean job response time using the SIOLBP and the DIOLBP are studied through numerical experimentation. The results show that the DIOLBP outperforms the SIOLBP in the mean job response time, at most about 48% in the range of parameter values examined while the overheads due to the two policies are not taken into account. The difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it increases as λ and μ increase. We also examined the job flow traffic in the proposed system model by computing the ratio that an arriving job at the system goes to Q_{MF} node under the SIOLBP and the DIOLBP. The results show that, there is a difference between the ratio that a job arriving at the system goes to the Q_{MF} under the SIOLBP and the DIOLBP. That difference is of a certain magnitude for the cases where $\lambda \sim \mu$ for rather large values of both and it decreases as λ and μ increase. Through the course of the numerical experimentation, we observed that if the $[L, q]$ threshold rule is used as a DIOLBP, in this case both of the control parameters L and q have a effect in satisfying the equilibrium in between the two system facilities. And also, it is noticed that the equilibrium threshold parameter L is a decreasing function of λ

and it approaches μ/θ.

In chapter 5, we present a number of numerical examples around the Braess-like paradox wherein adding a communication capacity to the system for the sharing of jobs between nodes leads to the performance degradation for all users in the class optimum for static load balancing. Three different types of communication means (A), (B) and (C) are considered. Based on the system parameter setting, three types of symmetries *(overall symmetry, individual symmetry and complete symmetry)* are defined. From the numerical examples, it is observed that in class optimum, the worst-case degree of the paradox (WCDP) is largest (i.e., the worst performance is obtained) in the complete symmetry case where the arrival rate approaches the processing rate. And, as the system parameter setting gradually departs the above-mentioned symmetric case without keeping any kind of symmetries, the WCDP decreases rapidly. It decreases slowly (slower) if the system parameter setting gradually departs the complete symmetry while keeping the individual (overall) symmetry property. Indeed, it is also observed that in complete symmetry, as the arrival rate approaches the processing rate, the WCDP converges to a certain limit if any of the communication means of types (A) and (B) is used and it may increase without bound if the communication means of type (C) is used. A final point is that, using any of the communication means of types (A) and (B), the WCDP increases as the number s of channels in every communication line increases and it is noticed that if $s > 1$, the WCDP increases to at most about $\sqrt{s}$ times of that obtained with the same parameters setting but with $s = 1$.

In our future work, we consider the possibility of an extension to the work done in chapter 5 using a dynamic load balancing policy. This is because the model studied in chapter 5 could be considered as a basic model for the GRID computing infrastructure where the dynamic load balancing is intended to be used for improving the performance of the GRID. We also think that more exhaustive research into Braess-like paradox problem is worth pursuing in order to gain insight into the optimal design and quality of service management of distributed computer systems, communication networks, etc.

Bibliography

[1] Hac A. and Jin X. Dynamic load balancing in a distributed system using a decentralized algorithm. In *IEEE Int. Conf. Dist. Computing Systems*, pages 170–177, 1987.

[2] Hac A. and Jin X. A decentralized algorithm for dynamic load balancing with file transfer. *J. of Systems and Software*, 16(6):37–52, 1991.

[3] Haurie A. and P. Marcotte. On the relationship between nash-cournot and wardrop equilibria. *Networks*, 15:295–308, 1985.

[4] Iqbal M. A., Saltz J. H., and Bokhari S. H. A comparative analysis of static and dynamic load balancing strategies. In *Proc. 6th IEEE Conf. on Distributed Computing Systems*, pages 140–147, 1986.

[5] Korilis Y. A., Lazer A. A., and Orda A. Architecting noncooperative networks. *IEEE J. Selected Areas in Communications*, 13:1241–1251, 1995.

[6] Korilis Y. A., Lazer A. A., and Orda A. Avoiding the braess paradox in noncooperative networks. *J. Appl. Prob.*, 36:211–222, 1999.

[7] Orda A., Rom R., and Shimkin N. Competitive routing in multiuser communication networks. *IEEE/ACM Trans. Networking*, 1:614–627, 1993.

[8] Shirazi B. A., Hurson A. R., and Kavi K. M. *Scheduling and load balancing in parallel and distributed systems*. IEEE Computer Society Press, Tokyo, 1995.

[9] Calvert B. The down-thomson effect in a markov process. *Prob in the Engineering and Info Sciences*, 11:327–340, 1997.

[10] Calvert B., Solomon W., and Ziedins I. Braess's paradox in a queueing network with state-dependent routing. *J. Appl. Prob.*, 34:134–154, 1997.

[11] George C. Dynamic load balancing for distributed memory multiprocessors. *Journal of Parallel and Distributed computing*, 7:279–301, 1989.

[12] Hui C. C. and Chanson S. T. Theoritical analysis of the heterogeneous dynamic load balancing problem using a hydrodynamic approach. *J. of Parallel and Distributed Computing*, 43:139–146, 1997.

[13] Kim C. and Kameda H. Optimal static load balancing of multi-class jobs in a distributed computer system. In *10th IEEE International Conference on Distributed Computing Systems*, pages 562–569, Paris, France, June 1990.

[14] Kim C. and Kameda H. Optimal static load balancing of multi-class jobs in a distributed computer system. *The Transactions of the IEICE*, E73(7):1207–1214, July 1990.

[15] Kim C. and Kameda H. An algorithm for optimal static load balancing in distributed computer systems. *IEEE Trans. Computers*, 41(3):381–384, March 1992.

[16] Kim C. and Kameda H. Parametric analysis of static load balancing of multi-class jobs in a distributed computer system. *IEICE Trans. Inf. and Syst.*, E75-D(4):527–534, July 1992.

[17] Luis M. C. and Isaac D. S. Rate of change load balancing in distributed and parallel systems. *Parallel Computing*, pages 1213–1230, 2000.

[18] Steve J. C. Distributed and multiprocessor scheduling. *ACM Computing Surveys*, 28(1):233–235, 1996.

[19] Braess D. Uber ein paradoxen aus der verkehrsplanung. *Unternechmensforschung*, 12:258–268, 1968.

[20] Stefano A. D., Bello L. L., and Mirabella O. Assessment of job allocation on heterogeneous computer networks. *Advances in Engineering Software*, 28:63–71, 1997.

[21] Stella C. D. Traffic assignment problem for multiclass user transportation networks. *Transportation Science*, 6:73–87, 1972.

[22] Edmundo de Souza e Silva and Mario G. Load balancing in distributed sytems with multiple classes and site constraints. In E. Gelenbe, editor, *PERFORMANCE '84*, pages 17–33. Elsevier Science Publishers B. V. (North-Holland), 1984.

[23] Altman E. and Kameda H. Equilibria for multiclass routing in multi-agent networks. In *Proc. of the 40th IEEE Conference on Decision and Control*, pages 604–609, 2001.

[24] Altman E., Kameda H., and Hosokawa Y. Nash equilibria in load balancing in distributed computer systems. *International Game Theory Review*, 4(2):91–100, 2002.

[25] Altman E. and Shimkin N. Individual equilibrium and learning in processor sharing systems. *Operations Research*, 46:776–784, 1998.

[26] Cohen J. E. and Jeffries C. Congestion resulting from increased capacity in single server queueing networks. *IEEE/ACM Trans. Networking*, 5:1220–1225, 1997.

[27] Cohen J. E. and Kelly F. P. A paradox of congestion in a queueing networks. *J. Appl. Prob.*, 27:730–734, 1990.

[28] Douglis F. and Ousterhout J. Transparent process migration: Design alternatives and the sprite implementation. *Software-Practice and Experience*, 21(8):757–785, 1991.

[29] El-Zogdhy S. F., Kameda H., and Li J. Numerical studies on braess-like paradoxes for non-cooperative load balancing in distributed computer systems. *International workshop (Networking Games and Resource Allocation)*, July 2002.

[30] El-Zoghdy S. F., Kameda H., and Li J. Numerical studies on a paradox for non-cooperative static load balancing in distributed computer systems. *Computers and Operations Research (Accepted)*.

[31] El-Zoghdy S. F., Kameda H., and Li J. A comparative study of static and dynamic individually optimal load balancing policies. In *Proc. of the IASTED International Conference on Networks Parallel and Distributed Processing, and Applications*, pages 200–205, Tsukuba Science City, Japan, October 2002.

[32] El-Zoghdy S. F., Kameda H., and Li J. Comparison of dynamic vs. static load balancing policies in a mainframe-personal computer network model. *INFORMATION*, 5(4):431–446, 2002.

[33] El-Zoghdy S. F., Kameda H., and Li J. Numerical studies on paradoxes in non-cooperative distributed computer systems. *Game Theory and Application*, 9, *(Accepted)*.

[34] Bean N. G., Kelly F. P., and Taylor P. G. Braess's paradox in a loss network. *J. Appl. Prob.*, 34:155–159, 1997.

[35] Boon G., Yoke-Hean L., and Sanjay J. Load balancing for conservative simulation on shared memory multiprocessor systems. *14th Workshop on Parallel and Distributed Simulation (PADS 2000)*, May 2000.

[36] Shivaratri N. G. and Krueger P. Two adaptive location policies for global scheduling algorithms. In *Proc. of the 14th Int. Conf. Dist. Computer Systems*, pages 502–509, 1990.

[37] Kameda H. How harmful the paradox can be in the braess/cohen-kelly-jeffries networks. In *Proc. IEEE INFOCOM*, 2002.

[38] Kameda H., Altman E., Kozawa T., and Hosokawa Y. Braess-like paradoxes in distributed computer systems. *IEEE Trans. Automatic Control*, 45:1687–1691, 2000.

[39] Kameda H., El-Zoghdy S. F., and Li J. A performance comparison of dynamic vs. satic load balancing policies in a mainframe–personal computer network model. *ISE Technical Report*, ISE-TR-01-183, September 2001.

[40] Kameda H., El-Zoghdy S. F., Inhwan R., and Li J. A performance comparison of dynamic vs. static load balancing policies in a mainframe–personal computer network model. In *Proc. 39th IEEE Conf. on Decision and Control*, pages 1415–1420, Sydney, Australia, December 2000.

[41] Kameda H., Li J., Kim C., and Zhang Y. *Optimal Load Balancing in Distributed Computer Systems*. Springer, Tokyo, 1997.

[42] Kameda H., Ohta M., and Hosokawa Y. Effects of symmetry on paradoxical cost degradation in a nash non-cooperative network system. In *Proc. IFAC Symposium on Modeling and control of Economic Systems*, September 2001.

[43] Kameda H. and Pourtallier O. Paradoxes in distributed decisions on optimal load balancing for networks of homogeneous computers. *J. of the ACM*, 49(3):407–433, 2002.

[44] Kameda H., Kozawa T., and Li J. Anomalous relations among various performance objectives in distributed computer systems. In *Proc. of the 1st IEEE World Congress on Systems Simulation*, pages 459–465, Singapore, 1997.

[45] Kameda H., Hosokawa Y., and Pourtallier O. Effects of symmetry on braess-like paradoxes in distributed computer systems - a numerical study -. In *Proc. of the 40th IEEE Conference on Decision and Control*, pages 831–836, 2001.

[46] Kameda H. and Zhang Y. Uniqueness of the solution for optimal static routing in open bcmp queueing networks. *Mathematical and Computer Modeling*, 22(10-12):119–130, 1995.

[47] Li J. and Kameda H. Optimal load balancing in tree networks with two-way traffic. *The international J. of Distributed Informatique*, 25(12):1335–1348, 1993.

[48] Li J. and Kameda H. A decomposition algorithm for optimal static load balancing in tree hierarchy network configurations. *IEEE Trans. Parallel and Distributed Systems*, 5(5):540–548, 1994.

[49] Li J. and Kameda H. Optimal load balancing in star network configurations with two-way traffic. *J. Parallel and Distributed Computing*, 23(3):364–375, 1994.

[50] Li J. and Kameda H. Load balancing problems for multiclass jobs in distributed/parallel computer systems. *IEEE Trans. Computer*, 47(3):322–332, 1998.

[51] Stankovic J. Simulations of three adaptive, decentralized controlled, job scheduling algorithms. *Computer Networks*, 8:199–217, 1984.

[52] Stankovic J. An application of bayesian decision theory of decentralized control of job scheduling. *IEEE Trans. on Computers*, C-34(2):117–130, 1985.

[53] Watts J. and Taylor S. A practical approach to dynamic load balancing. *IEEE Trans. on Parallel and Distributed Systems*, 9(3):235–248, 1998.

[54] Antonis K., Garofalakis J., Mourtos J., and Spirakis P. A hierarchical adaptive distributed algzrithm for load balancing. *http://citeseer.nj.nec.com/986548.html*, April 2003.

[55] Antonis K., Garofalakis J., and Spirakis P. A comparative symmetrical transfere policy for load sharing. In *Proc. Euro-Par*, pages 352–355, 1998.

[56] Benmohammed-Mahieddine K., Dew P. M., and Kara M. A periodic sysmmetrically-initiated load balancing algorithm for distributed systems. In *Proc. of the 14th Int. Conf. Dist. Computing Syst.*, pages 616–623, Poznan, Poland, June 1994.

[57] Goswami K., Devarakonda M., and Iyer R.K. Predication-based dynamic load-sharing heuristics. *IEEE Trans. Parallel and Distributed Systems*, 4(6):638–648, 1993.

[58] Helen D. K. and Ralph C. H. Load sharing in heterogeneous distributed systems. In *Proc. of the 2002 Winter Simulation Conference*, pages 489–496, 2002.

[59] Eager D. L., Lazowska E. D., and Zahorjan J. Adaptive load sharing in homogeneous distributed systems. *IEEE Trans. on Software Eng.*, SE-12(5):662–675, May 1986.

[60] Eager D. L., Lazowska E. D., and Zahorjan J. A comparison of receiver-initiated and sender-initiated adaptive load sharing. *Performance Evaluation*, 6(1):53–68, March 1986.

[61] Eager D. L., Zahorjan J., and Lazowska E. D. Speedup versus efficiency in parallel systems. *IEEE Transactions on Computers*, 38(3):408–423, March 1989.

[62] Fratta L., Gerla M., and Kleinrock L. The flow deviation method: An approach to store-and-forward communication network design. *Networks*, 3:97–133, 1973.

[63] Kalyani M. A. L., Wait R., and Ranasinghe D. N. Load balancing techniques for distributed memory multiprocessor. *http://www.tdb.uu.se/richard/pages/$IITC2000_papers$/Kalyani.pdf*, January 2001.

[64] Bozyigit M. History-driven dynamic load balancing for recurring applications on networks of workstations. *J. of Systems and Software*, 51:61–72, 2000.

[65] Dahlin M. Interpreting stale load information. *IEEE Trans. Parallel and Distributed Systems*, 11(10):1033–1047, 2000.

[66] Frank M. The braess paradox. *Mathematical Programming*, 20:283–302, 1981.

[67] Maurice M. Inria at the heart of grid computing research. *www.inria.fr/presse/dossier/gridcomputing/grid.en.pdf*, July 2003.

[68] Mitzenmacher M. How useful is old information? *IEEE Trans. Parallel and Distributed Systems*, 11(1):6–20, 2000.

[69] Sakib A. M. A note on optimal static load balancing in distributed computer systems. *http://arxiv.org/PS_cache/cs/pdf/0006/0006204.pdf*, Jun 2000.

[70] Schaar M., Efe K., Delcambre L., and Bhuyan L. N. Load balancing with network cooperation. In *IEEE Int. Conf. Dist. Computing Systems*, pages 328–335, 1991.

[71] Thomas L. M. Models and algorithms for predicting urban traffic equilibria. In M. Florian, editor, *Transportation Planning Models*, pages 153–185. Elsevier Science Publishers B. V. (North-Holland), 1984.

[72] Arora N. and Sen S. Resolving social dilemmas using genetic algorithms: Initial results. In *Proc. of the 7th International Conference on Genetic Algorithms*, pages 689–695, 1997.

[73] Tantawi A. N. and Towsley D. A general model for optimal static load balancing in star network configurations. In E. Gelenbe, editor, *PERFORMANCE '84*, pages 277–291. Elsevier Science Publishers B. V. (North-Holland), 1984.

[74] Tantawi A. N. and Towsley D. Optimal static load balancing in distributed computer systems. *Journal of the Association for Computing Machinery*, 32(2):445–465, April 1985.

[75] Kremien O. and Kramer J. Methodical analysis of adaptive load sharing algorithms. *IEEE Trans. Parallel and Distributed Systems*, 3(6):747–760, 1992.

[76] Dandamudi S. P. The effects of scheduling discipline on sender-initiated and receiver-initiated adaptive load sharing in homogeneous distributed systems. In *Proc. of the Int. Conf. Dist. Computing Syst.*, Vancouver, 1995.

[77] Dasgupta P., Majumber A. K., and Bhattacharya P. v_thr: An adaptive load balancing algorithm. *J. of Parallel and Distributed Computing*, 42:101–108, 1997.

[78] Dikshit P., Tripathi S. K., and Jalote P. Sahayog: A test bed for evaluating dynamic load-sharing policies. *Software - Practice and Experience*, 19(5):411–435, 1989.

[79] Kelly F. P. Loss networks. *Ann. Appl. Prob.*, 1:319–378, 1991.

[80] Kelly F. P. Network routing. *Phil. Trans. R. Soc. Lond*, A 337:343–367, 1991.

[81] Krueger P. and Shivaratri N. G. Adaptive location policy for global scheduling. *IEEE Trans. Softw. Eng.*, 20(6):432–444, 1994.

[82] Hassin R. and Haviv M. Equilibrium threshold strategies: the case of queues with priorities. In *Proc. Informs*, 1995.

[83] Keith W. R. and David D. Y. Optimal load balancing and scheduling in a distributed computer system. *J. of the ACM*, 38(3):676–690, July 1991.

[84] Mirchandaney R., Towsley D., and Stankovic J. A. Analysis of the effects of delays on load sharing. *IEEE Trans. on Computers*, 38(11):1513–1525, 1989.

[85] Mirchandaney R., Towsley D., and Stankovic J. A. Adaptive load sharing in heterogeneous distributed systems. *J. of Parallel and Distributed Computing*, 9:331–346, 1990.

[86] Mirchandaney R. and Stankovic J. Using stochastic learning automata for job scheduling in distributed processing systems. *J. Parallel and Distributed Computing*, 3:527–552, 1986.

[87] Pulidas S., Towsley D., and Stankovic J. A. Imbeding gradient estimators in load balancing algorithms. In *Proc. IEEE 8th International Conference on distributed computing systems*, pages 482–490, 1988.

[88] Zhou S. A trace-driven simulation study of dynamic load balancing. *IEEE Trans. Soft. Eng.*, 14(9):1327–1341, 1988.

[89] Boulogne T., Altman E., Kameda H., and Pourtallier O. Mixed equilibrium (me) for multiclass routing games. *IEEE Trans. Automatic Control*, 47(6):903–916, 2002.

[90] Matthew P. T. and Johnny S. K. W. A task migration algorithm for heterogeneous distributed computing systems. *Journal of Systems and Software*, 41(3):175–188, 1998.

[91] Roughgarten T. and Tardos E. How bad is selfish routing? In *Proc. of the 41th IEEE Annual Symposium on Foundation of Computer Science*, pages 93–102, 2000.

[92] Wang Y. T. and Morris R. J. T. Load sharing in distributed systems. *IEEE Transactions on Computers*, c-34(3):204–217, March 1985.

[93] Gopal T. V., Karthic N. S., Ramamurthy C., and Sankaranarayanan V. Load balancing in heterogeneous distributed systems. *Microelectron. Reliab.*, 36(9):1279–1286, 1996.

[94] Ioannis V. and Anthony E. Extension of the optimality of the threshold policy in heterogeneous multiserver queueing systems. *IEEE Trans. Automatic Control*, 33(1), 1988.

[95] Deng X., Liu H., Long J. S., and Xiao B. Competitive analysis of network load balancing. *J. of Parallel and Distributed Computing*, 40:162–172, 1997.

[96] Suen T. T. Y. and Wong J. S. K. Efficient task migration algorithm for distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 3(4):488–499, 1992.

[97] Zhang Y., Kameda H., and Hung S. L. Comparison of dynamic and static load balancing strategies in heterogeneous distributed systems. *IEE Proc. Compu. Digit. Tech.*, 144(2):100–106, 1997.

[98] Zhang Y., Hakozaki K., Kameda H., and Shimizu K. A performance comparison of adaptive and static load balancing in heterogeneous distributed systems. In *IEEE 28th Annual Simulation Symposium*, pages 332–340, April 1995.

[99] Yi-Chang Z., Ce-Kuen S., and Tyng-Yeu L. Automatic reconfiguration for maximizing system performance on software distributed sharde memory systems. In *Proc. of the IASTED International Conference on Networks Parallel and Distributed Processing, and Applications*, pages 53–58, Tsukuba Science City, Japan, October 2002.

Appendix A

Derivation of the overall mean response time of a job arriving at the MF-PC network model with the $[L, q]$ threshold rule

We derive here the overall mean response time of a job arriving at the MF-PC network model with the $[L, q]$ threshold rule, $E\left[W_{[L,q]}\right]$. Let P_k be the probability that the number of jobs in the Q_{MF} node is k. The state transition diagram is shown in Figure A.1. With this state transition diagram we have the following equations:

$$\begin{aligned} \lambda P_0 &= \mu P_1 \\ \lambda P_1 &= \mu P_2 \\ \cdots & \cdots \cdots \\ \lambda P_{L-1} &= \mu P_L \\ \lambda q P_L &= \mu P_{L+1}. \end{aligned} \tag{A.1}$$

Let $\rho = \lambda/\mu$. From A.1, we can easily have the recursions:

$$\begin{aligned} P_1 &= \rho P_0 \\ P_2 &= \rho^2 P_0 \end{aligned}$$

$$\begin{array}{rcl} \cdots & \cdots & \cdots \\ P_L & = & \rho^L P_0 \\ P_{L+1} & = & \rho^{L+1} q P_0, \end{array} \tag{A.2}$$

and if $\rho = 1$,

$$P_1 = P_2 = \cdots = P_L = P_0, \quad P_{L+1} = qP_0. \tag{A.3}$$

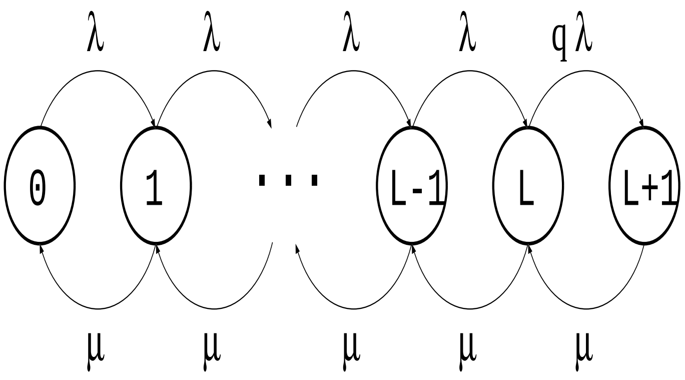

Figure A.1: State transition diagram

From A.2, we have

$$\begin{array}{rcl} P_1 + P_2 + \cdots + P_L & = & P_0(\rho + \rho^2 + \cdots + \rho^L) \\ & = & P_0 \dfrac{\rho - \rho^{L+1}}{1-\rho}. \end{array} \tag{A.4}$$

Note that $\sum_{i=0}^{L+1} P_i = 1$. We have

$$P_0 = \begin{cases} \dfrac{1-\rho}{1-\rho^{L+1}(1-q) - q\rho^{L+2}}, & \text{if } \rho \neq 1; \\ \dfrac{1}{L+1+q} & \text{if } \rho = 1. \end{cases} \tag{A.5}$$

Substituting relation A.5 to A.2 or A.3, we can have the probability that the number of jobs in the Q_{MF} node is k, $P_k(0 \le k \le L)$. With the above relations, we proceed to calculate the overall mean response time of a job arriving at the system. Let P be the probability that a job arriving at the system goes to the Q_{PC} node. With $[L, q]$ threshold rule, the arriving job will go to the Q_{PC} node with probability of 1 if the job finds the Q_{MF} node with states $L + 1, L + 2, \cdots$, and with probability of $1 - q$ if the job finds the Q_{MF} node with state L. Then P is expressed as

$$P = (1 - q)P_L + P_{L+1}. \tag{A.6}$$

The mean response time of a job that goes to Q_{PC} node is θ^{-1}. Let Q be the expected number of jobs (which includes the jobs in service) in the Q_{MF} node from state 0 to state $L + 1$ in the state transition diagram (see Figure 3.2 in chapter 3). By the Little's Law, the mean response time of a job arriving at the system goes to the Q_{MF} node is

$$QV^{-1},$$

where, V is the actual load rate to the Q_{MF} node, and is given by $V = \lambda(1-P)$. Therefore, the overall mean response time of a job arriving at the system with threshold $[L, q]$, $E\left[W_{[L,q]}\right]$, is

$$\begin{aligned} E\left[W_{[L,q]}\right] &= P\theta^{-1} + (1 - P)QV^{-1} \\ &= P\theta^{-1} + Q\lambda^{-1}. \end{aligned} \tag{A.7}$$

From A.4, Q can be calculated as follows:

$$Q = \sum_{i=1}^{L} iP_i + (L + 1)P_{L+1}. \tag{A.8}$$

By substituting relations A.6 and A.8 into A.7, we obtain the overall mean response time of a job arriving at the system with threshold $[L, q]$, $E\left[W_{[L,q]}\right]$. The relation is as follows:

$$E\left[W_{[L,q]}\right] = ((1-q)P_L + P_{L+1})\theta^{-1} + Q\lambda^{-1}, \tag{A.9}$$

where, if $\rho \neq 1$,

$$\begin{aligned}
P_L &= \rho^L P_0, && \text{(A.10)}\\
P_{L+1} &= q\rho^{L+1} P_0, && \text{(A.11)}\\
Q &= \sum_{i=1}^{L} iP_i + (L+1)P_{L+1}\\
&= P_0\rho\frac{(-(L+1)\rho^L)(1-\rho) + (1-\rho^{L+1})}{(1-\rho)^2}\\
&\quad +(L+1)P_0 q\rho^{L+1}, && \text{(A.12)}\\
P_0 &= \frac{1-\rho}{1-\rho^{L+1}(1-q) - q\rho^{L+2}}, && \text{(A.13)}
\end{aligned}$$

and if $\rho = 1$,

$$\begin{aligned}
P_L &= P_0, && \text{(A.14)}\\
P_{L+1} &= qP_0, && \text{(A.15)}\\
Q &= \sum_{i=1}^{L} iP_i + (L+1)P_{L+1}\\
&= \left(\sum_{i=1}^{L} i + (L+1)q\right)P_0\\
&= \left(\frac{L(L+1)}{2} + (L+1)q\right)P_0\\
&= \frac{(L+1)(L+2q)}{2(L+1+q)}, && \text{(A.16)}\\
P_0 &= \frac{1}{L+1+q}. && \text{(A.17)}
\end{aligned}$$

List of Publications and Presentations

April 2000 - March 2004

1. Hisao Kameda, Said Fathy El-Zoghdy, Inhwan Ryu, and Jie Li, “A Performance Comparison of Dynamic Vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network Model,” Proc. 39th IEEE Conference on Decision and Control (IEEE CDC’2000), Sydney, Australia, pp. 1415-1420, December 12-15, 2000.

2. Hisao Kameda, Said Fathy El-Zoghdy, and Jie Li, “A Performance Comparison of Dynamic Vs. Static Load Balancing Policies in a Mainframe - Personal Computer Network Model,” ISE Technical Report ISE-TR-01-183, September 14, 2001.

3. Said Fathy El-Zoghdy, Hisao Kameda, and Jie Li ,“Numerical Studies on Braess-Like Paradoxes for Non-Cooperative Load Balancing in Distributed Computer Systems,” International workshop “Networking Games and Resource Allocation,” Petrozavodsk, Karelia, Russia, (12- 15), July 2002.

4. Said Fathy El-Zoghdy, Hisao Kameda, and Jie Li, “A Comparative Study of Static and Dynamic Individually Optimal Load Balancing Policies,” IASTED International Conference on Networks, Parallel and Distributed Processing, and Applications (NPDPA 2002), Tsukuba, Japan, pp. 200-205, October (2-4) 2002.

5. Said Fathy El-Zoghdy, Hisao Kameda, and Jie Li, "A Performance Comparison of Dynamic Vs. Static Load Balancing Policies in a Mainframe-Personal Computer Network Model," INFORMATION, Vol. 5, No. 4, pp. 431-446, October 2002.

6. Said Fathy El-Zoghdy, Hisao Kameda, and Jie Li, "Numerical Studies on Paradoxes in Non-Cooperative Distributed Computer Systems," Game Theory and Applications, Vol. 9, (Accepted).

7. Said Fathy El-Zoghdy, Hisao Kameda, and Jie Li, "Numerical Studies on a Paradox for Non-Cooperative Static Load Balancing in Distributed Computer Systems," Computers and Operations Research, (Accepted).