Performance Evaluation of Contiguous and Noncontiguous Processor Allocation Strategies based on Common Communication Patterns for 2D Mesh Multicomputers

تقييم أداء استراتيجيات التخصيص المتجاور وغير المتجاور بناء على أنماط االتصال المعروفة في النظام ثنائي ألبعاد في متعددات الحواسيب

By

Areen Falah Ahmad Alabass

Supervisor

Prof. Ismail Ababneh

Co-Supervisor

Prof. Saad Bani-Mohammad

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Science in Computer Science

Deanship of Graduate Studies

Al al-Bayt University

1, 2019

Examination Committee

Examination committee	Signature
Prof. Ismail Ababneh (Supervisor)	……
Prof. Saad Bani-Mohammad (Co-Supervisor)	……
Prof. Omar Shatnawi	……
Prof. Khaled Batiha	……
Dr. Wail Mardini	……

Dedication

This thesis is dedicated to my parents, my Daughter, Prof. Ismail Ababneh, and Prof. Saad Bani-Mohammad for their courtliness, appreciating to support and encouragement.

Acknowledgments

This thesis was supervised by Prof. Ismail Ababneh and Prof. Saad Bani-Mohammed, who have spent a lot of time helping me to write the thesis. My great thanks for them to help me in choosing the main idea of the thesis, obtaining the results, writing the thesis and their encouragement to complete this thesis perfectly.

Also, my great thanks to my parents, my brothers, and my sisters, without their encouragements and support I could not do anything.

Areen Alabass

List of Content

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
MBS	Multiple Buddy Strategy
FCFS	First-Come-First-Served
FF	First Fit
BF	Best Fit
GABL	Greedy Available Busy List
FFT	Fast Fourier Transform
DQBT	Divide and Conquer Binomial Tree

Performance Evaluation of Contiguous and Noncontiguous Processor Allocation Strategies based on Common Communication Patterns for 2D Mesh Multicomputers

By

Areen Alabass

Supervisor

Prof. Ismail Ababneh

Co-Supervisor

Prof. Saad Bani-Mohammad

Abstract

Several studies in processors allocation indicate that noncontiguous allocation strategies dramatically better than contiguous allocation strategies with regard to mean system utilization and average turnaround time, regardless the used communication pattern. But, this is in reality not always true, the used communication pattern may have a great impact on the performance of contiguous and noncontiguous processor allocation in multi-computers, especially when each job does exactly one iteration of the given communication pattern. In this thesis,

the performance of most famous allocation strategies for 2D mesh-connected multi-computers is re-visited considering several important communication patterns, including the Near Neighbor, Ring, All to all, Divide and Conquer Binomial Tree (DQBT), Fast Fourier Transform (FFT), One to All, All to One, and Random communication patterns. The allocation strategies investigated are First Fit (FF) and Best Fit (BF) as contiguous allocation strategies

and Paging(0), Greedy Available Busy List (GABL), and Multiple Buddy Strategy (MBS) as noncontiguous allocation strategies. Two job size distributions have been considered which are uniform and uniform-decreasing distributions. Wide simulation experiments have been conducted to compare the performance of contiguous allocation with that of the noncontiguous allocation with regard to average turnaround time and mean system utilization using the ProcSimity simulator.

The simulation results for average turnaround time have shown that in near neighbor, FFT and DQBT communication patterns, the performance of contiguous allocation strategies (FF and BF) dramatically better than that of the noncontiguous allocation strategies (Paging(0), MBS and GABL) with regard to average turnaround; except for MBS in DQBT communication pattern. Also, the simulation results have shown that in one-to-all, random, ring and all-to-one communication patterns, the performance of noncontiguous allocation strategies (Paging(0), MBS and GABL) dramatically better than that of the contiguous allocation strategies (FF and BF) with regard to average turnaround time. For all-to-all communication pattern, the simulation results have shown that the performance of the contiguous allocation strategies (FF and BF) is better than that of the MBS noncontiguous allocation but the performance of GABL and Paging(0) is better than that of FF, BF, and MBS.

The results for system utilization time have shown that in all communication patterns that are considered in this research work, the noncontiguous allocation strategies dramatically better than the contiguous allocation strategies with regard to mean system utilization.

Chapter One

1. Introduction

Parallel computers have been used to solve difficult problems in many areas of science and engineering such as bioscience, genetics, and geology. This is due to concurrency, reliability, computational power that the parallel computer provides. A parallel computer is multiple processors that work together to solve computational problems (Foster, 1995; Grama, et al., 2003).

Parallel computers are categorized according to memory architecture into shared memory and distributed memory. In the shared memory architecture, also known as multi-processors, the memory is physically shared between various processors and processors communicate with each other via the shared memory (Grama, et al., 2003). However, in the distributed memory architecture, also known as multicomputers, different parts of the memory are physically associated with different processing units, and processors communicate with each other by exchanging messages through an interconnection network (Foster, 1995; Grama, et al., 2003).

An interconnection network provides a way for data transfer between nodes (processors). Interconnection networks can be separated into two types: static and dynamic. In dynamic networks, also called indirect networks, communication links are configured dynamically by switches to form paths between nodes (i.e., processors) (Grama, et al., 2003); an example of a dynamic network includes the bus-based network (Ferreira, et al., 1994). In static networks, also called direct networks, there is a point-to-point communication link (direct communication link) between nodes (i.e. processors); an example of static networks is the mesh network (Adve and Vernon, 1994).

Direct networks have been used in many large-scale multicomputers. This is due to their scalability: scalability means that it is able to simply scaled up by adding new nodes and channels that depend on the predefined network structure. Direct networks can also exploit communication locality (nearest neighbor communication) exhibited by many real-world applications (Bani-Mohammad, 2008). The mesh network is the most popular network used in multicomputer systems. This is because of many features that the mesh

network has, such as simplicity, regularity, ease of implementation and high scalability (Babbar and Krueger, 1994; Yoo and Das, 2002; Grama, et al., 2003). In two-dimensional meshes, each node (except the nodes at the edges) is connected to four neighbors by direct communication links. Matrix computations and image processing map very naturally onto a 2D mesh. The three-dimensional mesh is a generalization of the 2D mesh. Both 2D and 3D meshes have been adopted in many commercial and experimental multi-computers (Foster, 1995; Grama, et al., 2003). The Delta Touchstone (Intel Corporation, 1991) is an example of 2D mesh-connected multi-computers, and the IBM blueGene/L (Blumrich, et al., 2003) is an example of 3D mesh-connected multi-computers. Figure 1.1 shows an example of a 4×4 2D mesh, in which the allocated processors are indicated by black squares and free processors are indicated by white squares.

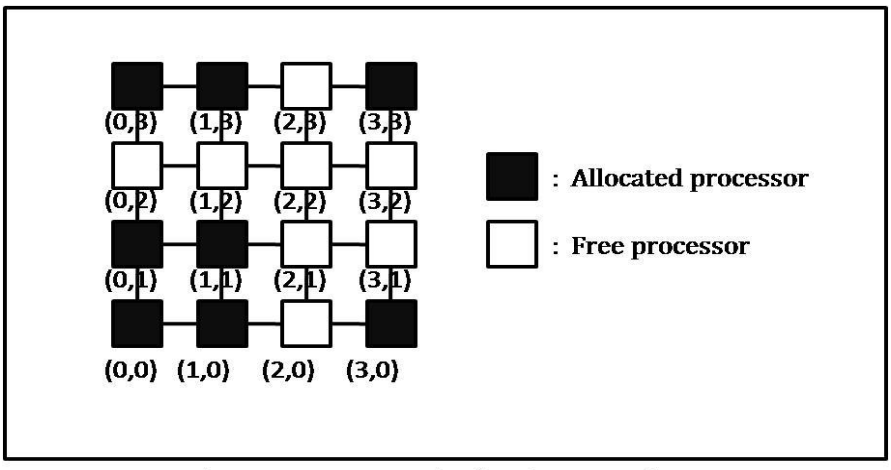

Figure 1.1: An Example of a 4*4 2D mesh

1.1 Processor Allocation

The resource management policy is a critical issue in designing the multicomputer operating system (OS) that supports multiple users. An efficient resource management policy is critical to improving system performance (Yoo and Das, 2001; Yoo and Das, 2002). Processor management system mainly comprised of two components: processor allocation and job scheduling. Processor allocation is responsible for assigning the desired number of processors to incoming jobs and job scheduling are responsible for deciding the order in which jobs are chosen for execution (Babbar and Krueger, 1994; Yoo and Das, 2002; Bani-Mohammad, 2008).

Incoming job in the mesh-connected multicomputer system determines the size of the needed sub-mesh before entering the system queue. The job scheduler chooses the next job for execution depending on the underlying scheduling policy. The processor allocator then tries to find free sub-mesh for the chosen job. If the processor allocator failed to find the required sub-mesh because there are no free processors or there are already awaiting jobs in the system, then the job joins the waiting jobs queue until an allocated job finished its execution and released a sub-mesh. When the processor allocator finds a free sub-mesh for the selected job then the job holds these processors in this sub-mesh until till it completes its execution. When the execution is finished, all the allocated processors are freed and can be used by other jobs (Chang and Mohapatra, 1998; Yoo and Das, 2002; Bani-Mohammad, 2008).

Processor allocation algorithms are responsible for finding the sub-meshes for incoming job requests. This capability is called the sub-mesh recognition capability. A processor allocation algorithm is considered to have a complete sub-mesh recognition capability if it can always determine a free sub-mesh for an incoming job if one is available. Having a complete sub-mesh recognition capability improves the performance of the system, but instead, it could increase the complexity and the allocation overhead (Yoo and Das, 2002).

Processor allocation strategies composed of two approaches: contiguous and noncontiguous. In contiguous allocation strategies, parallel jobs are allocated to distinct contiguous sub-meshes of physically adjacent processors and the sub-meshes have the same topology as the underlying multicomputer network. These strategies aim to eliminate contention between the messages of various jobs executing on the system and reduce inter-processor communication delays (Zhu, 1992; Yoo and Das, 2002; Ababneh, et al., 2010). Contiguous allocation strategies can lead to high processor fragmentation because of the contiguity condition (Lo, et al., 1997; Ababneh, et al., 2010). This fragmentation results in degrading of the system performance with regard to job turnaround time (i.e., the time that the job spends in the system from arrival to departure) and mean system utilization (i.e., the percentage of processors that are utilized over a given time) (ProcSimity User's Manual, 1997).

There are two types of Processor fragmentation: internal and external. Internal fragmentation happened when more processors are allocated to a job than it needs. This because of the restricted shape of sub-meshes allocation which results in extra processors to be allocated to a requested job and these processors are wasted and not used. External fragmentation happened when there are free processors enough in number to satisfy job request, but they cannot be allocated because they are not contiguous (Zhu, 1992; Lo, et al., 1997; Chang and Mohapatra, 1998; Seo, 2005;).

Figure 1.2 shows an example of internal and external fragmentation in the contiguous allocation algorithm. Figure 1.2 (a) shows an incoming job that requests 3×2 sub-mesh of processors, by using two dimensional buddy strategy that restrict to allocate a power of two contiguous sub-mesh then 16 processors are allocated resulting in 0.625 (10/16) internal fragmentation. Figure 1.2 (b) shows an incoming job that request 2×2 sub-mesh of processors, and the algorithm fails to allocate the requested sub-mesh because the available processors are not contiguous resulting in external fragmentation.

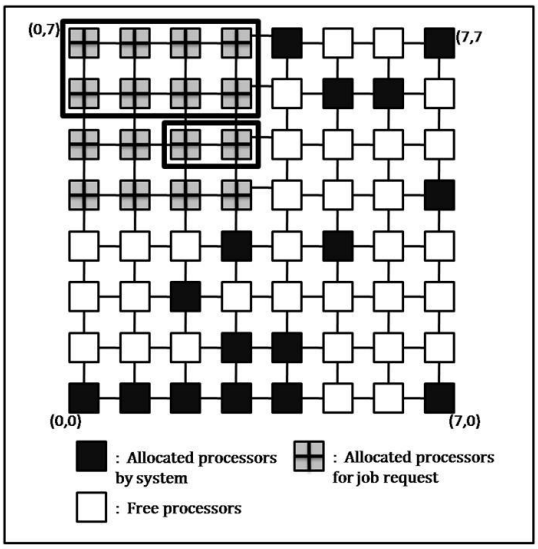

Figure 1.2: (a) An Example of internal fragmentation

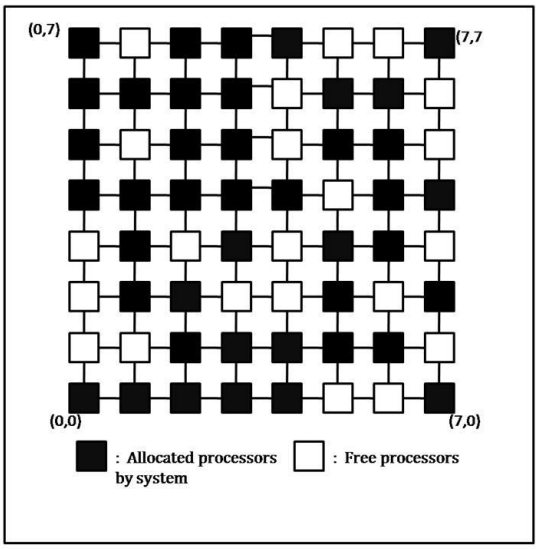

Figure 1.2: (b) An Example of external fragmentation.

Examples of contiguous allocation strategies for 2D mesh-connected multicomputers include the Two Dimensional Buddy System (2DBS)(Li and Cheng,1991), Frame Sliding (FS) (Chuang and Tzeng, 1994) and First Fit (FF) and Best Fit (BF) (Zhu, 1992).

Noncontiguous allocation strategies are suggested to reduce the fragmentation problem that occurs in contiguous allocation strategies. In noncontiguous allocation, a job can be allocated to multiple disjoint smaller sub-meshes instead of waiting for one sub-mesh of the requested size and shape to be available. Lifting the contiguity condition in noncontiguous allocation reduces processor fragmentation and increases processor utilization but instead increases the communication overhead due to the inter-process contention produced by messages from different jobs and long distances between the communicating nodes (Lo, et al., 1997; Chang and Mohapatra, 1998; Bani-Mohammad, et al., 2007). Examples of noncontiguous allocation strategies for 2D mesh-connected multicomputers include Random (Lo, et al., 1997), Paging (Lo, et al., 1997), Multiple Buddy Strategy (MBS) (Lo, et al., 1997).

A good processor allocation strategy is preferred to be hybrid between contiguous and noncontiguous allocation strategies. The processor allocation strategy should be able to divide the job while maintaining a high degree of contiguity between the allocated processors. The processor allocation strategy is responsible for recognizing and allocating the available sub-meshes in such way that minimizes the communication overhead and hence improves the overall system performance (Lo, et al., 1997; Bani-Mohammad, et al., 2007).

1.2 Motivation and Contribution

Various previous studies (Zhu, 1992; Lo, et al., 1997; Ababneh, 2008, Bani-Mohammad, et al., 2009 Ababneh, et al., 2010; Bani-Mohammad, et al., 2013) in processor allocation indicate that noncontiguous allocation strategies dramatically better than contiguous allocation strategies with regard to mean system utilization and average turnaround time. This is because the noncontiguous processor allocation strategies have solved the problem of fragmentation that exist in contiguous processor allocation strategies. In contiguous allocation, the parallel job must have the same topology as the multicomputer (Li and Cheng, 1991; Zhu, 1992; Lo, et al., 1997). Lifting the contiguity condition which is done in noncontiguous allocation allows to allocate several dispersed sub-meshes to a requested job (Mache and Lo, 1997) that results in improving the system performance with regard to

system utilization by up to 78% for common workloads (Wan, et al., 1996; Lo, et al., 1997) but this instead increases the message contention inside the network because messages that come from various jobs can collide together via competing for communication links and messages may traverse longer distances.

There are two types of contention (Min and Mutka, 1994): internal contention and external contention. Internal contention exists when two or more routing paths for the same job try to use a physical channel at the same time. The internal contention is an inherent property of each job, and it can exist in both contiguous and noncontiguous allocation strategies. External contention exists when two or more routing paths of different jobs try to use the same physical channel at the same time. External contention exists only in the noncontiguous allocation strategies. When using noncontiguous allocation in a system with wormhole routing technique, the external contention increases the delay of the communication time (Min and Mutka, 1994). Always, there is a tradeoff between the processor utilization due to the fragmentation problem and the jobs turnaround time due to the network contention (Min and Mutka, 1994; Moore and Lionel, 1996).

Two-factor play a role in contention, the switching technology and communication pattern between the allocated processors (Min and Mutka, 1994). The contention can be ignored if the software latency (i.e., the latency at sender and receiver for processing the message) is high or when the message size is small (Moore and Lionel, 1996). The message contention between the messages of different jobs results in increasing the communication overhead. This, in turn, increases the delay and weakness the gain of improved system utilization that results in degrading the system performance with regard to jobs turnaround time (Min and Mutka, 1994; Mache and Lo, 1997). To improve the performance of the noncontiguous allocation strategies, we should choose an allocation strategy that causes minimal message contention where the geometric location of the allocated sub-meshes in the mesh system plays a significant role in the interference between jobs' messages (Mache and Lo, 1997).

The existing noncontiguous allocation strategies (Lo, et al., 1997; Mache, et al., 1997; Chang and Mohapatra, 1998; Bani-Mohammad, et al., 2007; Ababneh, 2008; Bani-Mohammad, et al., 2012) use different techniques to determine and allocate free sub-meshes in the mesh

system. Noncontiguous allocation strategies focus on maintaining a high degree of contiguity among the processors in the allocated sub-meshes instead of reducing message contention in the sub-meshes that are allocated to different jobs.

Many research studies have been investigated the processor allocation in multicomputers, especially those based on mesh network (Li and Cheng, 1991; Zhu, 1992; Chuang and Tzeng, 1994; Lo, et al, 1997; Chang and Mohapatra, 1998; Ababneh, 2001; Bani-Mohammad, et al., 2007; Ababneh, 2008; Ababneh, et al., 2010; Bani-Mohammad, et al., 2012). But to the best of our knowledge, there is no study that considers the effect of the Near Neighbor, Ring, All to all, Divide and Conquer Binomial Tree (DQBT), Fast Fourier Transform (FFT), One to All, All to One, and Random communication patterns on the performance of contiguous and noncontiguous processor allocation in multicomputer, especially
when each job does exactly one iteration of the given communication pattern. The communication pattern used can have a great impact on the performance of contiguous and noncontiguous processor allocation in multicomputer (Bani-Mohammad, et al., 2013).

In this thesis, the performance of the most famous contiguous allocation strategies (First Fit, Best Fit) and most famous noncontiguous allocation strategies (GABL, Paging, MBS) for 2D mesh multi-computers is re-visited considering several important communication patterns. Which are one-to-all (ProcSimity Manual, 1997), near neighbor (Bani-Mohammad and Ababneh, 2013), random (ProcSimity Manual, 1997), all-to-all (ProcSimity Manual, 1997; Lo, et al., 1997), ring (ProcSimity Manual, 1997; Lo, et al., 1997), Divide and Conquer Binomial Tree (DQBT)(Lo, et al., 1996 ; Lo, et al., 1997; Valero-Garcia, et al.,1997; Grama, et al.,2003), Fast Fourier Transform (FFT) (James W. Cooley and John W. Tukey, 1964; Lo, et al., 1997; Grama, et al.,2003; Chan, et al., 2008), all-to-one (Grama, et al., 2003)

. Those communication patterns have been chosen because they have been used in related works (Suzaki, et al., 1996; Mache, et al., 1997; Lo, et al., 1997; Bani-Mohammad, et al., 2007; Ababneh, 2008; Bani-Mohammad, et al., 2012; Bani-Mohammad and Ababneh, 2013) and because they are common, and they cover many communications patterns used very frequently by highly parallel applications (Lo, et al., 1997). Two distributions were considered for generation job side lengths, they are the uniform and uniform-decreasing

distributions. Similar distributions have been used in the literature (Lo et al., 1997; Chang and Mohapatra 1998; Chiu and Chen, 1999, Bani-Mohammad and Ababneh, 2013). Wide simulation experiments have been conducted to compare the performance of contiguous allocation with that of noncontiguous allocation with regard to average turnaround time and mean system utilization using the ProcSimity simulator.

The simulation results have shown that in near neighbor, FFT and DQBT communication patterns, the performance of contiguous allocation strategies (FF and BF) dramatically better than all noncontiguous allocation strategies (Paging(0), MBS and GABL) with regard to average turnaround; except for MBS in DQBT communication pattern, the performance of MBS is very close to that of FF and BF. These results prove that the taken fact that say that noncontiguous allocation strategies always dramatically better than contiguous allocation strategies with regard to average turnaround time is not always true. Also, the simulation results for average turnaround time have shown that in one-to-all, random, ring and all-to-one communication patterns,

the performance of the noncontiguous allocation strategies (Paging(0), MBS and GABL) dramatically better than that of the contiguous allocation strategies (FF and BF) with regard to average turnaround time. For all-to-all communication pattern, the simulation results have shown that the performance of contiguous allocation strategies (FF and BF) is better than that of the MBS noncontiguous allocation but the performance of GABL and Paging(0) is better than that of FF, BF, and MBS.

The results for system utilization have shown that in all communication patterns that are considered in this research work, the noncontiguous allocation strategies dramatically better than contiguous allocation strategies with regard to mean system utilization.

1.3 Structure for the Thesis

The rest of the thesis is organized as follows:

Chapter 2 explains the most famous contiguous and noncontiguous allocation strategies that are considered in this thesis. Also, it gives some preliminaries needed for understanding the following chapters and provides a list of assumptions used in this thesis.

Chapter 3 this chapter explains the method of study used in this thesis and analyzes and discusses the results of the simulation experiments and compares the performance of contiguous and noncontiguous allocation strategies.

Chapter 4 summarizes the major results obtained in this thesis and outline possible directions to continue this work in the future.

Chapter Two

Background and Preliminaries

The main aim of this chapter is to explain some of the most famous contiguous and noncontiguous allocation strategies that have been suggested in the literature (Li and Cheng, 1991; Zhu, 1992; Chuang and Tzeng, 1994; Lo, et al, 1997; Bani-Mohammad, et al., 2007) for 2D mesh-connected multicomputers. The chapter also explains the system model assumed in this thesis. This chapter gives a background that helps to understand the following chapters.

2.1 Related Allocation strategies

This section explains some of the existing contiguous and noncontiguous allocation strategies that have been suggested for 2D mesh-connected multicomputers.

2.1.1 Contiguous allocation strategies

There are many contiguous allocation strategies that have been suggested for 2D mesh-connected multicomputers. Most of the contiguous allocation strategies have focused on reducing fragmentation caused by contiguous allocation. High processor fragmentation problem can impact the system performance (Zhu, 1992; Ababneh, et al., 2010). Below we explains some of the most famous strategies.

Two Dimensional Buddy Strategy (2DBS): The 2DBS allocation (Li and Cheng, 1991) is applied to square meshes with a side length of the power two. The requested job is allocated to sub-mesh that is also squared with a side length that is rounded up to the nearest power of two. When a job requests a sub-mesh of size x × y, such that x ≤ y, the 2DBS allocates a sub-mesh of size s× , where = 2⌈log2 max(,)⌉. For example, if a job requests 2×4 sub-mesh of processors, it is allocated a square sub-mesh of processors with a size 4×4, that result in 8 idle processors and an internal fragmentation of 50% as shown in Figure 2.1. The 2DBS suffers from internal and external processor fragmentation due to the side length condition and lacks complete sub-mesh recognition capability. The 2DBS can only be used to square meshes (Zhu, 1992; Lo, et al., 1997; Chang and Mohapatra, 1998).

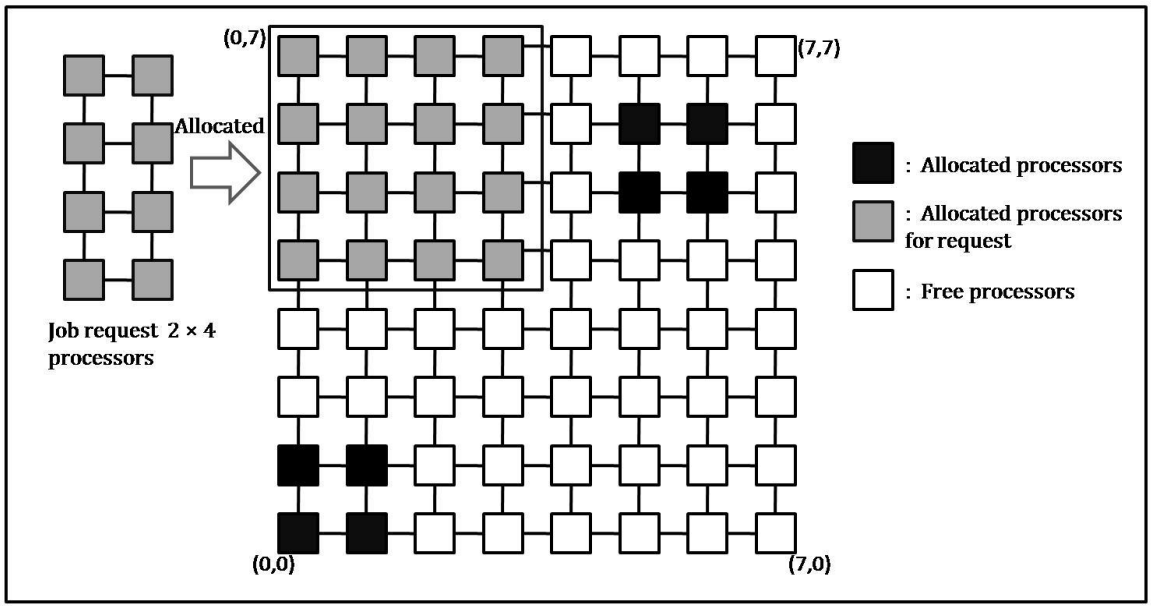

Figure 2.1: an allocation using the 2DBS

Frame Sliding (FS) Strategy: The frame sliding strategy (Chuang and Tzeng, 1994) is used to solve the fragmentation problem occurs in 2DBS. The FS strategy applies to any size of mesh system and any shape of a sub-mesh request which means that there is no internal fragmentation. The FS strategy slides a frame of a requested sub-mesh through a bit array that represents free and allocated processors to finds an available sub-mesh. The FS strategy starts to examine the first candidate (frame) at the lower leftmost free processor and slides the candidate frame vertically or horizontally equivalent to width or height of the requested sub-mesh, respectively. The searching process stops when an available frame is found or when all candidate frames are exhausted. The FS suffers from large external fragmentation and it cannot recognize all available sub-meshes, which means that even if there is a free sub-mesh the FS fails to allocate it because of the jumps by width and height of the job's request (Zhu, 1992; Lo, et al., 1997; Chang and Mohapatra, 1998). Figure 2.2 gives an example of such situation. Figure 2.2 shows a 6 × 5 mesh system and an incoming request of 3 × 2 sub-mesh.

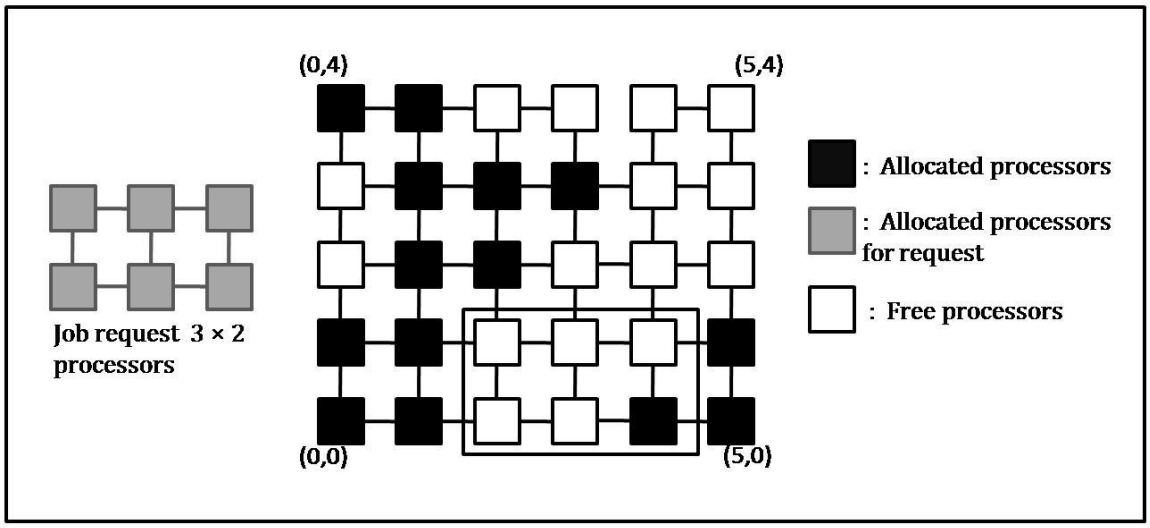

Figure 2.2: An allocation using the frame sliding strategy

First Fit (FF) and Best Fit (BF) Strategies: The FF and BF strategies (Zhu, 1992) use a bit array for scanning of free processors. Both FF and BF strategies solve the problem of losing an existing possible allocation occurred in previous strategies. In FF, the first found sub-mesh with a sufficient number of processors is allocated, whereas in BF, a sub-mesh with the largest number of busy neighbors (processors) and smallest number of free neighbors (processors) is allocated. Both FF and BF strategies can discover all large-enough free sub-meshes but haven't complete sub-mesh recognition capability because they do not consider switching the requested shape orientation. The BF strategy attempts to reduce the probability of fragmentation. Both FF and BF strategies suffer from significant external fragmentation (Zhu, 1992; Lo, et al., 1997). Figure 2.3 shows the allocation of a job request for a 2×2 sub-mesh using FF and BF.

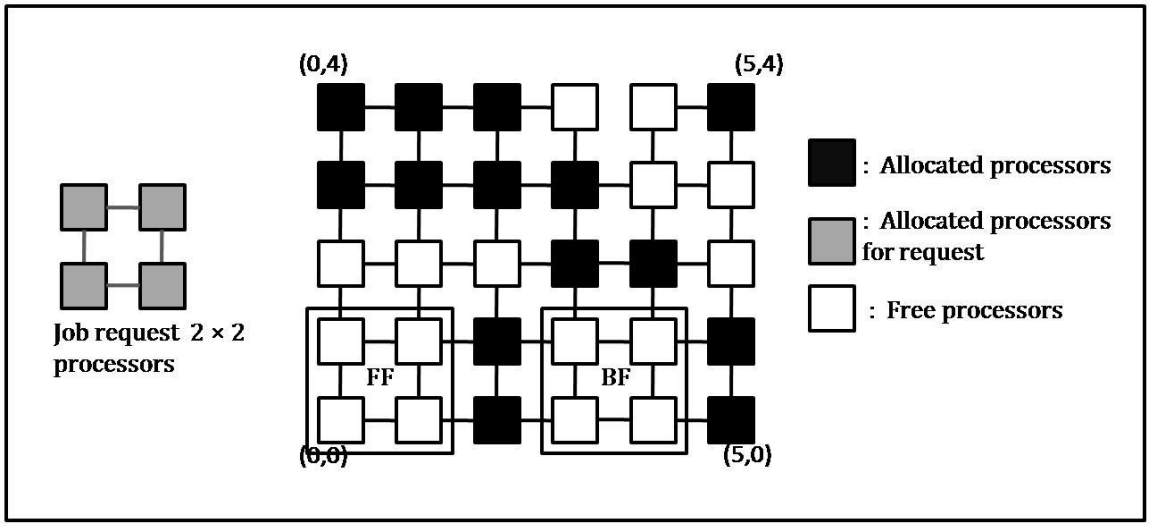

Figure 2.3: An allocation using First Fit and Best Fit strategies

2.1.2 Noncontiguous Allocation Strategies

Little improvement in the performance can be gained by the refinements of contiguous allocation strategies (Lo, et al., 1997; Chang and Mohapatra, 1998). The communication latency becomes less sensitive to the distance between the communicating processors because of the wormhole routing (Ni and McKinley, 1993) and faster switching technique, and this makes allocating a job to noncontiguous processors is feasible(Lo, et al., 1997; Chang and Mohapatra, 1998; Bani-Mohammad, et al., 2007). Noncontiguous allocation permits a job to be executed if there are enough number of free processors in the mesh. Many noncontiguous allocation strategies have been suggested for 2D mesh multicomputers (Lo, et al., 1997; Chang and Mohapatra, 1998; Bani-Mohammad, et al., 2007). Some of the most famous noncontiguous allocation strategies that have been considered in the thesis are explained below.

Random allocation strategy: This strategy (Lo, et al., 1997) is a simple strategy in which a job request for a number of processors is satisfied with a randomly chosen number of processors. It removes both internal and external fragmentations because an exact number of processors are allocated to the job. There is no contiguity enforced by this strategy which results in much communication interference between jobs (Lo, et al, 1997).

Paging strategy: In paging strategy (Lo, et al., 1997), the whole mesh is divided into square pages with equal side lengths of 2 _ ; where _ is a positive integer. The page is the main unit of allocation. The term indexing scheme means in which order are the pages scanned. Several indexing schemes are used for indexing the pages (row-major, shuffled row-major, snake-like, and shuffled snake-like indexing) as shown in Figure 2.4. The Paging algorithm is represented as Pagingindexing_scheme (page_size). For a job request for k processors is allocated ⌈⁄22 page_size⌉ pages. This is done by scanning the free page list according to the given indexing scheme. Indexing schemes maintained some degree of contiguity among allocated pages. The contiguity can be increased by increasing the page size but increasing the page size results in much internal fragmentation. For paging with page size equal zero, both internal and external fragmentations are removed and for page size greater than or equal to one, internal fragmentation is occurred (Lo, et al., 1997). This strategy is good, but it is still cannot allocate a job contiguously although a one sufficient sub-mesh is free in the mesh system and such situation is shown in Figure 2.4. Figure 2.4 shows an example of Paging row-major (0).

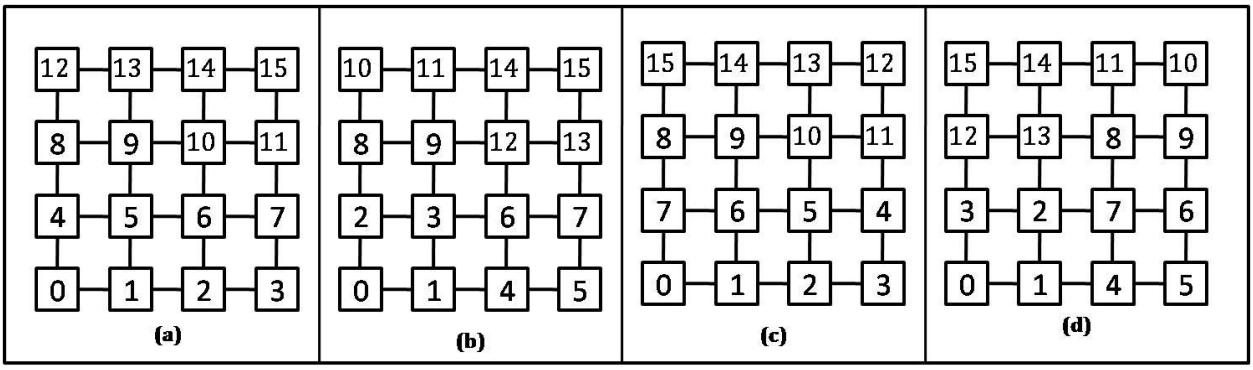

Figure 2.4: Paging(0) using different indexing schemes: (a) Row-major indexing, (b) Shuffled row-major, (c) snake-like indexing, and (d) shuffled snake-like indexing.

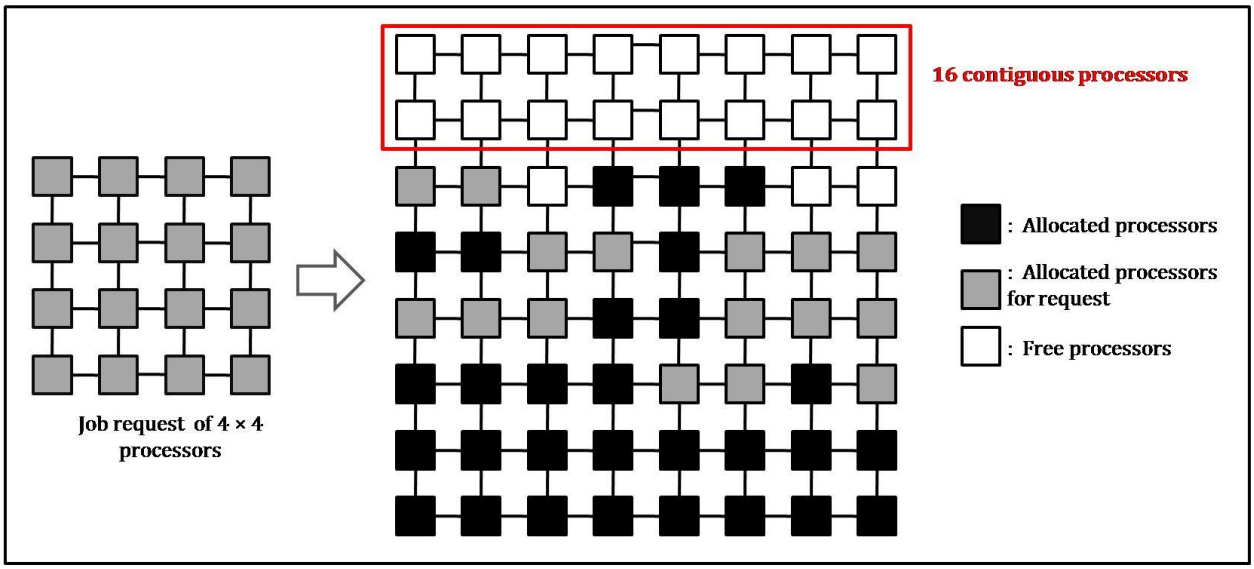

Figure 2.5: An allocation using Paging Row_major (0) strategy.

Multiple Buddy Strategy (MBS): The MBS (Lo, et al., 1997) is an extension of the 2D buddy strategy (2DBS). MBS removes both internal and external fragmentation problems that occur in 2DBS by allowing individual contiguous blocks to be allocated to a job noncontiguously. The whole mesh in this strategy is divided into distinct square sub-meshes with side lengths equal to the powers of two at the initialization stage. At factoring stage, the requested number of processors for an incoming job is factorized into a base of four representation of ∑log04 p di × (2i × 2i), where 0 ≤ di ≤ 3. Then, the job request is allocated depending on the factorized number where di free processor blocks of size equal to 2i × 2i are required for every term i. When the required block is unavailable then the MBS searches for a larger block and divides it into buddies and stop when it produces blocks of the required size. If it fails then the requested block is broken into four requests for smaller blocks and the searching process is repeated again (Lo, et al., 1997, Ababneh, et al., 2010). The restriction to base four blocks of allocation results in failure in allocating a free sub-mesh contiguously to a requested job. Figure 2.6 shows an example of MBS allocation.

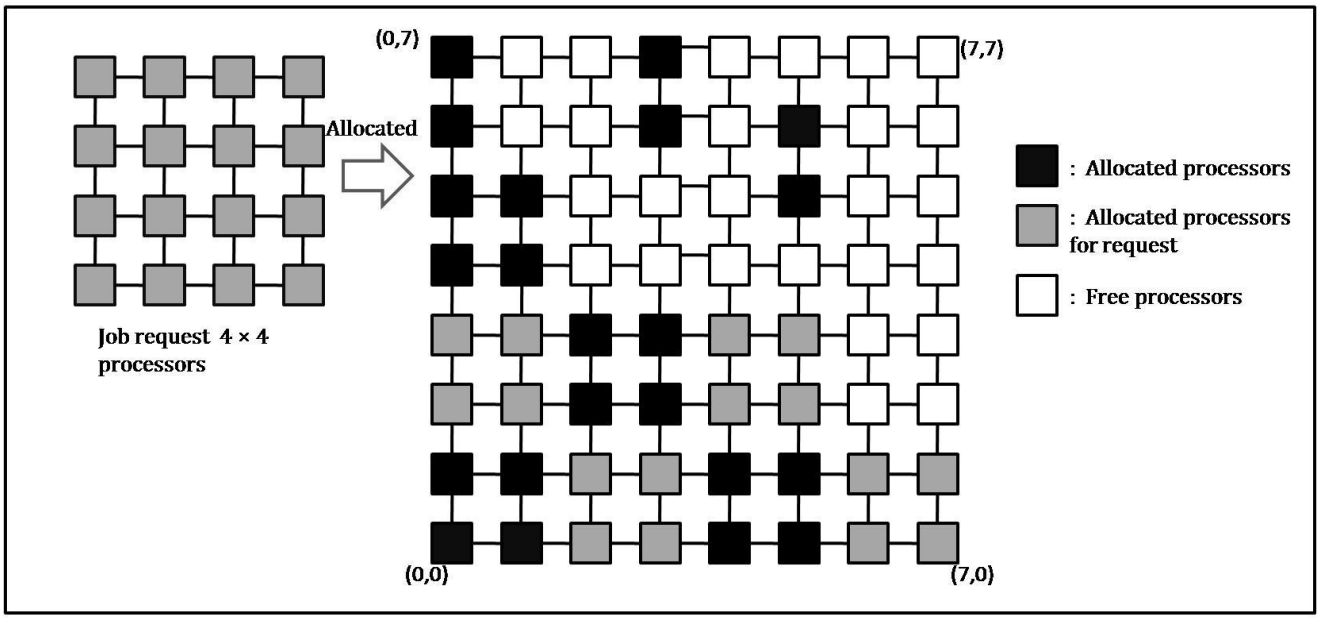

Figure 2.6: An allocation using MBS allocation strategy.

Greedy Available Busy List (GABL): In GABL strategy (Bani-Mohammad, et al., 2007, AlHarafsheh, 2016, Alsardia, 2017), upon the selection of parallel job for allocation, a sub-mesh appropriate for the whole job is searched for. If requested sub-mesh exists then it is allocated to the job and the allocation happens but if it does not exist then the largest free sub-mesh that can fit inside the request job size is allocated. After that, the largest free sub-mesh with side lengths that do not exceed the corresponding side lengths of the previously allocated sub-mesh is searched for, and this allocation must not result in allocating more processors than the desired size

. The strategy repeats the last step till the desired number of processors is allocated to the job. All the allocated sub-meshes are stored in a busy list. Every element in the busy list includes the id of the job that the sub-mesh is allocated to. The busy list is updated after each allocation and de-allocation operation. An efficient approach proposed in (Chiu and Chen, 1999) are used in GABL to detect free sub-meshes with low allocation overhead. The goal of the GABL strategy is to maintain a high degree of contiguity among processors allocated to the job and this decreases the number of sub-meshes allocated to a job and minimizes the distance traversed by messages, which then reduces message contention inside the network (Bani-Mohammad, et al., 2007). However, GABL still may allocate sub-meshes that are far apart from each other.

Figure 2.8 shows an example of how GABL allocates a job request. Suppose a job request of size 4×3 arrives to the system. GABL always tries to allocate any job request contiguously. GABL searches for a free sub-mesh of the desired size (4×3) in the mesh. GABL failed to find a contiguous sub-mesh of size 4×3. GABL then starts searching again by subtracting one from the maximum side length of the desired sub-mesh, and this step is repeated till it finds a suitable free sub-mesh. In this case, a 2×3 available sub-mesh of processors with the coordinates (6,0,7,2) is found, where the first two coordinates specify the lower left corner of the sub-mesh and the last two coordinates specify the upper right corner of the sub-mesh. Then it continues to allocate another sub-mesh (3,0,5,1) by rotating the current request to be 3×2 provided that the number of allocated processors does not exceed the original request.

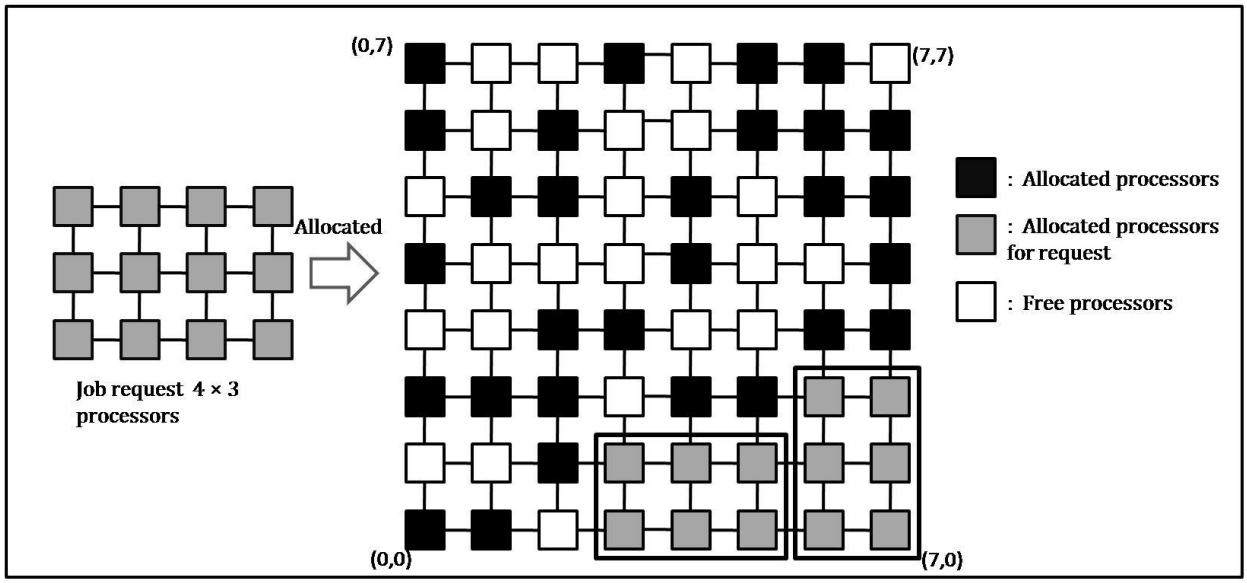

Figure 2.7: an example of allocation using GABL allocation strategy.

2.2 Switching Method

The switching method is responsible for specifying the way to transmit a message as they travel across intermediate nodes. Switching is used at the router and composed of the receipt of a message, specifying the suitable output node, and then transmitting the message across this node (Ni and McKinley, 1993; Lo, et al., 1997; Bani-Mohammad, 2008, Alsardia, 2017). The switching technique has a significant impact on the communication latency in the direct network multicomputer systems. This section briefly describes the

three most important switching techniques for multicomputer networks: Store-and-forward (Grama, et al., 2003), Virtual cut-through (Drewes, 1996), and Wormhole switching (Ni and McKinley, 1993; Mohapatra, 1998).

Store-and-forward switching: In store-and-forward switching, the message is partitioned into fixed-length packets in which those packets are routed from source to destination. Each packet includes a header that hold sits destination address. Each intermediate node keeps the entire packet before forwarding it to the next node in its path to the destination node. The store-and-forward switching has two main disadvantages: a large buffer space is needed to hold the whole packets at each intermediate node and the time to transmit a packet from source node to destination node is proportional to the distance between those nodes (Ni and McKinley, 1993; Mohapatra, 1998).

Virtual cut-through switching: Virtual cut-through (Drewes, 1996) has been proposed as an improvement for the store-and-forward switching so as to reduce the time spent in transmitting data and to reduce the space overhead for storing the whole packet at each intermediate node. In virtual cut-through switching, the header included in the packet which includes routing information is checked upon coming at an intermediate node. If the next wanted channel is busy then the packet is entirely stored at the intermediate node; otherwise, it is transmitted to the next node without buffering. This reduces the effect of the distance between the communicating nodes on the communication latency. Each node must provide a very large buffer space for all blocked packets passing across it because multiple packets may become blocked at the same time. The requirement of high buffer space results in an increase in the implementation cost (Ni and McKinley, 1993; Mohapatra, 1998).

Wormhole switching: The Wormhole switching (Duato, et al., 1997) is suggested to solve the needed for large buffer spaces and to reduce the sensitivity of the communication latency to the distance between the communication nodes that occur in virtual cut-through switching. In wormhole switching, a packet is partitioned into a sequence of fixed-size units, and those units called flits (flow control unit), which is the smallest unit of data transmission. The header flit controls the route by using the contained routing information

and start establishing the path across the network and the remaining data flits contiguously follow the header over the same path in a pipelined fashion. If the header flit blocked because of the resource contention then all the remaining data flits blocked and keeping all allocated links and buffers at the intermediate nodes occupied and at each intermediate node, there is only one flit. This blocking prevents other packets from using these channels, which leads to a deadlock; packets wait for each other in a cycle without being able to move forward anymore. A critical issue in wormhole switching is deadlock prevention. Deadlock prevention can be achieved using a suitable choice for routing function (Ni and McKinley, 1993; Mohapatra, 1998).

Because of the pipelines during transmission in the wormhole routing, the wormhole routing can perform well even in high-diameter networks, such the mesh (Min, 2003). The iWARP (Peterson, et al., 1991) and the MIT J-machine (Noakes, et al., 1993) experimental machines have used wormhole switching. The Intel Paragon (Intel Corporation, 1991), the IBM blueGene/L (Blumrich, et al., 2003), and the Cray XT3 (Cray, 2005) commercial machines have used wormhole switching. In this thesis, wormhole switching has been used when examining the performance of the allocation strategies. The Wormhole switching has been used since it has been used in the previous allocation strategies (Lo, et al., 1997; Mache, et al., 1997; Bani-Mohammad, et al., 2007; Ababneh, et al., 2010).

2.2.1 Routing Algorithm

An efficient algorithm to route a message from its source to its destination is critical to the performance of parallel multicomputers that use direct networks. A direct network topology must let any node to send packets to every other node. Mesh network, which is a direct network, provides many physical paths for routing a packet among any two nodes. A routing algorithm specifies the path that a packet takes from its source node to its destination node. Routing algorithms can be classified into two types: deterministic and adaptive. In deterministic routing, the unique path of the packet is completely specified by the source and destination; intermediate nodes cannot redirect packets to any alternative paths. In adaptive routing, the path of the packet is specified based on the current state of the network such as the presences of failure or congestion and accordingly routes the

packet along alternative paths. Routing algorithm must handle deadlock if the dead lock occurs; deadlock exists when no packet can reach its destination due to the busy channels and buffers (Ni and McKinley, 1993; Mohapatra, 1998; Grama, et al., 2003).

Dimension-order routing is an example of deterministic routing technique where the sent packet is routed in one dimension at a time until it reaches the proper coordinate then it routed in the next dimension towards the destination. The Dimension-order routing in two-dimensional mesh networks is called routing, and it provides deadlock-free routing because packets' path cannot form a deadlock cycle. The packet in routing goes along the dimension (width of mesh) until it reaches the column of the destination node then it goes along the dimension (height of mesh) until it reaches the destination node (Ni and McKinley, 1993; Grama, et al., 2003). Figure 2.7 shows an example of routing among source node and the destination node in an × 2D mesh-connected network. in this thesis, the routing is used when studying the performance of the allocation strategies since it has been used in the previous allocation strategies (Lo, et al., 1997; Bani-Mohammad, et al., 2007;Ababneh, 2008;Ababneh, et al., 2010).

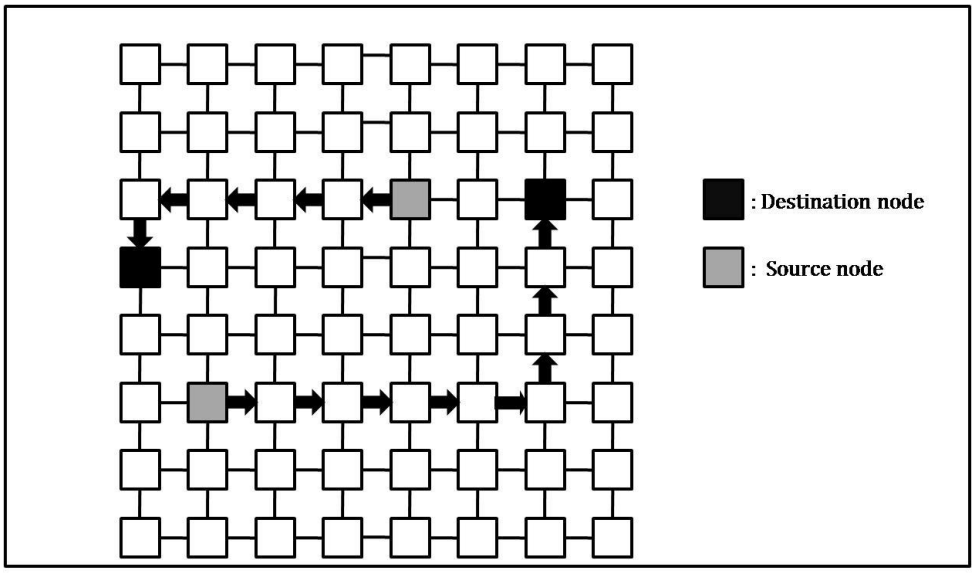

Figure 2.7: Dimension-ordered (XY) routing in an 8×8 2D mesh-connected network

2.2.2 Communication Patterns

The allocated processors to a parallel job often swapping messages together based on a specified communication pattern (Lo, et al., 1997). An important parameter to measure when evaluating noncontiguous allocation is message contention that comes from swapping messages and its impact on overall system performance. In this thesis, eight communication patterns have been considered to assess the performance of contiguous and noncontiguous allocation algorithms. First, one-to-all communication pattern (ProcSimity Manual, 1997, Lo, et al., 1997, AlHarafsheh, 2016), where a randomly chosen processor transmits a message to each other processors allocated to the same job. Second, random communication pattern (ProcSimity Manual, 1997), where a randomly chosen processor transmits messages to randomly chosen destination within a group of processors allocated to the same job. Third, all-to-all communication pattern (ProcSimity Manual, 1997, Lo, et al., 1997, AlHarafsheh, 2016), where every processor transmits a message to all other processors allocated to the same job. One-to-all, all-to-all and random communication patterns have been used since they have been used in related works (Suzaki, et al., 1996; Mache, et al.,1997; Lo, et al., 1997; Bani-Mohammad, et al., 2007; Ababneh, 2008; Bani-Mohammad, et al., 2012; Bani-Mohammad and Ababneh, 2013, Alsardia, 2017) and because they are common, and they cover many communications patterns used very frequently by highly parallel applications (Lo, et al., 1997). Forth, near neighbor communication pattern (Bani-Mohammad and Ababneh, 2013, AlHarafsheh, 2016, Alsardia, 2017), where all the processor that are allocated to a job are mapped to a virtual two-dimensional array. The size of the 2D array is equal to the job's allocation request. Each processor communicates with its virtual neighbors. Near neighbor communication has been used since it is a common communication pattern for simulations of physical phenomena such as heat and wave propagation (Bani-Mohammad and Ababneh, 2013).

Fifth, ring communication pattern (ProcSimity Manual, 1997; Lo, et al., 1997), where each processor allocated to a job transmits a message to its successor in the linear array. The successor of the last processor is the first one. Ring communication has been used since it is

common in matrix computations. Sixth, all-to-one communication pattern (Grama, et al., 2003), where all processors allocated to the same job transmits a message to a randomly chosen processor. All-to-one has been used since it is used in several important parallel algorithms including matrix-vector multiplication, shortest paths, and vector inner product. Seventh, Fast Fourier Transform (FFT) communication pattern (James W. Cooley and John W. Tukey, 1964; Lo, et al., 1997; Grama, et al., 2003; Chan, et al., 2008), which is an efficient algorithm that is used to compute the Discrete Fourier Transform (DFT) and its inverse. FFT consists of two transforms, the forward and a backward transform. The forward operation transforms a function f(x) in real space X to a function F(k) in Fourier space K while the backward transform does the reverse operation that transforms F(k) in Fourier space K to f(x) in real space X. FFT has been used because it has been one of the most popular and widely used numerical methods in many areas of scientific computing, such as digital signal processing and solving linear partial differential equations (Grama, et al.,2003).

Eighth, Divide and Conquer Binomial Tree (DQBT)(Lo, et al., 1996; Lo, et al., 1997; Valero-Garcia, et al., 1997; Grama, et al., 2003), where a message is transmit to all processors allocated to the same job using a binomial broadcast tree. Because of these restrictions associated with this pattern, the widths and lengths of jobs are truncated to the nearest power of two. A DQBT algorithm has two stages. In the first stage, which is called the division stage, the original problem is decomposed into n subproblems. Each subproblem is recursively decomposed into n subproblems till the subproblems are small enough to be solved by a processor without any further decomposition. During this stage, each processor receives a message from its parent exactly once but may send messages multiple times. In the second stage, which is called the combine stage, the result of subproblems is combined to produce the final result, and during this stage, the message traffic is in the opposite direction of that of the divide stage; each processor may receive many times but sends exactly once. The patterns of data flow in the two stages (divide and combine) are identical except for the direction. DQBT has been used because it is used in many parallel applications (Valero-Garcia, et al., 1997), such as sorting algorithms and matrix multiplication.

2.3 Assumptions

In the subsequent chapters, wide simulation experiments will be presented to evaluate the allocation strategies. In this study, we make the following assumptions, which have been mostly used in the literature (Zhu, 1992; Babbar and Krueger, 1994; Suzaki, et al., 1996; Mache, et al., 1997; Chang and Mohapatra, 1998; Yoo and Das, 2002; Seo, 2005; Bani-Mohammad, et al., 2007; Ababneh, 2008; Bani-Mohammad, 2008; Ababneh, et al., 2010, AlHarafsheh, 2016, Alsardia, 2017).

The inter-arrival times of jobs are independent and follow an exponential distribution.

Jobs are scheduled on a First-Come-First-Served (FCFS) basis, FCFS has been used because its fairness.

The execution times of jobs depend on the time needed for flits to be routed through the node, packet sizes, the number of message sent, message contention and distances messages traverse.

The side lengths of the sub-meshes requested by jobs are generated separately and follow a given probability distribution. Two distributions have been considered in this thesis. The first is the uniform distribution over the range from 1 to the mesh side length ( ). The second is the uniform-decreasing distribution. It is determined by four probability 1, 2, 3, and 4, and four integers 1, 2, 3 and 4, where the probability that the width (length) of a request falls in the ranges [1, 1], [1+1, 2], [2+1, 3] and [3+1, 4] is 1, 2, 3, and 4, respectively.

The side lengths within a range are equally likely to happen. For the simulation experiments in this research work, 1=0.4, 2=0.2, 3=0.2, 4=0.2, 1= /8, 2= /4, 3= /2, and 4= . These two distributions have often been used in the literature (Zhu, 1992; Lo, et al, 1997; Chang and Mohapatra, 1998; Chiu and Chen, 1999; Ababneh and Bani-Mohammad, 2003; Bani-Mohammad, et al., 2006, Bani-Mohammad, et al., 2010, Bani-Mohammad and Ababneh, 2013).

Messages are sent inside the network using wormhole switching along with the XY routing.

Messages are of a fixed length (i.e., a fixed number of flits). Furthermore, the number of messages that are generated by a given job are correlated to the job size in the one-to-all, all-to-all, ring, all-to-one, FFT, DQBT, and near-neighbor communication patterns, since each job does exactly one iteration of the given communication pattern, and it is only one message per job in the random communication pattern.

Chapter Three
Simulation Tool and Simulation Results

3.1 Simulation Tool (ProcSimity Simulator)

This section provides a description of the used simulation tool which is called Procsimity (Windisch, et al., 1995; ProcSimity Manual, 1997). Procsimity is used as a software tool for research in the domain of processor allocation and job scheduling in multicomputers and it was written in the C programming language. ProcSimity has been selected since it is open source and includes a detailed simulation of important operations of multicomputers networks. Moreover, the simulator has been widely validated in (Windisch, et al., 1995; ProcSimity Manual, 1997).

The goal of using ProcSimity is to give an environment to analyze the performance of processor allocation and job scheduling algorithms. In particular, ProcSimity is designed in such a way to examine some of the processor allocation problems (i.e., fragmentation and communication overhead problems). The k-ary n-cube and mesh interconnection topologies with dimension-ordered routing are supported in this tool and the tool support flow control technology. The architecture in ProcSimity has been designed to be a network of processors interconnected via message routers at every node. Neighboring nodes are connected by bidirectional communication links. Messages may be routed by either wormhole switching or store-and-forward (Windisch, et al., 1995; ProcSimity Manual, 1997).

The target machine environment that is specified by ProcSimity includes the network topology, routing, and flow control mechanisms, and it provides the users with libraries of predefined scheduling and allocation algorithms. Also, allocation algorithms and scheduling algorithms and even a new communication pattern can be added into ProcSimity tool by any user. Procsimity involves specification of the simulation experiments; it supports stochastic job streams and communication patterns from actual parallel applications. Detailed simulation of message-passing overhead is set by the user at the flit level (Windisch, et al., 1995; ProcSimity Manual, 1997).

3.2 Simulation Results

Wide simulation experiments have been conducted under several communication patterns to compare the performance of the of the existing most famous allocation strategies: First Fit (FF)(Zhu, 1992), Best Fit (BF) (Zhu, 1992), Paging (Lo, et al., 1997), MBS (Lo, et al., 1997) and GABL (Bani-Mohammad, et al., 2007). The performance of the contiguous FF and BF allocation strategies have been chosen in this comparison because they have been shown an average performance in comparison with other allocation strategies in its class (Lo, et al., 1997). The Paging and MBS allocation strategies have been chosen because they have been shown to perform well in (Lo, et al., 1997), and also GABL has been shown to perform well in (Bani-Mohammad, et al., 2007; Bani-Mohammad, et al., 2010; Bani-Mohammad, et al., 2012). The simulation tool employed is ProcSimity that has been used for processor allocation and job scheduling in mesh-connected multicomputers (Windisch, et al., 1995; ProcSimity Manual, 1997).

The target mesh system in this research is a 2D square mesh with a side length . Jobs are assumed to have exponential inter-arrival time. The load of the system is defined as the inverse of mean inter-arrival time of jobs. The jobs in the system are served based on a First-Come-First-Served (FCFS) scheduling policy. The purpose of this research work is to evaluate and compare the performance of the allocation strategies base on FCFS scheduling

. The job execution time is the time needed by a job for completion minus the time of the allocation of a job and job starts the execution. Job execution time depends on the time required for flits to be routed across the nodes, packet sizes, the number of messages to be sent, the message contention inside the network and the distances that the messages traverse (Bani-Mohammad, 2008). The side lengths of each sub-mesh requested by jobs are generated independently and follow a given probability distribution, and two distributions have been considered in this thesis. The first one that is the uniform distribution over the range from 1 to the mesh side length ( ). The second one is the uniform-decreasing distribution. It is determined by four probability 1, 2, 3, and 4, and four integers 1, 2, 3 and 4, where the probability that the width (length) of a request falls in the ranges [1, 1], [1+1, 2], [2+1, 3] and [3+1, 4] is 1, 2, 3, and 4, respectively.

The side lengths within a range are equally likely to happen. For the simulation experiments in this research work, =16, 1=0.4, 2=0.2, 3=0.2, 4=0.2, 1= /8, 2= /4, 3= /2, and 4= . These distributions have been selected because they have been used in the literature (Zhu, 1992; Lo, et al., 1997; Chang and Mohapatra, 1998; Chiu and Chen, 1999; Ababneh and Bani-Mohammad, 2003; Bani-Mohammad, et al., 2006; Bani-Mohammad, et al., 2010, Bani-Mohammad and Ababneh, 2013). The interconnection network uses wormhole routing and XY routing (Ni and McKinley, 1993; Mohapatra, 1998). Flits are assumed to take one-time unit to move between two adjacent nodes, and ts time units to be routed across a node. Plen represents packet sizes. As mentioned previously in Chapter 2, Section 2.4, the allocated processors to a parallel job often exchange messages together based on a given communication pattern (Lo, et al., 1997). In this thesis, eight communication patterns have been considered to evaluate the performance of contiguous

and noncontiguous allocation algorithms. They are one-to-all communication pattern (ProcSimity Manual, 1997, Lo, et al., 1997), random communication pattern (ProcSimity Manual, 1997), all-to-all communication pattern (ProcSimity Manual, 1997, Lo, et al., 1997), near neighbor communication pattern (Bani-Mohammad and Ababneh, 2013), ring communication pattern (ProcSimity Manual, 1997; Lo, et al., 1997), all-to-one communication pattern (Grama, et al., 2003), Fast Fourier Transform (FFT) communication pattern (James W. Cooley and John W. Tukey, 1964; Lo, et al., 1997;Grama, et al.,2003; Chan, et al., 2008), Divide and Conquer Binomial Tree (DQBT)(Lo, et al., 1996 ; Lo, et al., 1997; Valero-Garcia, et al.,1997; Grama, et al.,2003).

The performance figures presented in the following sections in this chapter adopt the following parameters: the mesh size is a 16×16, ts= 3 time units, Plen= 8 flits. Parameters are explained in Table 3.1, it is worth noting that most of the values of these parameters have been recommended in (ProcSimity Manual, 1997) and have been adopted in the literature (Zhu, 1992; Babbar and Krueger, 1994; Lo, et al., 1997; Bani-Mohammad, et al., 2010).

Table 3. 1: The System Parameters used in the Simulation Experiments.

Simulation Parameter	Value
Dimensions of the Mesh	16×16
Packet Length	8 flits
Flow Control Mechanism	Wormhole Routing
Routing Delay	3 time units
Router Type	Mesh XY Routing
Allocation Strategy	FF, BF, GABL, MBS, and Paging(0)
Scheduling Strategy	FCFS
Job Size Distribution	Uniform: Job widths and lengths are uniformly distributed over the range from 1 to the mesh side lengths. Uniform Decreasing: Represents the case where most jobs are small relative to the size of the system.
Inter-arrival Time	Exponential with different values for mean. The values are determined through

	experimentation with the simulator, ranged from lower values to higher values.
Mean Time between Sends	0.0
Communication Pattern	One-to-all, Random, All-to-all, All-to-one, Ring, FFT, DQBT and Near Neighbor.
Messages per job	Messages per job are correlated to the job size, since each job does exactly one iteration of the given communication pattern, except for Random communication pattern, where the number of messages per job is only one.
Number of Runs	The number of runs should be enough so that the confidence level is 95% and the
	relative errors are below 5% of the means. The number of runs ranged from dozens to hundreds.
Number of Jobs per Run	1000

Each simulation run consists of 1000 completed jobs. Simulation experiments are repeated for independent runs until the confidence level reaches 95%, and the relative errors do not exceed 5% (note: ProcSimity simulator count the percentage of error for each run by itself).

The main performance parameters used are mean system utilization and the average turnaround time of jobs. The turnaround time of a job is the time that the job spends in the system from arrival to departure while the system utilization is the percentage of processors that are utilized over a given period of time (Bani-Mohammad, 2008). The most important independent variable in the simulation is the system load. System load is defined as the inverse of the mean inter-arrival time of jobs and its range of values from low to heavy loads, and it has been determined through experimentation with the simulator allowing each allocation strategy to reach its upper limits of utilization (Bani-Mohammad, 2008). In the figures that are presented below, the x-axis represents the system load while the y-axis represents the results of the performance metric of interest.

3.1 Turnaround Time

In Figures 3.1 and 3.2, the average turnaround times are plotted against the system load for the near neighbor communication pattern. The performance of contiguous allocation strategies (FF and BF) substantially better than all noncontiguous allocation strategies (Paging, MBS, and GABL). This is because the allocated processors for jobs are contiguous and form rectangular shapes, which means that there is no interference between messages of different jobs and in near neighbor communication pattern, each node allocated to a job communicates only with its neighbors (up, down, right, left), which means that there is no any message contention between messages of this job, and this results in reducing the overall communication overhead and hence improves the system performance. The

performance of FF is very close to that of BF. Also, the performance of Paging(0) is very close to that of MBS, because in near neighbor communication pattern, each node allocated to a job communicates with its neighbors (up, down, right, left) that are allocated to the same job and those strategies maintain a high degree of contiguity between the allocated processors for a job and the allocated sub-meshes form rectangular shapes.

The GABL noncontiguous allocation strategy performs better than the other noncontiguous allocation strategies (Paging and MBS). This is because GABL combines the desirable features of both contiguous and noncontiguous allocation; it allocates sub-meshes in a rectangular form and tries to maintain a high degree of contiguity between the processors in the allocated sub-meshes, which means that the distances between communicating nodes are relatively low, where the distances have significant impact on message latency when messages are short (in this research work, the length of packets is 8 flits). If the distances traversed by messages are short then they are less likely to collide with other messages, which in turn decreases the communication overhead. Consequently, the turnaround time is lower.

For example, in Figure 3.1, the average turnaround times of FF and BF are 0.2%, 0.59% and 0.2% of that of Paging (0), GABL, MBS, respectively, when the system load is 0.01 jobs/time unit. In Figure 3.2, the same relative performance can be seen when the uniform decreasing distribution is used, but the differences in the relative performance are less severe.

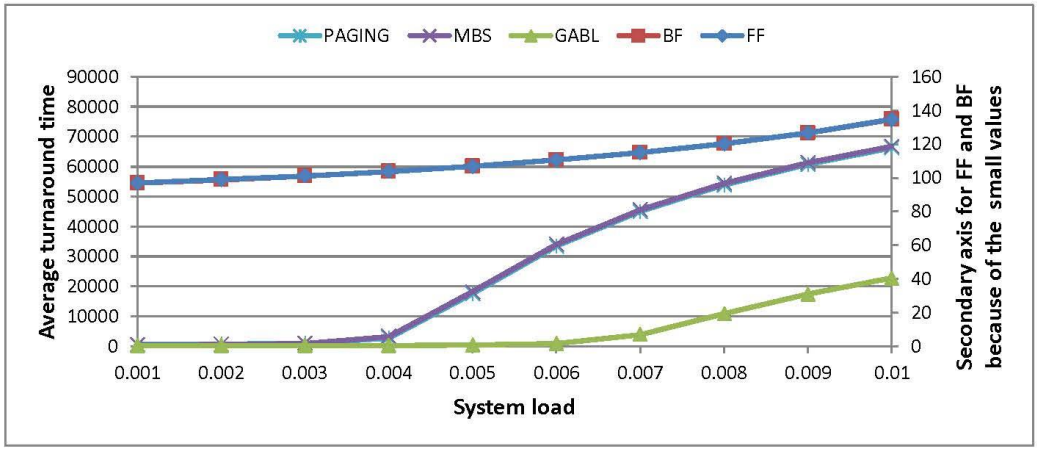

Figure 3.1: Average turnaround time vs. system load for the near neighbor communication pattern and uniform side lengths distribution in a 16×16 mesh.

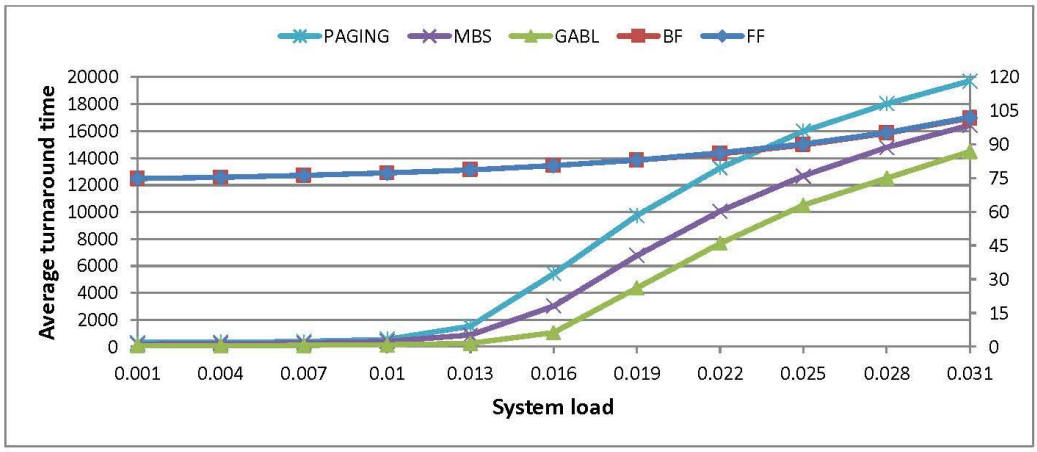

Figure 3.2: Average turnaround time vs. system load for the near neighbor communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

In Figures 3.3, the average turnaround times of jobs are plotted against the system load for the one-to-all communication pattern. The results reveal that in most cases, the performance of noncontiguous allocation strategies (Paging(0), MBS, GABL) is relatively the same and they perform better than both FF and BF contiguous allocation strategies for both job distributions considered in this research work. This is due to the elimination of both internal and external fragmentation in noncontiguous allocation strategies (Paging(0), MBS, GABL) that results in better system utilization and hence improves the system performance with regard to jobs turnaround time. The improvement of system utilization outbalanced the impact of the interference between messages of different jobs encountered in noncontiguous allocation. Also, the results reveal that the performance of FF is very close to that of BF. For example, the average turnaround times of GABL are 58%, 58%, 97% and 99% of that of FF, BF, MBS, and Paging(0), respectively, when the system load is 0.001 jobs/time unit.

In Figure 3.4, when the uniform decreasing distribution is used, the average turnaround times of all noncontiguous and contiguous allocation strategies are improved, but the relative performance remains almost the same as when the uniform distribution is used.

The increased probability of small jobs to be allocated is the cause for this improvement in turnaround times. For example, the average turnaround times of GABL are 53%, 53%, 99% and 99% of that of FF, BF, MBS, and Paging, respectively, when the system load is 0.0054 jobs/time unit.

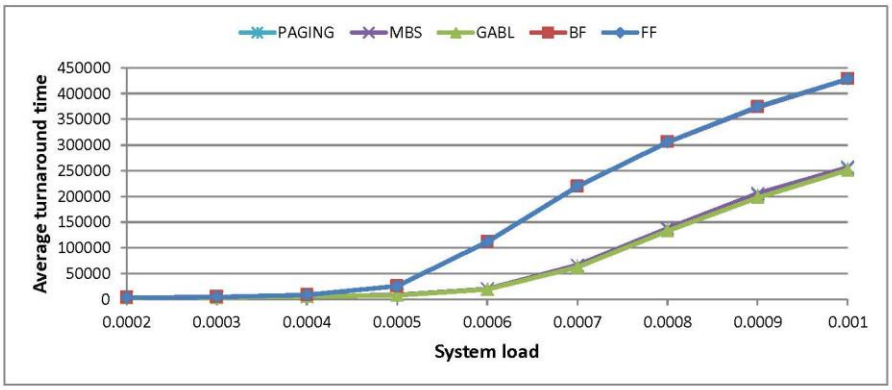

Figure 3.3: Average turnaround time vs. system load for the one-to-all communication pattern and uniform side lengths distribution in a 16×16 mesh.

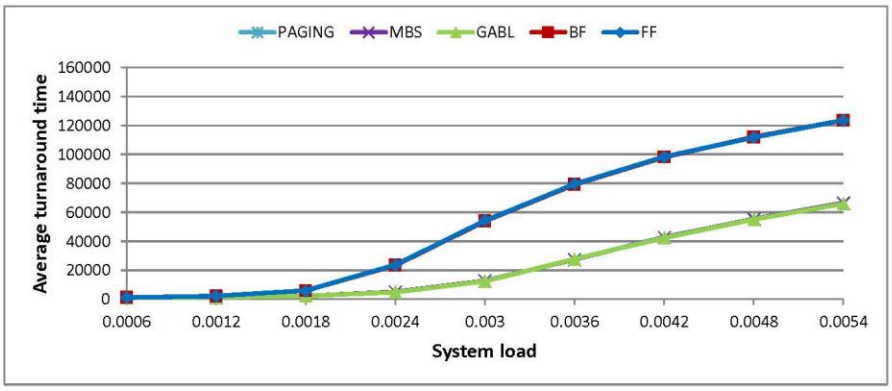

Figure 3.4: Average turnaround time vs. system load for the one-to-all communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

In Figures 3.5 and 3.6, the average turnaround times are plotted against the system load for the random communication pattern. In Figure 3.5, the results reveal that in most cases, the performance of noncontiguous strategies (Paging(0), MBS, GABL) is relatively the same and they all better than both the FF and BF contiguous allocation strategies for uniform job distributions considered in this thesis. Also, the results reveal that the performance of FF is very close to that of BF. For example, the average turnaround times of GABL are 60%, 61%, 96% and 96% of that of FF, BF, MBS, and Paging(0), respectively, when the system load is 0.091 jobs/time unit. Figure 3.6, shows a slight relative performance improvement when the uniform decreasing distribution is used. This is due to the increased probability of small jobs (relative to mesh size) when using this distribution. For example, the average turnaround times of MBS are 57%, 57%, 98% and 96% of that of FF, BF, Paging(0), and GABL, respectively, when the system load is 0.29 jobs/time unit.

The random communication pattern can only give a glance about the ability of the noncontiguous allocation strategies to mitigate the message contention. However, the contention generated when using the random communication pattern is not sufficient to recognize between the allocation strategies. This is because in random communication pattern each job sent only one message from a randomly selected source node to randomly selected destination node.

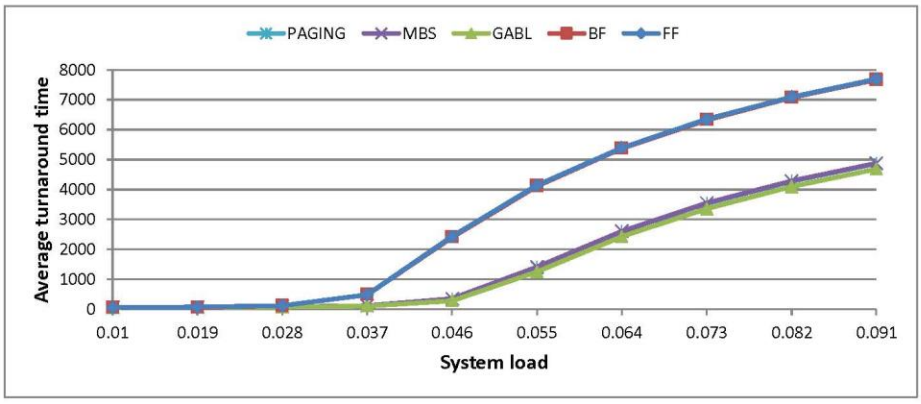

Figure 3.5: Average turnaround time vs. system load for the random communication pattern and uniform side lengths distribution in a 16×16 mesh.

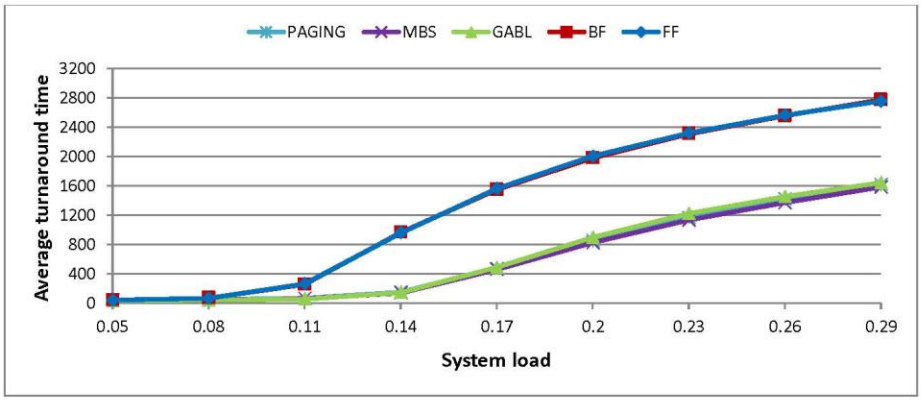

Figure 3.6: Average turnaround time vs. system load for the random communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

In Figures 3.7 and 3.8, the average turnaround times of jobs are plotted against the system load for the all-to-all communication pattern. The results reveal that FF and BF contiguous allocation strategies substantially better than the MBS noncontiguous allocation for uniform side lengths distribution, and FF and BF performs better than MBS for uniform decreasing distribution. This refers to the much massage contention that exists in all-to-all communication pattern which considered as the weak point of the noncontiguous allocation strategies (Suzaki, et al., 1996), where the number of messages per job increases dramatically as the job size increases. The delay would increase when the message contention increases and this, in turn, defeat the gain of the improved system utilization; and consequently, degrades the system performance with regard to jobs turnaround time (Min and Mutka, 1994; Mache and Lo, 1997). This is what happened with MBS. In MBS, the allocated sub-mesh is restricted to a base 4 square blocks, which means that it may fail to allocate a requested sub-mesh contiguously even if there is a one exist, and may divide a sub-mesh request without any need to do that and allocate the parts far apart of each other, especially for large jobs, and this can seriously increase the message contention.

The results reveal that GABL produces the best results in all cases. This is because GABL has been designed for achieving a high level of contiguity, and this is done by giving priority to allocating the largest possible free sub-meshes while avoiding external processor fragmentation. Also, Paging(0) is better than MBS, and this is because the distances between the allocated processors in Paging(0)is less than those in MBS, which decreases the probability of the interference among job's messages, and that decreases the contention and hence improves the system performance with regard to average turnaround time of jobs. Also, the results reveal that the performance of FF is very close to that of BF.

For example, in Figure 3.7, the average turnaround times of GABL are 74%, 75%, 83% and 42% of that of FF, BF, Paging(0), and MBS, respectively, when the system load is (0.00009) jobs/time unit.

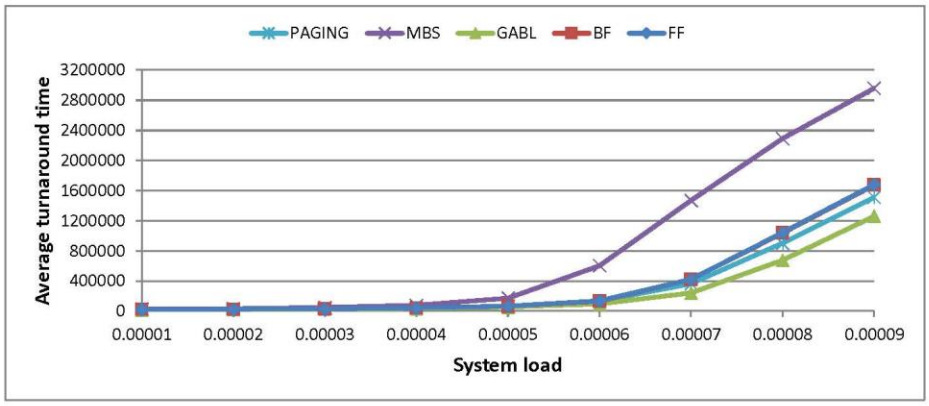

Figure 3.7: : Average turnaround time vs. system load for the all-to-all communication pattern and uniform side lengths distribution in a 16×16 mesh

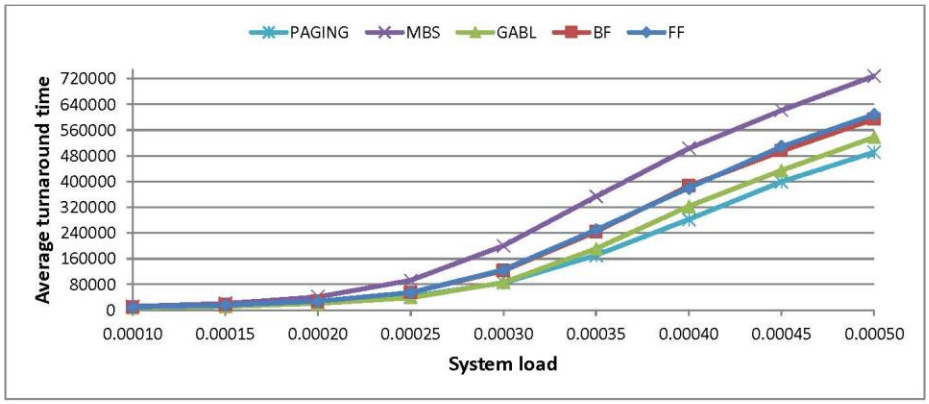

Figure 3.8: : Average turnaround time vs. system load for the all-to-all communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

In Figures 3.9 and 3.10, the average turnaround times are plotted against the system load for the FFT communication pattern. The results in these figures reveal that FF and BF contiguous allocation strategies dramatically better than all noncontiguous allocation strategies (MBS, Paging(0), and GABL). This is because the allocated processors for jobs are contiguous and form rectangular shapes, which means that there is no interference between messages of different jobs and in FFT communication pattern, the allocated number of processors for a job is divided by two, resulting in two halves, each processor in the first half communicate with its corresponding processor in the second half. This stage is repeated until one processor remains, which means that there is a less message contention among the messages of the same job, and this results in reducing the overall communication overhead and hence improves the system performance with regard to jobs turnaround time. The performance of FF is very close to that of BF. Also, the figures reveal that the MBS noncontiguous strategy perform better than the other noncontiguous allocation strategies (Paging(0) and GABL) and it performs well for both job size distributions considered in this research work. This is because the side lengths of the job request are truncated to a power of two that favors MBS and power of two sizes are suitable to the request partitioning process used in MBS. Jobs are allocated to a small number of square sub-meshes, and these sub-meshes are often neighbors due to the method used for maintaining and allocating free blocks in MBS allocation strategy. Also, the results reveal that Paging(0) and GABL perform poorly in this communication pattern because they can allocate processors that are relatively far apart, which can increase the distance traversed by messages and hence increases the message contention, which in turn results in degrading the system performance with regard to jobs turnaround time. In Figure 3.9, for example, the average turnaround times of FF and BF are 38%, 21%, and 15% of that of MBS, Paging(0), and GABL, respectively, when the job arrival rate is 0.013 jobs/time unit.

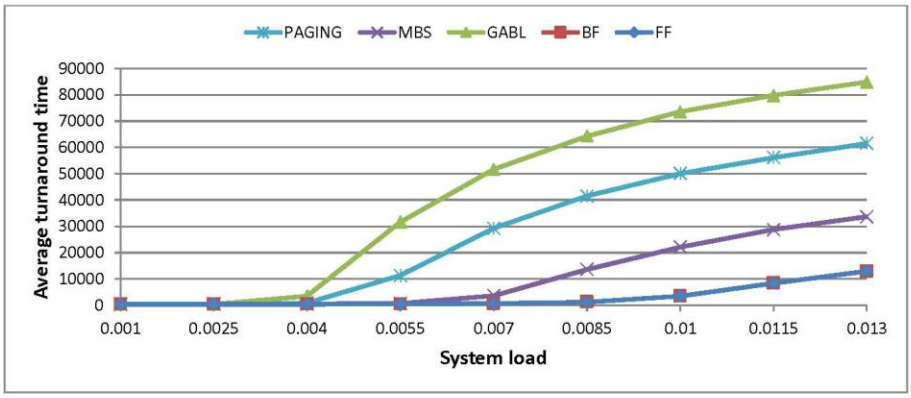

Figure 3.9: Average turnaround time vs. system load for the FFT communication pattern and uniform side lengths distribution in a 16×16 mesh.

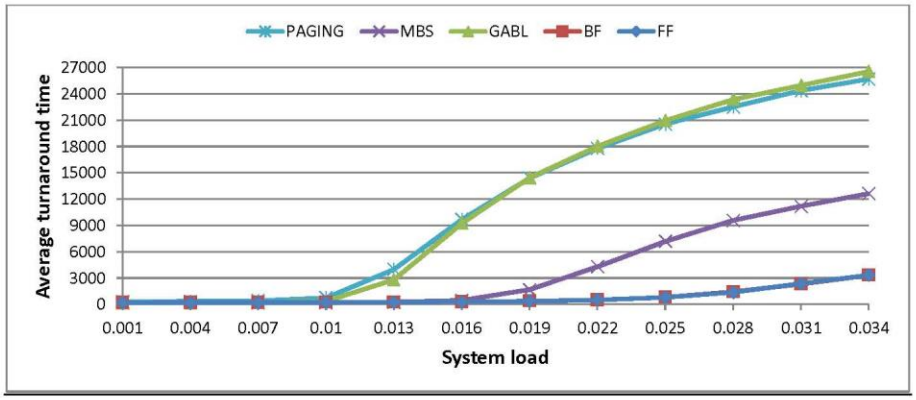

Figure 3.10: : Average turnaround time vs. system load for the FFT communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

In Figures 3.11 and 3.12, the average turnaround times are plotted against the system load for the DQBT communication patterns. The results in these figures reveal that FF, BF, and MBS allocation strategies perform well for both job size distributions and dramatically better than the other noncontiguous allocation strategies (Paging(0) and GABL). FF and BF perform well because the allocated processors for jobs are contiguous and form rectangular shapes, which means that there is no interference between messages of different jobs and in DQBT communication pattern, the allocated number of processors for a job is divided by two, resulting in two halves. This stage is repeated until one processor remains. In the combine stage, one processor in the first half communicate with one processor in the second half, which means that there is a less message contention between the messages of the same job, and this results in reducing the overall communication overhead and hence improves the system performance with regard to jobs turnaround time.MBS performs well because the side lengths of the requested sub-mesh are truncated to a power of two that favors MBS, and power of two sizes are suitable to the request partitioning process that it is used in MBS. Jobs are allocated to a small number of square sub-meshes, and these sub-meshes are often neighbors due to the method used for maintaining and allocating free blocks in MBS allocation strategy.

Also, the results reveal that Paging(0) and GABL perform poorly in this communication pattern because they can allocate processors that are relatively far apart, which can increase distances traversed by messages and hence message contention is increased, and this in turn results in degrading in the system performance with regard to jobs turnaround time. In Figures 3.11, for example, the average turnaround times of FF and BF are 96%, 48%, and 31% of those of MBS, Paging(0), and GABL, respectively, when the job arrival rate is 0.046 jobs/time unit.

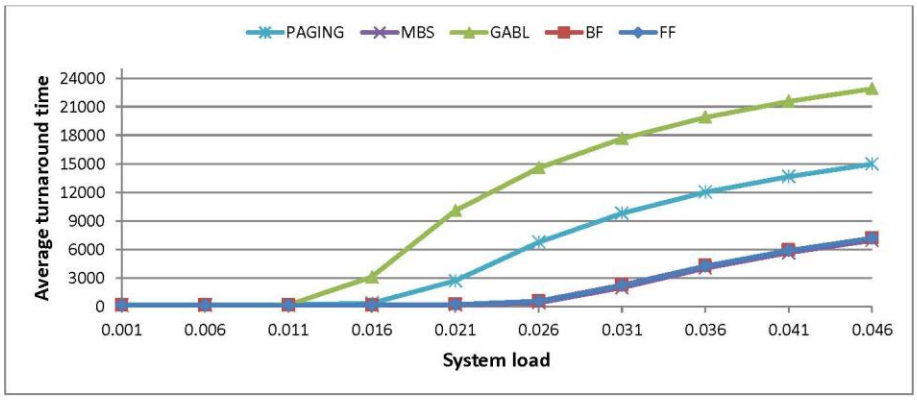

Figure 3.11: Average turnaround time vs. system load for the DQBT communication pattern and uniform side lengths distribution in a 16×16 mesh.

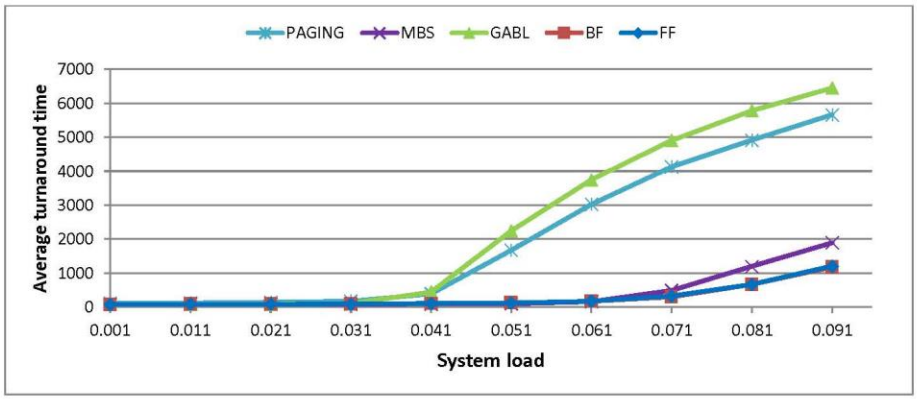

Figure 3.12: : Average turnaround time vs. system load for the DQBT communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

In Figures 3.13 and 3.14, the average turnaround times of jobs are plotted against the system load for the Ring communication pattern. The results in these figures reveal that noncontiguous allocation strategies (MBS, Paging(0), and GABL) perform better than contiguous allocation strategies (FF and BF) for both job size distributions. This is because although there is an increase in the contention but still remains relatively low due to the fact that some degree of contiguity is maintained, and this allows the ring communication to still be executed efficiently and in this communication pattern, each processor allocated to a job sends a packet only to its successor which means that the distances traversed by messages are short and number of massages is less so they are less likely to collide with other messages. This results in reducing the communication overhead and hence the turnaround time is lower. In figures 3.13, the performance of noncontiguous strategies (Paging(0), MBS, GABL) is relatively the same, and the performance of FF is very close to that of BF. In figures 3.14, the performance of Paging(0) allocation strategy is better than the other noncontiguous strategies (MBS, GABL) and the performance of FF is very close to that of BF. In figure 3.13, for example, the average turnaround times of Paging(0) are 60%, 60%, 99%, and 92% of that of FF, BF, MBS, and GABL, respectively, when the job arrival rate is 0.046 jobs/time unit.

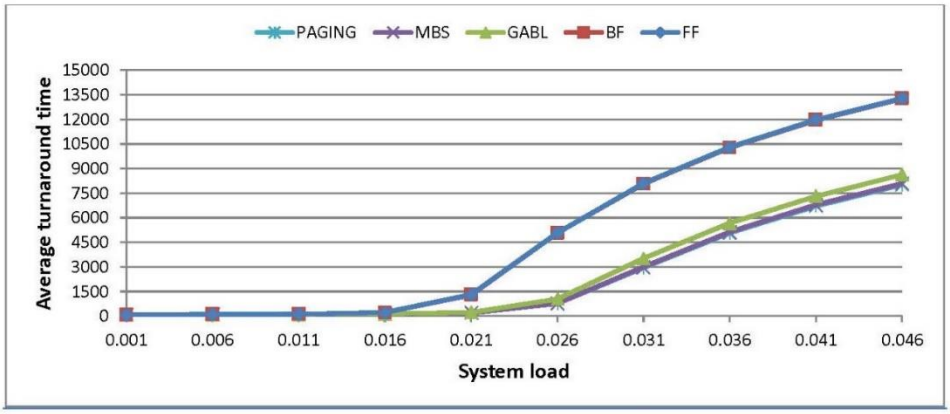

Figure 3.13: Average turnaround time vs. system load for the Ring communication pattern and uniform side lengths distribution in a 16×16 mesh.

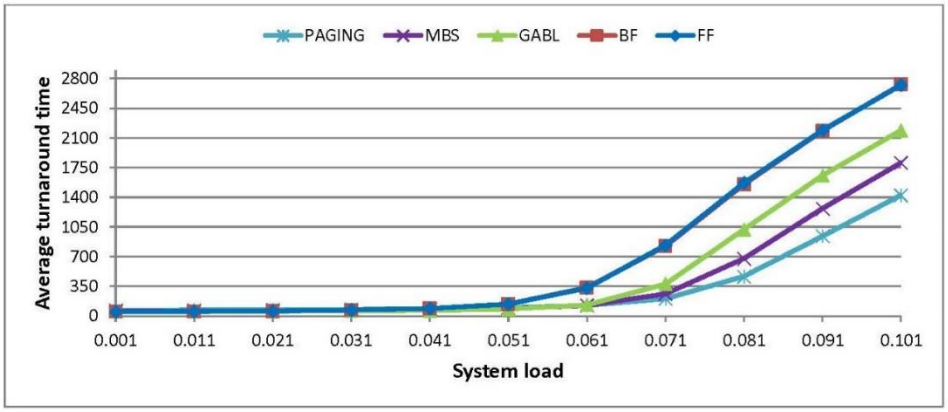

Figure 3.14: Average turnaround time vs. system load for the Ring communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

In Figures 3.15, the average turnaround times of jobs are plotted against the system load for the All-to-One communication pattern. The results reveal that in most cases, the performance of noncontiguous allocation strategies (Paging(0), MBS, GABL) is relatively the same and they perform better than both FF and BF contiguous allocation strategies for uniform job distributions considered in this thesis. This due to the elimination of both internal and external fragmentation in noncontiguous allocation strategies (Paging(0), MBS, GABL) that results in better system utilization and that can improve the system performance with regard to jobs turnaround time. The improvement of system utilization outbalanced the impact of the interference between messages of different jobs encountered in noncontiguous allocation. Also, the results reveal that the performance of FF is very close to that of BF. For example, in Figure 3.15, the average turnaround times of Paging are 30%, 30%, 77% and 76% of those of FF, BF, MBS, and GABL, respectively, when the job arrival rate is 0.001 jobs/time unit.

In Figure 3.16, when the uniform decreasing distribution is used, the average turnaround times of all noncontiguous and contiguous allocation strategies are improved, but the relative performance remains almost the same as when the uniform distribution is used.

The increased probability of small jobs to be allocated is the reason for this improvement in turnaround times. Furthermore, message contention decreased in noncontiguous allocation strategies because, in the All-to-One communication pattern, the number of messages for a job is correlated to the job size.

Note: this is the same explanation for the One-to-All, because All-to-One is the dual of One-to-All communication; a dual of a communication is the opposite of the original operation (Grama, et al., 2003).

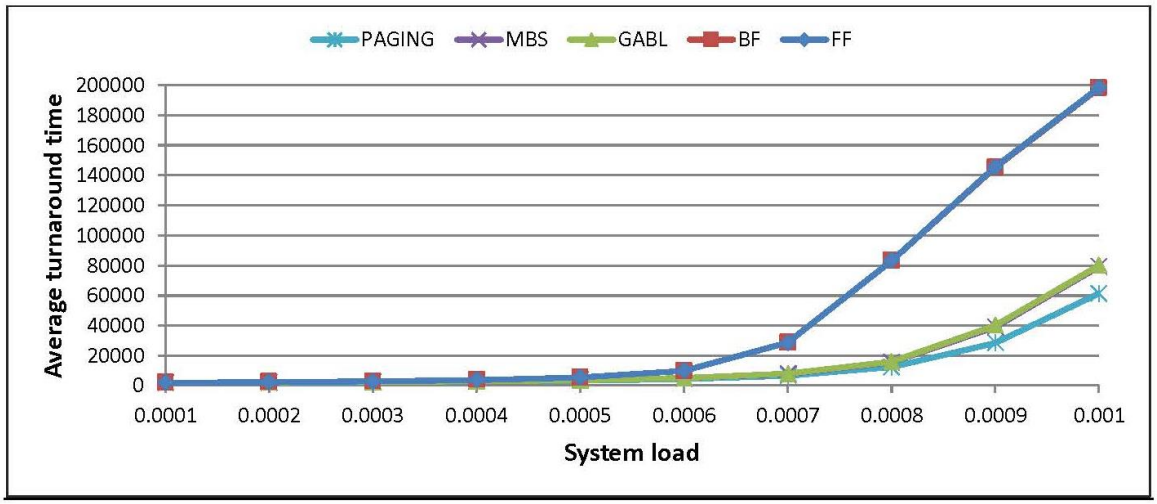

Figure 3.15: Average turnaround time vs. system load for the All-to-One communication pattern and uniform side lengths distribution in a 16×16 mesh.

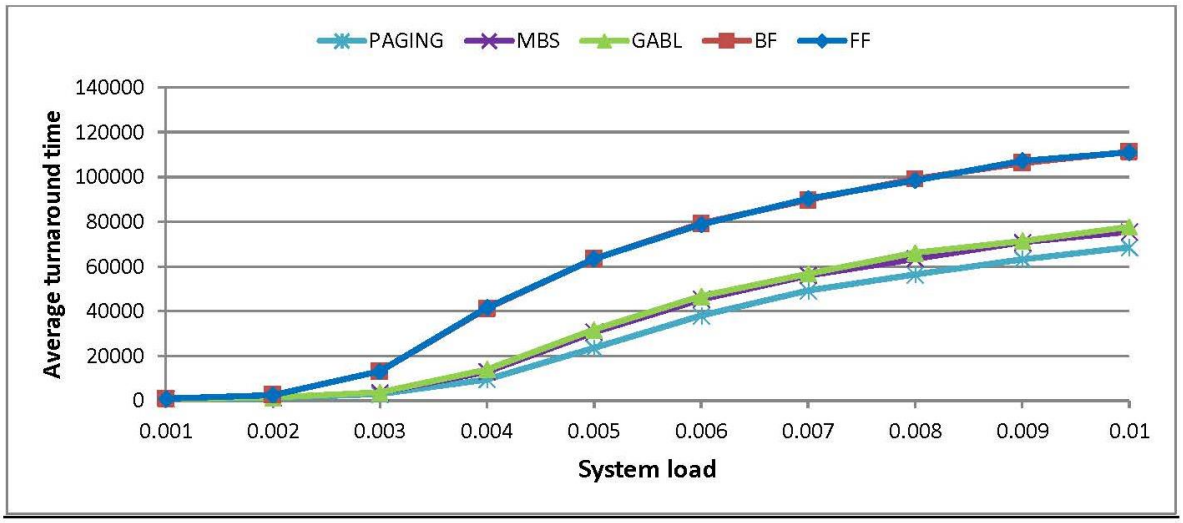

Figure 3.16: Average turnaround time vs. system load for the All-to-One communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

3.3 System Utilization

Figures 3.33 - 3.34 show the mean system utilization of the considered allocation strategies (FF, BF, Paging(0), MBS, and GABL) using the eight communication patterns and two job size distributions. The values of the load were obtained for heavy system loads, and the heavy loads cause the waiting queue to be filled very early which let the allocation strategies to reach the upper limit of the system utilization. The results reveal that noncontiguous allocation strategies dramatically better than contiguous allocation strategies with regard to mean system utilization. This is because contiguous allocation results in high fragmentation because the allocation of a requested sub-mesh needs contiguity between its processors and the sub-mesh of the allocated processors must have the same topology as multicomputer; these conditions reduce the chance of successful allocation and consequently reduce the mean system utilization. The contiguous FF and BF strategies cannot exceed 71% and 63% utilization for uniform and uniform decreasing job size distributions, respectively. The results for uniform decreasing job size distribution is less good than those of uniform job size distribution, and this because it represents the case where most jobs are small relative to the size of the mesh system and hence decreases the

number of allocated processors for the job requests, which in turn affects negatively on system performance with regard to system utilization. The noncontiguous allocation strategies (Paging(0), MBS, GABL) achieve a mean system utilization of 92% for uniform and uniform decreasing job size distributions, respectively. The performance of the noncontiguous allocation strategies considered in this research is very close because they have the same ability to eliminate internal and external processor fragmentation and always the allocation succeed if there are enough free processors.

Figures 3.19 - 3.34 in the appendix, show the mean system utilization of the considered allocation strategies (FF, BF, Paging(0), MBS, and GABL) using the eight communication patterns and two job size distributions. The values of the load ranged from moderate to heavy loads. The results reveal that noncontiguous allocation strategies dramatically better than contiguous allocation strategies with regard to mean system utilization

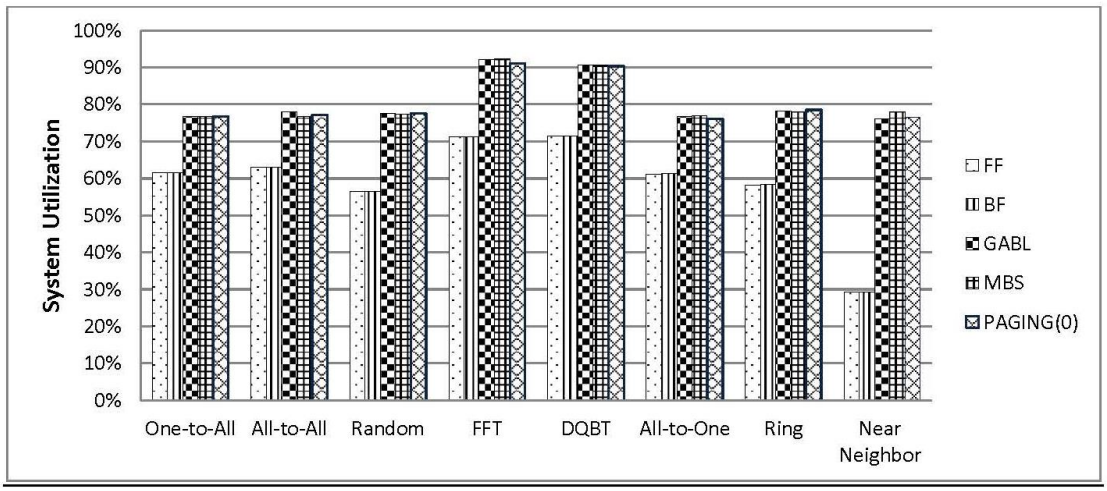

Figure 3.17: System utilization of the contiguous and noncontiguous allocation strategies (FF, BF, GABL, MBS, and Paging(0)), for the eight communication patterns tested, and uniform side lengths distribution in a 16×16 mesh.

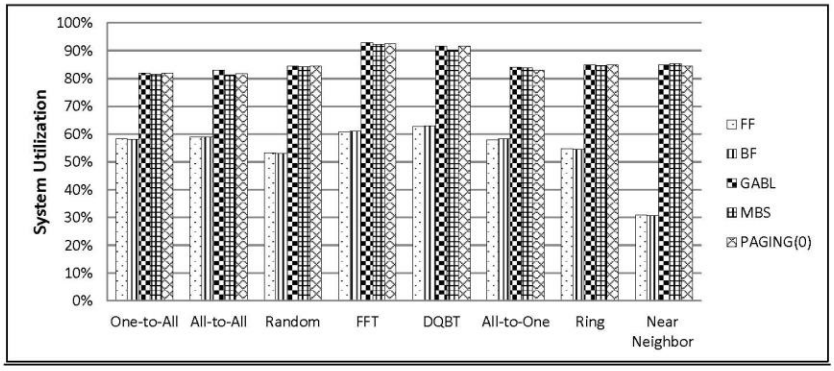

Figure 3.18: System utilization of the contiguous and noncontiguous allocation strategies (FF, BF, GABL, MBS, and Paging(0)), for the eight communication patterns tested, and uniform decreasing side lengths distribution in a 16×16 mesh.

Chapter Four

Conclusion and Directions for future work

4.1 Conclusion

Many research studies have been investigated the processor allocation in multicomputers, especially those based on mesh network (Li and Cheng, 1991; Zhu, 1992; Chuang and Tzeng, 1994; Lo, et al, 1997; Chang and Mohapatra, 1998; Ababneh, 2001; Bani-Mohammad, et al., 2007; Ababneh, 2008; Ababneh, et al., 2010; Bani-Mohammad, et al., 2012). But to the best of our knowledge, there is no study that considers the effect of the Near Neighbor, Ring, All to all, Divide and Conquer Binomial Tree (DQBT), Fast Fourier Transform (FFT),One to All, All to One, and Random communication patterns on the performance of contiguous and noncontiguous processor allocation in multicomputers, especially when each job does exactly one iteration of the given communication pattern. The communication pattern used by a program may have a great impact on the performance of contiguous and noncontiguous processor allocation in multicomputers (Bani-Mohammad, et al., 2013). In this thesis, the performance of the most famous contiguous allocation strategies (First Fit, Best Fit) and most famous noncontiguous allocation strategies (GABL, Paging, MBS) for 2D mesh multi-computers is re-visited considering several important communication patterns, including one-to-all (ProcSimity Manual, 1997), near neighbor (Bani-Mohammad and Ababneh, 2013), random (ProcSimity Manual, 1997),

all-to-all (ProcSimity Manual, 1997; Lo, et al., 1997), ring (ProcSimity Manual, 1997; Lo, et al., 1997), Divide and Conquer Binomial Tree (DQBT)(Lo, et al., 1996 ; Lo, et al., 1997; Valero-Garcia, et al.,1997; Grama, et al.,2003), Fast Fourier Transform (FFT) (James W. Cooley and John W. Tukey, 1964; Lo, et al., 1997;Grama, et al.,2003; Chan, et al., 2008), all-to-one (Grama, et al., 2003). Two distributions have been considered in this research work, which they are the uniform and uniform-decreasing distributions. Wide simulation experiments have been conducted to compare the performance of contiguous allocation with that of noncontiguous allocation with regard to average turnaround time and mean system utilization using the ProcSimity simulator.

The simulation results for average turnaround time have shown that in near neighbor, FFT and DQBT communication patterns, the performance of contiguous allocation strategies (FF and BF) dramatically better than all noncontiguous allocation strategies (Paging(0), MBS and GABL) with regard to average turnaround; except for MBS in DQBT communication pattern, where its performance is very close to that of FF and BF. These results prove that the taken fact that said the noncontiguous allocation strategies always dramatically better than contiguous allocation strategies with regard to average turnaround time is not absolutely true. Also, the simulation results have shown that in one-to-all, random, ring and all-to-one communication patterns, the performance of noncontiguous allocation strategies (Paging(0), MBS and GABL) dramatically is better than that of contiguous allocation strategies (FF and BF) with regard to average turnaround time. For all-to-all communication pattern, the simulation results have shown that the performance of contiguous allocation strategies (FF and BF) is better than that of the MBS noncontiguous allocation strategy, but the performance of GABL and Paging(0) is better than that of FF, BF, and MBS.

The results for system utilization have shown that in all communication patterns that are considered in this research work, the noncontiguous allocation strategies dramatically better than the contiguous allocation strategies with regard to mean system utilization.

4.2 Directions for the Future Works

There are interesting issues that can be considered as an expansion of this research work in the future. Some of these issues are briefly described below

It would be interesting to re-examine the performance of the allocation strategies with other possible scheduling approaches, such as Out-of-Order (OO) (Ababneh, 2001), Shortest-Service-Demand-First (SSD) (Krueger, et al., 1994), and Window-based job scheduling (Ababneh and Bani-Mohammad, 2011)

It would be interesting to re-examine the performance of most famous allocation strategies for other common multicomputer networks, such as the torus and hypercube networks, considering several important communication patterns that were used in this research work.

References

Ababneh, I. (2001). Job scheduling and contiguous processor allocation for three-dimensional mesh multicomputers. AMSE Advances in Modelling and Analysis, 6(4), pp. 43-58.

Ababneh, I. (2008). Availability-based noncontiguous processor allocation policies for 2D mesh-connected multicomputers. Journal of Systems and Software, 81(7), pp. 1081-1092.

Ababneh, I., Bani-Mohammad, S., and Hamdan, M. (2010). Comparative Performance Evaluation of Non-Contiguous Allocation Algorithms in 2D Mesh-Connected Multicomputers. Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT 2010), pp. 2933–2939. Washington, DC: IEEE Computer Society.

Ababneh, I., and Bani-Mohammad, S. (2011). A new window-based job scheduling scheme for 2D mesh multicomputers. Simulation Modelling Practice and Theory, 19(1), pp. 482-493.

Adve, V., and Vernon, M. (1994). Performance analysis of mesh interconnection networks with deterministic routing. IEEE Transactions on Parallel and Distributed Systems, 5(3), pp. 225-246.

AlHarafsheh, R. (2016). Irregular Shape Strategy for Non-contiguous Sub-mesh Allocation in 2D Mesh-Connected Multicomputers. Master thesis in Computer Science from Al al-Bayt University.

Alsardia, D. (2017). A Row Based Non-Contiguous Processor Allocation Strategy for 2D Mesh-Connected Multicomputer. Master thesis in Computer Science from Al al-Bayt University.

Athas, W.C., C.L. Seitz (1988). Multicomputers: message-passing concurrent computers, IEEE Computer, 21(8), pp. 9–24.

Bailey, D.H., E. Barszcz, L. Dagum, and H.D. Simon (1994).“NAS Parallel Benchmark Results 3-94,” Technical Report RNR-94-006, NASA Ames Research Center, Moffett Field, Calif., Mar.

Babbar, D., and Krueger, P. (1994). A performance comparison of processor allocation and job scheduling algorithms for mesh-connected multiprocessors. Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing, pp. 46-53. Dallas, TX.

Bani-Mohammad, S. (2008). Efficient Processor Allocation Strategies for Mesh-Connected Multicomputers. PhD Thesis, Department of Computing Science, University of Galsgow, Glasgow, U.K.

Bani-Mohammad, S., and Ababneh, I. (2009). Comparative evaluation of contiguous allocation strategies on 3D mesh multicomputers. Journal of Systems and Software, 82,pp.307–318

Bani-Mohammad, S., and Ababneh, I. (2013). On the performance of non-contiguous allocation for common communication patterns in 2D mesh-connected multicomputers. Simulation Modelling Practice and Theory, 32, pp. 155-165.

Bani-Mohammad, S., Ababneh, I., and Yassen, M. (2012). Non-contiguous processor allocation in the mesh-connected multicomputers using compaction. International Conference on Computer Systems and Industrial Informatics, pp. 1-8. Sharjah.

Bani-Mohammad, S., Ould-Khaoua, M., and Ababneh, I. (2007). An Efficient Non-Contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers. Journal of Information Sciences, 177(14), pp. 2867-2883.

Blumrich, M., Chen, D., Coteus, P., Gara, A., Giampapa, M., Heidelberger, P., Singh, S., Steinmacher-Burow, B., Takken, Steinmacher-Burowmin T. and Vranas, P. (2003).Design and Analysis of the BlueGene/L Torus Interconnection Network. IBM Research Report RC23025, IBM Research Division. Thomas J. Watson Research Center.

Chan, A., Balaji, P., Thakur, R., W. Gropp, E. Lusk (2008). Communication Analysis of Parallel 3D FFT for Flat Cartesian Meshes on Large Blue Gene Systems, in: HiPC, Bangalore, India.

Chang, C.-Y and Mohapatra, P. (1998). Performance improvement of allocation schemes for mesh- connected computers. Journal of Parallel and Distributed Computing, 52(1), pp. 40-68.

Chiu, G.-M., and Chen, S.-K. (1999). An efficient submesh allocation scheme for two-dimensional meshes with little overhead. IEEE Transactions on Parallel and Distributed Systems, 10(5), pp. 471-486.

Chuang, P., and Tzeng, N. (1994). Allocating precise submesh in mesh-connected systems.

IEEE Transaction on Parallel and Distributed Systems, 5(2), pp. 211-217.

Cray. (2005). Cray XT3 Datasheet.

Drewes, C. (1996). Simulating Virtual Cut-through and Wormhole Routing in a Clustered Torus. M.Sc. Thesis, Laboratory of Computer Architecture and Digital Techniques (CARDIT), Faculty of Electrical Engineering, Delft University of Technology.

Duato, J., Yalamanchili, C., and Ni, L. (1997). Interconnection Networks: An Engineering

Approach (1st ed.). Los Alamitos, CA, USA: IEEE Computer Society Press.

Ferreira, A., velLejbman, G., and Song, S. (1994). Bus based parallel computers: A viable way for massive parallelism. Proceedings of Parallel Architectures Languages Europe (PARLE '94), Lecture Notes in Computer Science 817, pp. 553-564. Berlin, Heidelberg: Springer Berlin Heidelberg.

Foster, I. (1995). Designing and building parallel programs: concepts and tools for parallel

software engineering. MA: Addison-Wesley.

Grama, A., Kumar, V., Gupta, A., and Karypis, G. (2003). Introduction to Parallel Computing.

Rewood City, California: The Benjamin/Cummings publishing company, Inc.

Intel Corporation. (1991). A Touchstone DELTA system description.

Intel Corporation. (1991). Paragon XP/S product overview. Beaverton, Oregon:

Supercomputer Systems Division.

James W. Cooley, John W. Tukey (1964). An algorithm for the machine calculation of complex fourier series, Mathematics of Computation 19 (90), pp. 297–301.

Krueger, P., Lai, T., and Radiya, V. (1994). Job scheduling is more important than processor allocation for hypercube computers. IEEE Transactions on Parallel and Distributed Systems, 5(5), pp. 488-497.

Li, k., and Cheng, K. -H. (1991). A Two-Dimensional Buddy System for Dynamic Resource Allocation in a Partitionable Mesh Connected System, Journal of Parallel and Distributed Computing, 12(1), pp. 79-83.

Lo, V., S. Rajopadhye, J.A. Telle (1996). Parallel divide and conquer on meshes, IEEE Transactions on Parallel and Distributed Systems, 7(10),pp. 1049–1057.

Lo, V., Windisch, K., Liu, W., and Nitzberg, B. (1997). Non-contiguous processor allocation algorithms for mesh-connected multicomputers. IEEE Transactions on Parallel and Distributed Systems, 8(7), pp. 712-726.

Mache, J., and Lo, V. (1997). The Effects of Dispersal on Message-Passing Contention in Processor Allocation Strategies. Third Joint Conference on Information Sciences, Sessions on Parallel and Distributed Processing, pp. 223-226.

Mache, J., Lo, V., and Windisch, K. (1997). Minimizing Message-Passing Contention in Fragmentation-Free Processor Allocation. Proceedings of the 10th International Conference on Parallel and Distributed Computing Systems, pp. 120-124.

Min, D., and Mutka, M. (1994). A multipath contention model for analyzing job interactions in 2-D mesh multicomputers. Proceedings of 8th International Parallel Processing Symposium, pp. 744-751. Cancun.

Min, G. (2003). Performance Modelling and Analysis of Multicomputer Interconnection Networks. Ph.D. Thesis, Department of Computing Science, University of Glasgow, Glasgow, U.K.

Mohapatra, P. (1998). Wormhole Routing Techniques for Directly Connected Multicomputer Systems. ACM Computing Surveys, 30(3), pp. 374-410.

Moore, S., and Lionel, M. (1996). The Effects of Network Contention on Processor Allocation Strategies. In Proceedings of the 10th International Parallel Processing Symposium, pp. 268-274.

Ni, L., and McKinley, P. (1993). A survey of wormhole routing techniques in direct networks. IEEE Computer, 26(2), pp. 62-76.

Noakes, M., Dally, W. J., and Wallach, D. A. (1993). The J-machine multicomputer: an architecture evaluation. Proceedings of the 20th International Symposium Computer Architecture, pp. 224-235. New York, NY, USA: ACM.

Peterson, C., Sutton, J., and Wiley, P. (1991). iWarp: a 100-MOPS, LIW microprocessor for multicomputers. IEEE Micro, 11(3), pp. 26-29.

ProcSimity V4.3 User's Manual, University of Oregon, 1997.

Seo, K.-H. (2005). Fragmentation-efficient node allocation algorithm in 2D mesh-connected systems. Proceedings of the 8th International Symposium on Parallel Architecture, Algorithms and Networks (ISPAN'05), pp. 318-323. Washington, DC, USA: IEEE Computer Society Press.

Suzaki, K., Tanuma, H., Hirano, S., Ichisugi, Y., Connelly, C., and Tsukamoto, M. (1996). Multi-tasking method on parallel computers which combines a contiguous and a non-contiguous processor partitioning algorithm. Proceedings of the 3rd International Workshop on Applied Parallel Computing, Industrial Computation and Optimization , pp. 641-650. London: Springer.

Valero-Garcia, M., A. Gonzalez, L.D. Cerio, D. Royo (1997).Divide-and-Conquer Algorithms on Two-Dimensional Meshes, Technical Report No. UPC-DAC-1997-30, Department of Computer Architecture, Polytechnique University of Catalunia,.

Wan, M., Moore, R., Kremenek, G., and Steube, K. (1996). A batch scheduler for the Intel Paragon with a non-contiguous node allocation algorithm. Proceedings of the

Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '96, pp. 48-64.

Berlin, Heidelberg: Springer Berlin Heidelberg.

Windisch, K., Miller, J., and Lo, V. (1995). ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems. Proceedings of the 5th Symposium on the Frontiers of Massively Parallel Computation (Frontiers'95), pp. 414-421. Washington, DC, USA: IEEE Computer Society Press.

Yoo, B.-S., and Das, C.-R. (2001). Efficient Processor management schemes for mesh-connected multicomputer. Elsevier Science B.V, parallel computing, 27(1) pp. 1057-1078.

Yoo, B.-S., and Das, C.-R. (2002). A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers. IEEE Transactions on Parallel and Distributed Systems, 51(1), pp. 46-60.

Zhu, Y. (1992). Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers. Journal of Parallel and Distributed Computing, 16(4), pp. 328-337.

الملخَّص

تقييم أداء استراتيجيات التخصيص المتجاور وغير المتجاور بناء على أنماط االتصال المعروفة في النظام ثنائي ألبعاد في متعددات الحواسيب

للباحثة عرين فالح احمد العباس

المشرف الرئيسي ألستاذ الدكتور إسماعيل عبابنة المشرف المساعد ألستاذ الدكتور سعد بني محمد

تشير الدراسات السابقة المختلفة في تخصيص المعالجات إلى أن استراتيجيات التخصيص غير المتجاور تتفوق بشكل كبير على استراتيجيات التخصيص المتجاور من حيث متوسط استخدام النظام ومعدل مكوث المهام في النظام، بغض النظر عن نمط االتصال المستخدم، ولكن هذا في الواقع ليس صحيحًا تمامًا، حيث يمكن أن يكون لنمط االتصال المستخدم تأثيراً كبيراً على أداء التخصيص المتجاور وغير المتجاور في متعددات الحواسيب الشبكية، خاصة عندما تقوم كل مهمة بتكرار واحد لنمط االتصال المحدد. في هذه ألطروحة، تمت إعادة النظر في أداء استراتيجيات التخصيص المعروفة في متعددات الحواسيب الشبكية المتصلة بالشبكة ثنائية ألبعاد مع مراعاة العديد من أنماط االتصال المهمة. وهي الجار القريب والكل للكل والواحد إلى الكل والكل إلى واحد والدائري والتقسيم والتجميع ذو الحدين)DQBT(وتحويل فورييه السريع)FFT(والعشوائي كأنماط اتصال. تم استخدام االستراتيجيات) First Fit, Best Fit(كممثل الستراتيجيات للتخصيص المتجاور واالستراتيجيات)GABL, Paging, MBS(كممثل الستراتيجيات للتخصيص غير المتجاورة كما تم استخدام توزيعين لحجم المهمة هما التوزيع الموحد والتوزيع المتناقص. أجريت تجارب محاكاة شاملة لمقارنة أداء التخصيص المتجاور بالتخصيص غير المتجاور من حيث متوسط استخدام النظام ومعدل مكوث المهام في النظام باستخدام المحاكي ProcSimity.

كشفت نتائج المحاكاة في أنماط االتصال الجار القريب و FFT و DQBT، أن أداء استراتيجيات التخصيص المتجاور)FF,BF(يتفوق بشكل كبير على أداء جميع استراتيجيات التخصيص غير المتجاور) MBS,GABL(,(0) Paging من حيث معدل مكوث المهام في النظام، باستثناء MBS في نمط االتصال DQBT، حيث يقترب اداء ال MBS من أداء FF وBF. تثبت هذه النتائج أن الحقيقة التي تقول بأن استراتيجيات التخصيص غير المتجاور تتفوق دائمًا على استراتيجيات التخصيص المتجاور فيما يتعلق بمعدل مكوث المهام في النظام غير صحيحة تمامًا. كما بينت نتائج المحاكاة أنه في أنماط االتصال الواحد إلى الكل والعشوائي والدائري والكل إلى واحد، تتفوق استراتيجيات التخصيص غير المتجاور) Paging (MBS,GABL ,(0) بشكل كبير على استراتيجيات التخصيص المتجاور) FF

BF,(من حيث معدل مكوث المهام في النظام،وفيما يتعلق بنمط االتصال الكل للكل، بينت نتائج المحاكاة أن أداء استراتيجيات التخصيص المتجاور)FF,BF(أفضل من استراتيجيات التخصيص غير المتجاور MBS ولكن GABL و(0) Paging أفضل من FF و BF و MBS.

كما تظهر نتائج المحاكاة، انه في جميع أنماط االتصاالت، تتفوق استراتيجيات التخصيص غير المتجاور بشكل كبير على استراتيجيات التخصيص المتجاور في ما يتعلق بمتوسط استخدام النظام.

Appendix

Appendix

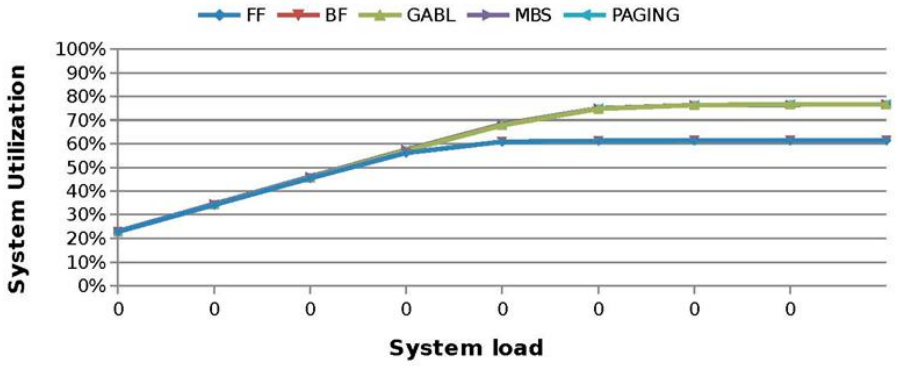

Figure 3.19: Mean system utilization vs. system load for the One-to-All communication pattern and uniform side lengths distribution in a 16×16 mesh

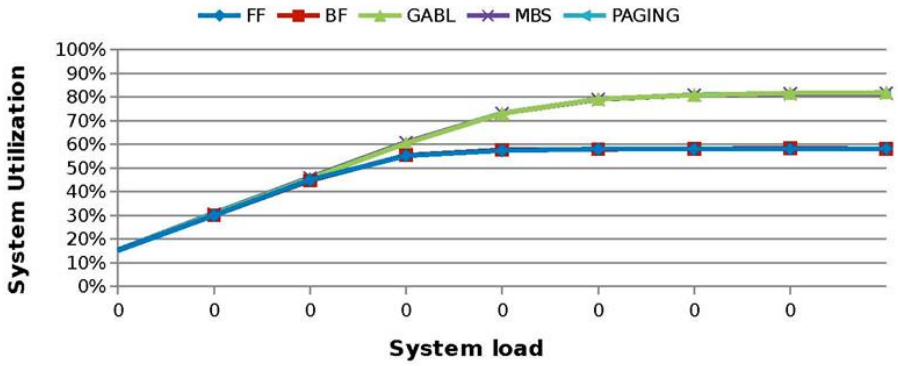

Figure 3.20: Mean system utilization vs. system load for the One-to-All communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

59

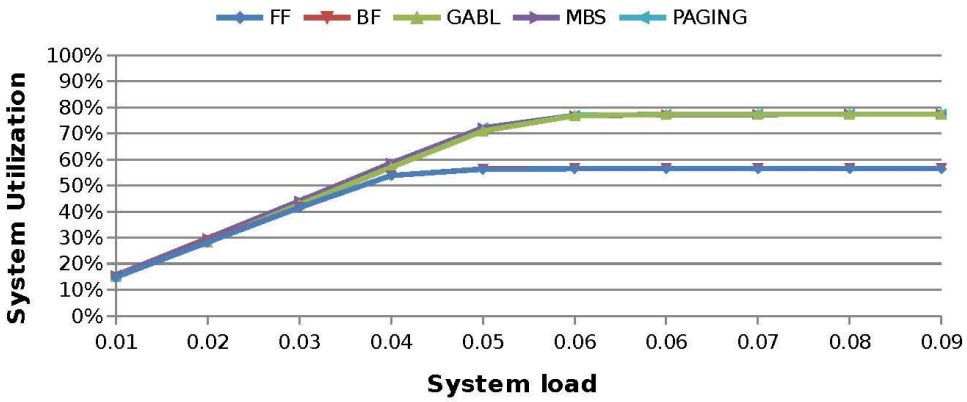

Figure 3.21: Mean system utilization vs. system load for the Random communication pattern and uniform side lengths distribution in a 16×16 mesh

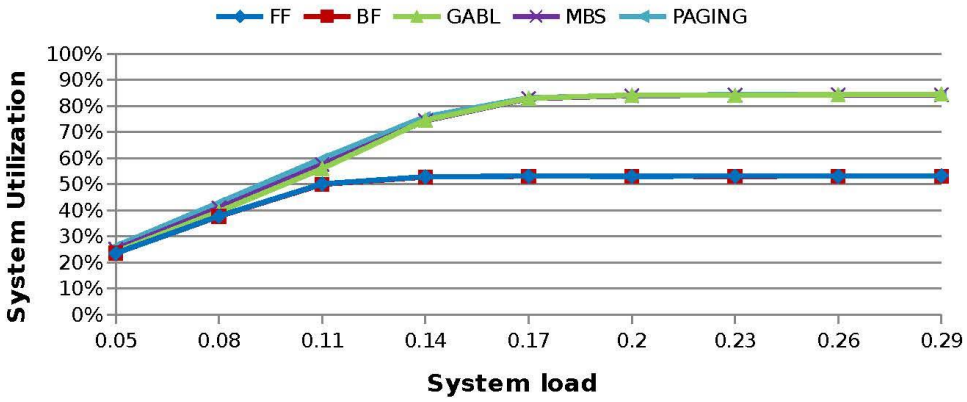

Figure 3.22: Mean system utilization vs. system load for the Random communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

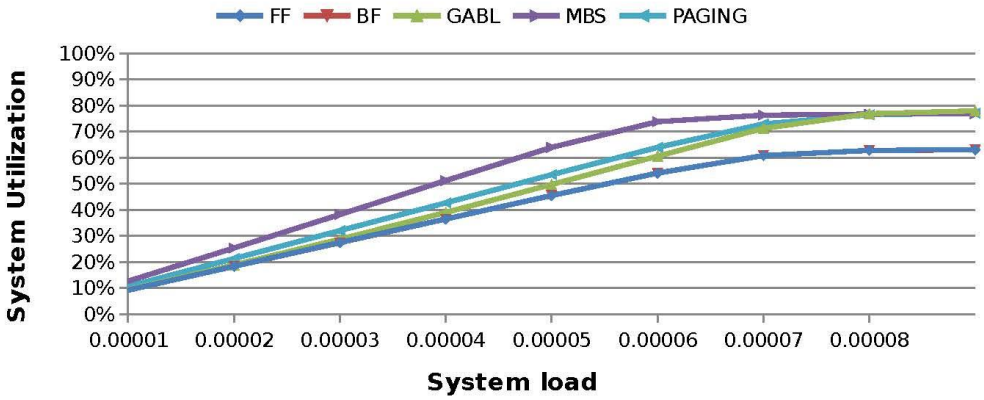

Figure 3.23: Mean system utilization vs. system load for the All-to-All communication pattern and uniform side lengths distribution in a 16×16 mesh

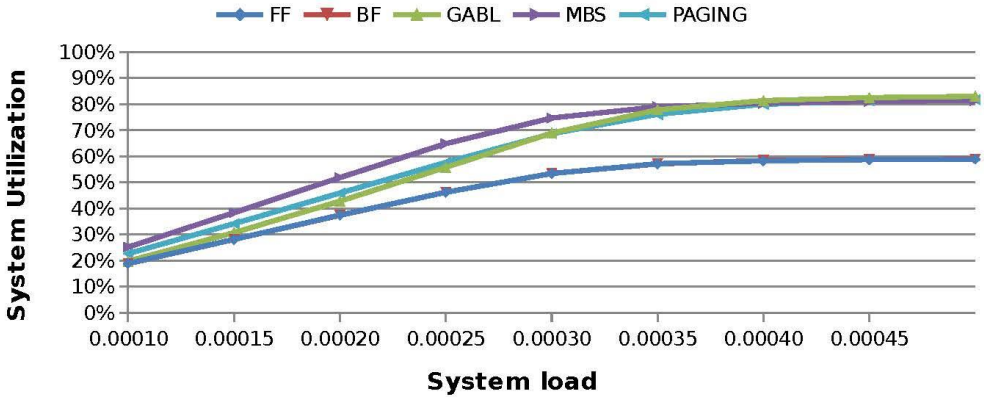

Figure 3.24: Mean system utilization vs. system load for the All-to-All communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

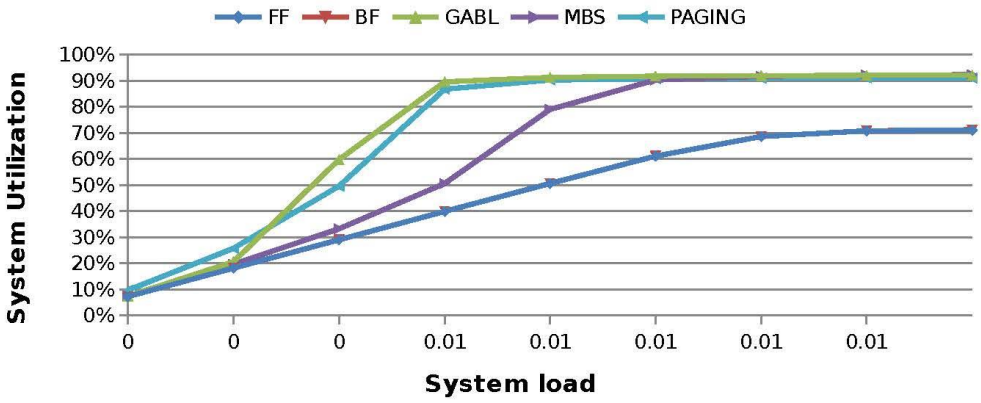

Figure 3.25: Mean system utilization vs. system load for the FFT communication pattern and uniform side lengths distribution in a 16×16 mesh

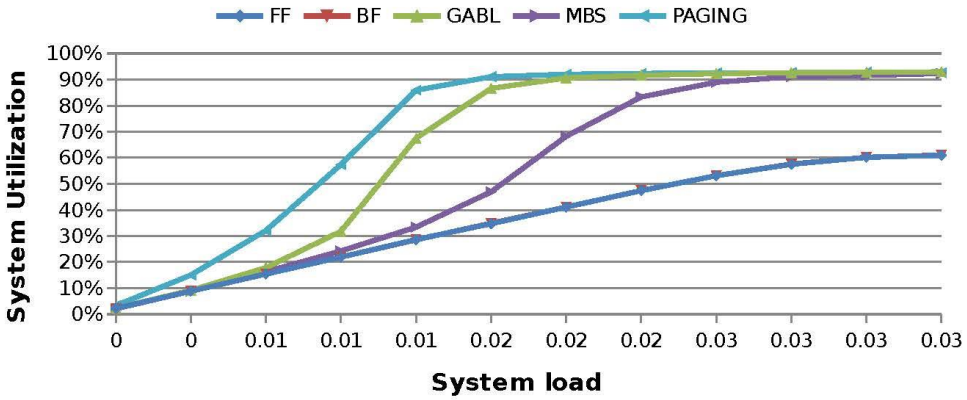

Figure 3.26: Mean system utilization vs. system load for the FFT communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

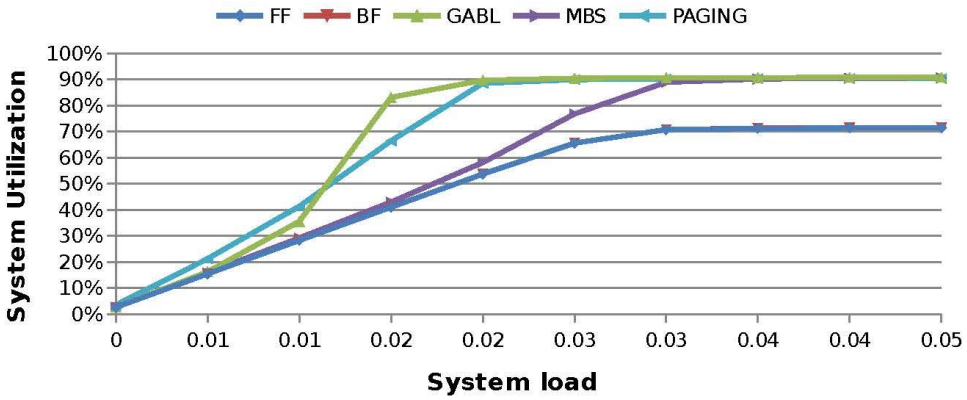

Figure 3.27: Mean system utilization vs. system load for the DQBT communication pattern and uniform side lengths distribution in a 16×16 mesh

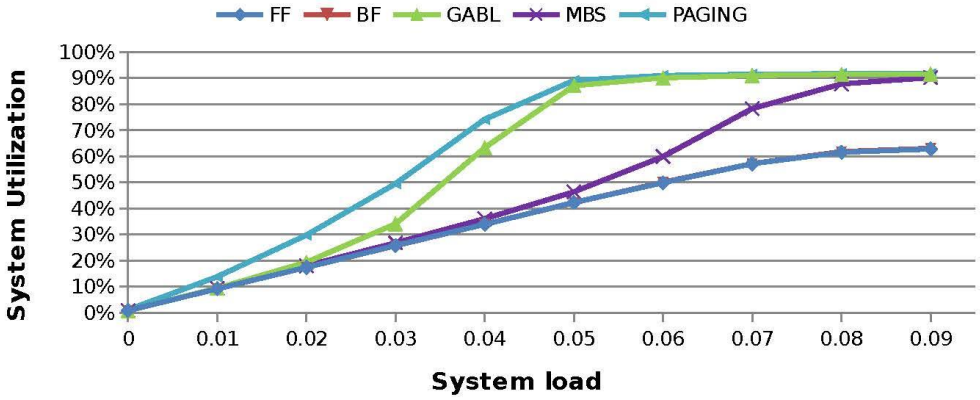

Figure 3.28: Mean system utilization vs. system load for the DQBT communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh.

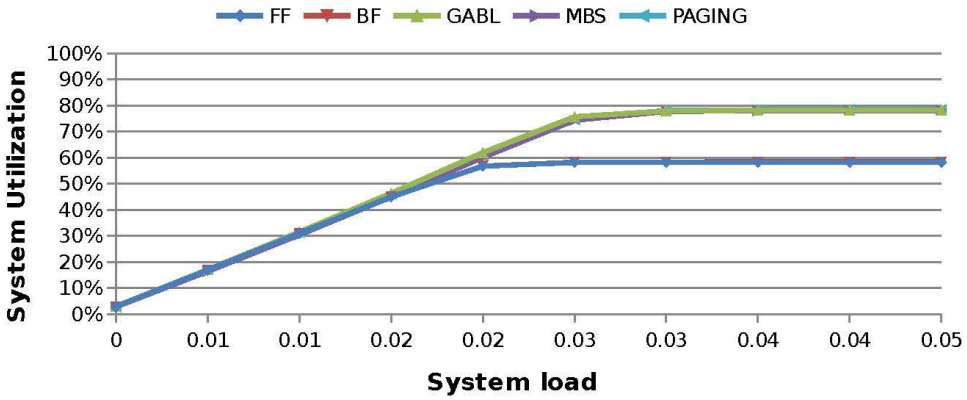

Figure 3.29: Mean system utilization vs. system load for the Ring communication pattern and uniform side lengths distribution in a 16×16 mesh

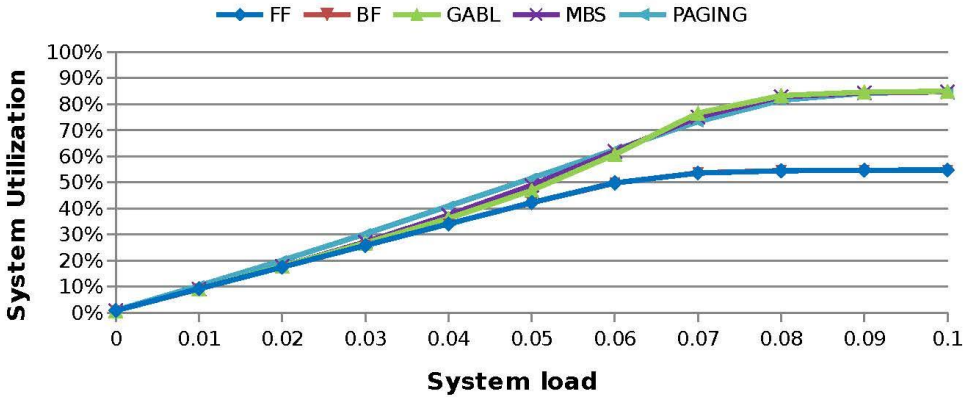

Figure 3.30: Mean system utilization vs. system load for the Ring communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

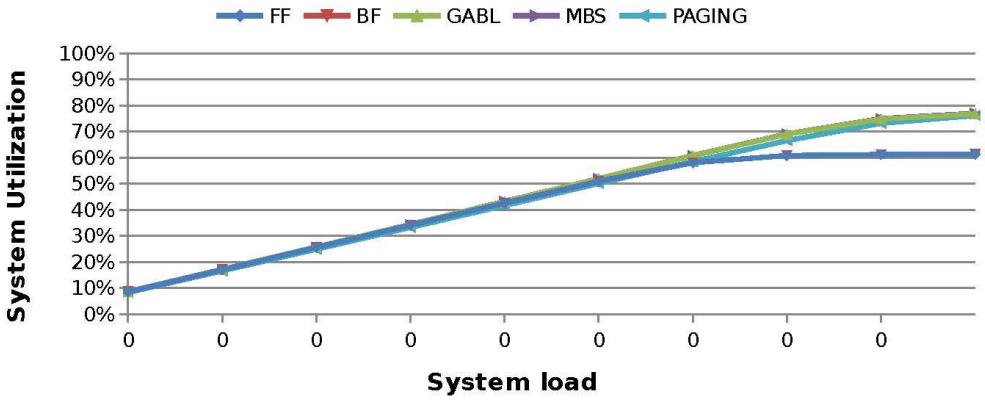

Figure 3.31: Mean system utilization vs. system load for the All-to-One communication pattern and uniform side lengths distribution in a 16×16 mesh

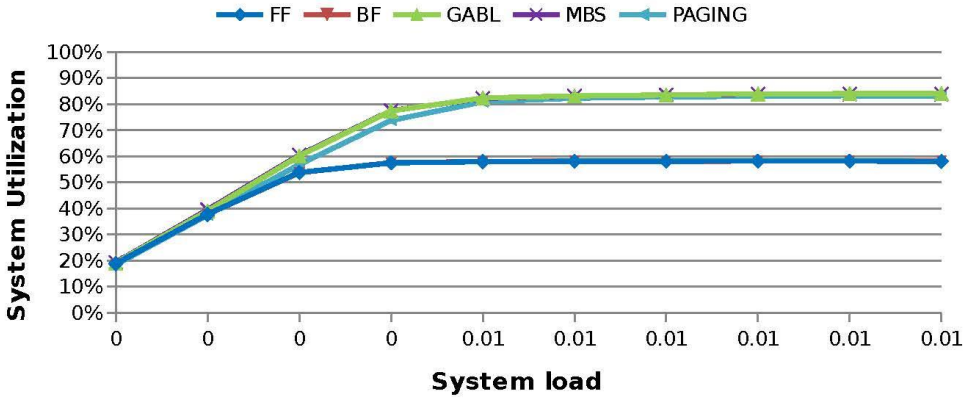

Figure 3.32: Mean system utilization vs. system load for the All-to-One communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh

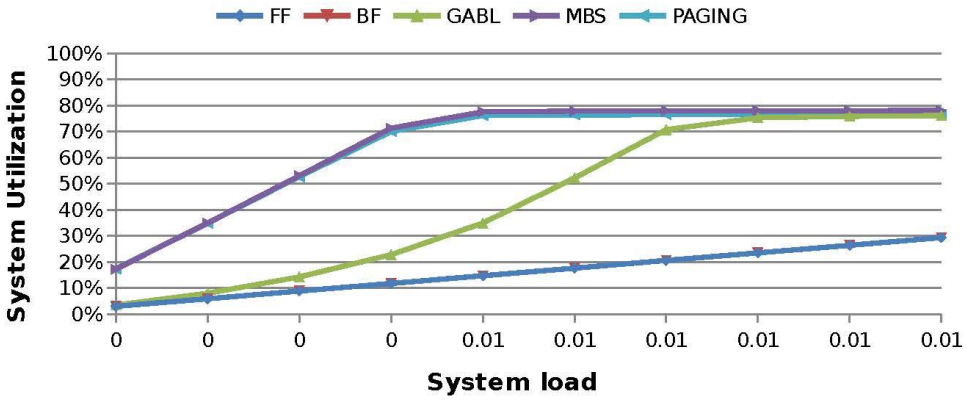

Figure 3.33: Mean system utilization vs. system load for the near neighbor communication pattern and uniform side lengths distribution in a 16×16 mesh

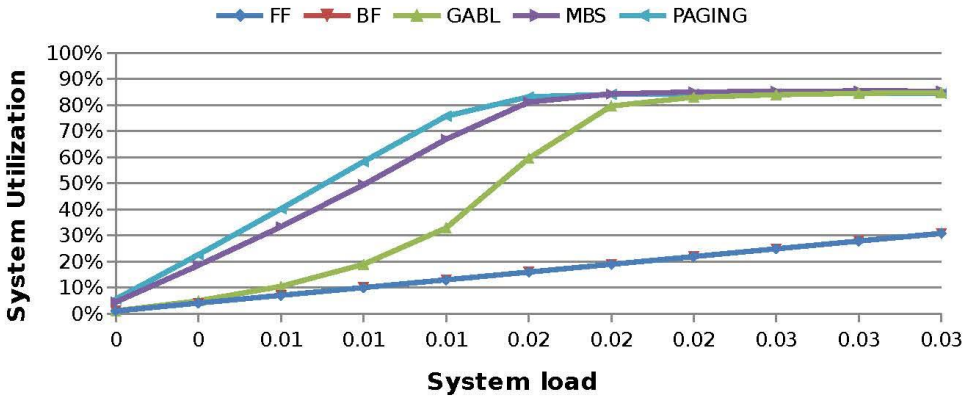

Figure 3.34: Mean system utilization vs. system load for the near neighbor communication pattern and uniform decreasing side lengths distribution in a 16×16 mesh