Communication Impact on Non-Contiguous Allocation Strategies for 2-D Mesh Multicomputer Systems

By

Zaid Mustafa Al-Lami
(400910419)

Supervisors

Dr. Sulieman Bani-Ahmad, Prof. Reyadh Naoum

Master Thesis

Submitted in partial fulfillment of the Requirements of the Master Degree in Computer Information Systems

Faculty of Information Technology
Middle East University

Amman-Jordan

May 2011/2012

Middle East University

Authorization Statement

إقرار تفويض

أنا زيد مصطفى عبد الفتاح اللامي ،أفوض جامعة الشرق الأوسط بتزويد نسخ من رسالتي للمكتبات أو الهيئات أو الأفراد عند طلبها.

التوقيع:

التاريخ: ١٣/٦/٢٠١٢م

I, Zaid Mustafa Abed Alfatah Allami, authorize Middle East University to supply hardcopies and electronic copies of my thesis to libraries, establishments, or bodies and institutions concerned with research and scientific studies upon request, according to the university regulations.

Name: Zaid Mustafa Abed Alfatah Allami

Date: 13/6/2012

Signature:

Middle East University
Examination Committee Decision

This is to certify that the thesis entitled **"Communication Impact on Non-Contiguous Allocation Strategies for 2-D Mesh Multicomputer Systems"** was successfully defined and approved on 2012.

Supervisor: Dr. Sulieman Ahmad Bani-Ahmad

Department Computer Information Systems

Al-Aalqa Applied University

.....

Prof. Alaa Fathi Shata

Department of Computer Science

The World Islamic Sciences & Education University

.....

Dr. Sherif Mahrous Jad

Middle East University

.....

Acknowledgments

In the name of Allah the Most Gracious, the Most Merciful

My guidance cannot come except from Allah, in Him I trust, to Him I repent, and Him I praise and thanks always go.

I offer my sincerest gratitude to my advisors, Dr. Suleiman Bani-Ahmad and Prof. Reyadh Naoum, for their valuable contributions, knowledge, encouragement and helpful advices. As well as their vision that brought this work forward, for being there any time I knocked at their door. I wish them both more and more success and giving.

I am highly indebted to my parents; they raised me well, encouraged and gave me hope and unconditional love. I wish both of them happiness and good health. Thanks are due to my brother, sisters, relatives and friends for their supporting.

Also very special thanks are due to my mother; without her support, this thesis couldn’t have been done.

Finally, I would like to take a moment to thank my university "Middle East University", lecturers and employees, for the moral support and encouragement during my entire graduate studies.

Dedication

I dedicate this work to my mother for her love, understanding and support, she were the light in my path.

Without her, nothing of this would have been possible.

Thank you for everything, I love you!

Table of Contents

List of Figures

Page number

List of Tables:

List of Abbreviations

(ART) Average Response Time

(ASU) Average system utilization

(APBT) Average Packet Blocking Time

(APL) Average Packet Latency

(FCFS) First Come, First Serve

(LCLS) Last Come, Last Server

(SSD) Shortest Service Demand First

(SHT) Shortest Hold Time First

(LSD) Longest Service Demand First

(LHT) Longest Hold Time First

(NAS) Numerical Aerodynamic Simulation

(DC) Divide and Conquer

(FFT) Fast Fourier Transform

(PE) Processing Element

(NPB) NAS parallel Bench marks

(CFD) Computational Fluid Dynamics

(MJST) Mean Job Service Time

(FF) First-Fit

(MPL) Mean Packet Latency

(BF) Best-Fit

(MPBT) Mean Packet Blocking Time

(MJRT) Mean Job Response Time

(MBS) Multiple Buddy System

(BGP-BF) Bounded Gradual Partitioning Best Fit

(DQBT) Divide and Conquer Binomial Tree

(TDB) Two Dimensional Buddy

(PSU) Percent System Utilization

(FS) Frame Sliding

(AS) Adaptive Scan

(IS) Integer Sort

(EP) Embarrassingly Parallel

(CG) Conjugate Gradient

(MG) Multi-Grid

(FT) Discrete 3D Fast Fourier Transform

(BT) Block Tri

(SP) Scalar Penta

(LU) Lower-Upper Gauss

Abstract

Many processor allocation strategies have been proposed in literature. The amount of communication conducted between parallel jobs to be allocated is the key performance factor that can highlight the difference between these strategies, because it affects the performance (e.g., System utilization, Throughput, Response time) of multicomputer systems. This research aims to experiment different types of communication patterns, select processor allocation strategies and depict impacts of using these communication patterns on the performance of the processor allocation strategies that are tested. Wide range of communication patterns were tested, evaluated and compared to the communication patterns used in previous researches. These are the one-to-all and all-to-all communication patterns. We add the following patterns to these two communication patterns: (i) FFT "Fast Fourier Transform" pattern, (ii) NAS "Numerical Aerodynamic Simulation" pattern and (iii) DQBT "Divide and Conquer Binomial Tree" pattern. ProcSimity tool has been used as simulation environment, because it has been used in similar research in literature, and because of its dynamic nature allowing researcher to expand or invent new communication patterns and allocation strategies. Results are considered valid if the confidence level is more than 95% and the relative error below 5%. All allocation strategies were experimented on simulated Mesh topology of 32x32 computing units, and First-Come-First-Serve to choose next process to run. Results of this research show that the communication behavior of the allocated parallel jobs can have a significant impact on the performance of the applied processor allocation strategy such as (MBS, Random, FF, BF, BGP-BF, and BGP-FF).

الملخص

العديد من إستراتيجيات الحجز أقترحت في الماده المطبوعه؛ إن إدارة الشبكات والإتصال بين المهمات "الأعمال" التي تريد أن تحجز هو عامل الكفاءه الرئيسي وذو أهميه خاصة في الإختلاف في هذه الإستراتيجيات ؛لأنه يؤثر على كفاءة أنظمة الحسابات المتعدده (إستخدام النظام,الإنتاجيه,وقت الإستجابه)هدف هذا البحث تجربة أنواع مختلفه من أنماط الإتصال ,إختيار إستراتيجيات حجز المعالج ووصف التأثيرات لإستخدام هذه الأنماط على كفاءة هذه الإستراتيجيات التي تم إختبارها,تم إختبار أنماط الإتصال على نطاق واسع,وقد تم مقارنة وتقييم أنماط الإتصال التي أستخدمت في الأبحاث السابقه ؛هذه الأنماط هي ما يسمى All-to-All و One-to-All , وقد أضفنا الأنماط التاليه على هذين النمطين.(FFT,NAS,DQBT)

إن أداة الProcSimity تم أستخدامها كبيئة محاكاه ,لأنها مستخدمة في مثل هذا البحث في المواد المطبوعه"الدراسات ",وبسبب طبيعتها الديناميكيه حيث تسمح للباحث بتوسيع وإضافة أنماط إتصال وإستراتيجيات حجز,النتائج تعتبر صحيحه إذا كان مستوى الثقه أكثر من 95% ونسبة الأخطاء أقل من 5%,جميع إستراتيجيات الحجز أختبرت على محاكاة ال Mesh Topology بحجم 32*32 وحدات حسابيه ,ويتم أختيار المهمه القادمه لتشغيلها عن طريق ما يسمى يأتي أولاً ينفذ أولاً,نتائج هذا البحث بينت أن سلوك الأتصال للمهمات المتوازيه المحجوزه ذات تأثير هام على كفاءة أستراتيجيات حجز المعالج مثل , BF, FF, Random, MBS .BGP-BF,BGP-FF

Chapter one: Introduction

1.1 Overview

Nowadays, Multi-computer systems are widely used, because of being cost-effective alternatives of the traditional supercomputers (Chang and Mohapattra, 1998). The topology of a multi-computer is defined as a style based on the processing units of this interconnected multi-computer. The mesh-based topologies are considered one of those topologies, such as the two-dimensional (2-D) and the three-dimensional topologies (see figure 1.1). These are the most common topologies; according to being simple, regular and scalable (Ababneh, 2006); (Bani-Mohammad et al, 2007); (Bani-Ahmad (a), 2011).Most of the modern commercial and experimental multi-computer systems (such as the IBM BlueGene/L and the Intel Paragon) have been built based on (2-D)architecture (Chang, and Mohapattra, 1998); (Bani-Ahmad (a), 2011).

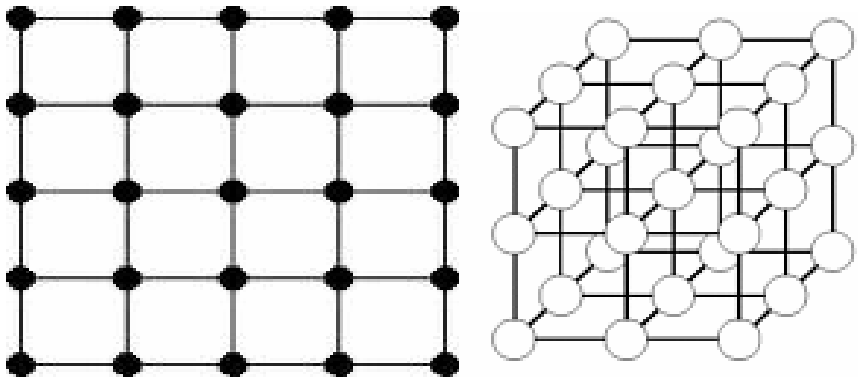


Figure 1.1: 2-D 5x5 mesh and 3-D 3x3 mesh multi-computers.

1.2 Processor Allocation in Multi-computers

In a multi-computer system, the processor allocator applies processor allocation strategies in order to assign unallocated multi-computer nodes to parallel jobs and identify them (Chang and Mohapatra, 1998). Both of processor allocation algorithms and better idle-nodes recognition ability for

multi-computers or idle sub-meshes can significantly reduce the job waiting time and improve the chance of assigning a parallel job into the system (Chang and Mohapatra, 1998).

The proper choice of the processor allocation strategy in 2-D-Mesh multi-computer is considered a critical issue in deciding the performance of a given multi-computer system. Because it significantly affects performance (system utilization, response time, throughput … etc) of any parallel system (Chang and Mohapattra, 1998; Bani-Ahmad(a), 2011).

Allocation strategies fall into two main categories, as follows:

- First: contiguous allocation strategies seek to allocate a sub-mesh, such as a contiguous set of processing units of the same size and shape of parallel request. The contiguity condition can be summarized in the relaxing of the non-contiguous allocation strategy (Bani-Ahmad (a), 2011). Contiguous allocation strategies have low system utilization and high internal and external fragmentation, which is considered a problem similar to the problem to the contiguous allocation strategies.
- Second: non-contiguous strategies can produce high execution times of parallel jobs according to high communication latencies (Bani-Ahmad(a-c), 2011). In general, talking about Latency means a measurement of time delay experienced in a system, the precise definition of which is based on the system and the time being measured. Noting that, latencies may have different meanings in different contexts (en.wikipedia.org, 2011).

The fragmentation problem prevents utilization of idle processors. It can be classified as internal and external fragmentation. Internal fragmentation is the result of allocating jobs only to certain size sub-meshes. It occurs when a job

is assigned to more than required processors, and when the extra nodes allocated are not used for actual computation and are wasted (Chang and Mohapattra, 1998). Moreover, it happens when a job requests a sub-mesh that does not fit requirement of the allocation algorithm. For example, internal fragmentation occurs when a job does not require a square sub-mesh with sides equal to power of two and is allocated using the TDB scheme.

The external fragmentation happens when the allocation scheme cannot allocate the available processors to the incoming jobs (Chang and Mohapattra, 1998).

In addition, the non-contiguous allocation strategies try to fix problems of external and internal fragmentation, along with the low system utilization by allowing parallel requests to be partitioned and allocated non-contiguously into smaller sub-frames in case contiguous allocation fails (Chang and Mohapattra, 1998; Bani-Ahmad(a), 2011; Bani-Ahmad, 2010). Studies showed that allocation strategies integrate advantages of both contiguous and non-contiguous allocation strategies through preserving some level of contiguity within allocated parallel job (Bani-Ahmad, 2010).

Chapter Two: Literature Review and Related Work

2.1 Introduction

The performance of parallel systems can be significantly affected by the processor allocation strategy used in the 2D-Mesh multicomputer system in hand (Ababneh, 2006). Some studies have proposed allocation strategies. All proposed strategies can fall under two categories: namely; contiguous and non-contiguous strategies.

2.2 Contiguous Processor Allocation Strategies

In contiguous processor, allocation of a given parallel requires a contiguous set of processing units of the same size and shape of that request to be assigned to job of interest (Yoo and Das, 2002; Ababneh, 2006).

In contiguous allocation strategies, low level of system utilization can be raised and external and/or internal fragmentation can be observed (Chang and Mohapattra, 1998; Lo et al, 1997). When a job needs to be allocated to a number of processors, it *may* be assigned to more processing units than what it actually requires. This is referred to as the *internal fragmentation* problem (Chang and Mohapattra, 1998; Bani-Mohammad et al, 2007). *External fragmentation*, on the other hand, occurs when enough number of idle processors is available in the system but cannot be allocated to the scheduled parallel job because of the *necessity of contiguity* (Ababneh, 2006; Bani-Ahmad (a), 2011). Many research efforts tried to solve or decrease the problem of external fragmentation (Chang and Mohapattra, 1998; Bani-Ahmad(a), 2011; Mao, Chen and Watson, 2005), and one proposed solution was to use non-contiguous allocation strategies.

All the above results of allocation strategies are referred to as contiguous allocations because they consider only contiguous regions for the execution

of a parallel job. In contiguous allocation, communication cost is minimal (Bani-Ahmad, 2011). However, the requirement is that a parallel job has to be allocated to contiguous set multicomputer reduces the chance of successfully allocating the job due to the problem of fragmentation (Bani-Ahmad, 2011; Bani-Mohammad et al., 2006).

- ✓ **Internal fragmentation – Example:**

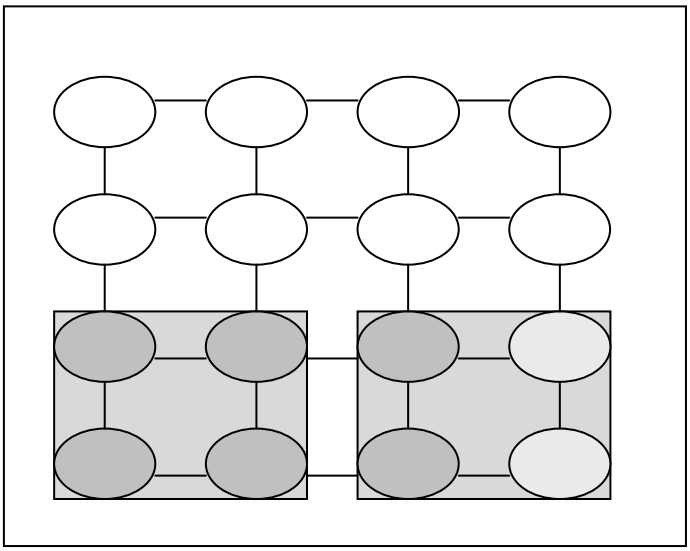


Figure 2.1: Internal fragmentation problem

Where:

Free node  Allocated node 

The important notes about figure 2.1: are that the parallel job has requested 6 processing elements. In the allocation strategy used above, the mesh of the system (of size 4x4) is divided into pages or blocks, each page is of size 2x2. Parallel requests are assigned to jobs in blocks. To fulfill this request, two blocks (a total of 8 nodes) are assigned to this job. Only 6 of these nodes are used. The remaining two are not used by the job and still cannot be assigned to any other job. This is referred to as internal fragmentation.

- ✓ **External fragmentation – Example:**

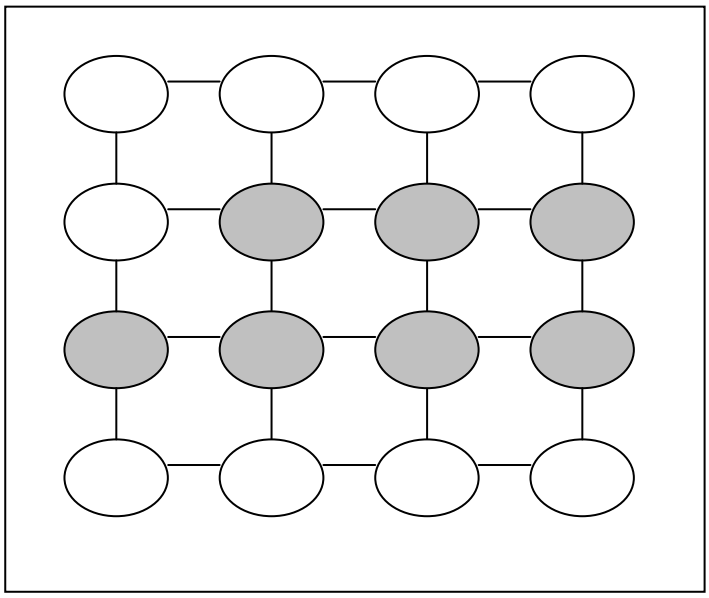


Figure 2.2: External fragmentation problem.

The important note about figure 2.2: is that the parallel job has requested 3x2 processing elements. In the allocation strategy used above, a contiguous sub-mesh of the same size and shape of that request is required to fulfill it. Despite the fact that the system does have enough idle nodes, the request cannot be fulfilled due to the condition of contiguity. This phenomenon is referred to as external fragmentation.

Below is a list of existing contiguous processor allocation strategies that can be found in literature:

- ✓ **Two Dimensional Buddy (2DB)** (Li and Cheng, 1991)**:** in the Two Dimensional Buddy (2DB) strategy, the system is assumed to be a square with side lengths equal to a power of two (2^i, where i=0, 1, 2,…). The size of a requested sub-mesh is *rounded up* to a square with side-lengths as the nearest power of two. The resulting sub-mesh after rounding up can be larger than the original sub-mesh. Consequently, this allocation strategy suffers from *internal fragmentation* because it only allocates square sub-meshes whose side lengths are equal to a

power of two. In other words, because of rounding up the sides of requests, the allocated sub-mesh can be larger than the originally requested sub-mesh.

- ✓ **Example (the Two Dimensional Buddy (2DB) strategy):**

The important note about figure 2.3: is that the parallel job has requested 2x1 processing elements. In the allocation strategy used, a contiguous sub-mesh of size 2x2 the same size and shape of that request is required to fulfill it.

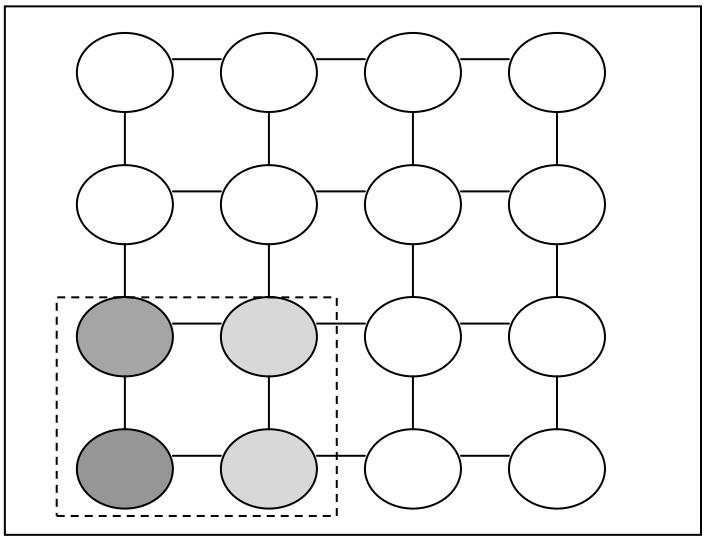


Figure 2.3: External fragmentation problem.

- ❖ **Frame Sliding (FS)** (Chuang and Tzang, 1994)**:** The Frame Sliding (FS) method was proposed to reduce the internal fragmentation problem of the 2DB allocation. This is achieved through allowing sub-meshes of any arbitrary size to be allocated to parallel jobs.

This allocation strategy works as follows: Viewing the requested sub-mesh of the job in hand as a frame, the FS algorithm slides the frame across the system to examine for a free sub-mesh to execute the job (Chuang and Tzang, 1994).

✓ **Example on Frame Sliding algorithm:**

The sub-mesh size equal 4*4 and the job coming requesting a sub-mesh of size 3x2. So the FS algorithm will start at lowest left-most node searching for a free sub-mesh of the same size and orientation of the request in hand. If it fails, the 3x2 frame slides to right direction. The sliding step is equal to the width of the request; that is 2 in our case. If the width of the mesh is covered, the algorithm restarts at column one of the mesh after sliding upward with a step size that is equal to the height of the current request. If the algorithm reaches the top rightmost node of the mesh without finding an idle sub-mesh, it fails.

Note that:

- An idle sub-mesh may be available in the system but this algorithm cannot identify it.
- The algorithm does not do rotation to the request.

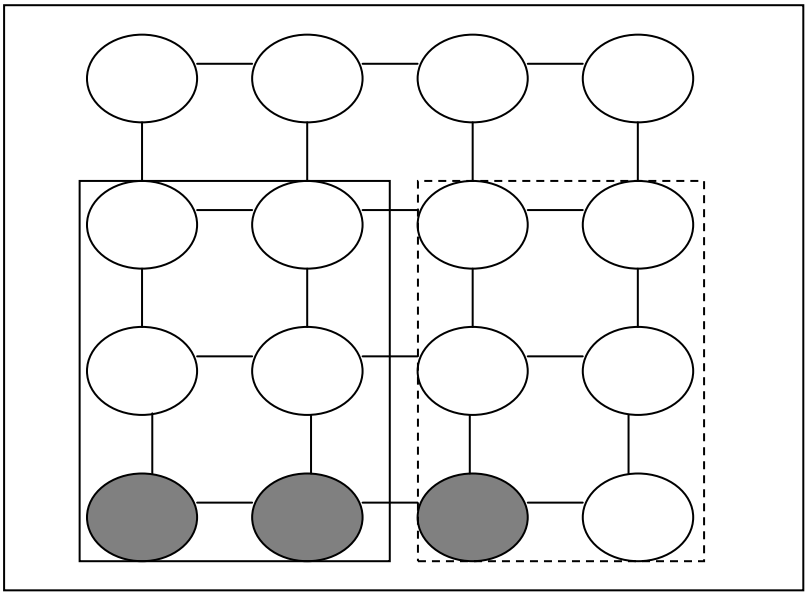


Figure 2.4: The FS algorithm.

❖ **First-Fit (FF) and Best-Fit (BF) Strategies (Zhu, 1992):** The FF and BF algorithms guarantee the recognition of a free sub-mesh, provided

it exists. The two algorithms work by scanning the entire mesh for possible allocation.

- ✓ **Example on the First-Fit (FF) and Best-Fit (BF) algorithms (Figure 2.5):**

The sub-mesh size equal 4*4 and the job coming requesting a sub-mesh of size 2x2. So the FF algorithm will start at lowest left-most node searching for a free sub-mesh of the same size *and* orientation of the request in hand. If it fails, the 2x2 frame slides to right direction. The sliding step is equal to *one*. If the width of the mesh is covered, the algorithm restarts at column one of the mesh after sliding upward with a step size that is equal to one also. If the algorithm reaches the top rightmost node of the mesh without finding an idle sub-mesh it fails. The first free sub-mesh is allocated to the requesting job. Notice that the sub-mesh that will be allocated is the one surrounded with the solid line square. However, it is clear that a better choice is to allocate the idle sub-mesh with the busiest neighbors; that is the sub-mesh with the dashed line square (which is identifiable by the Best Fit approach).

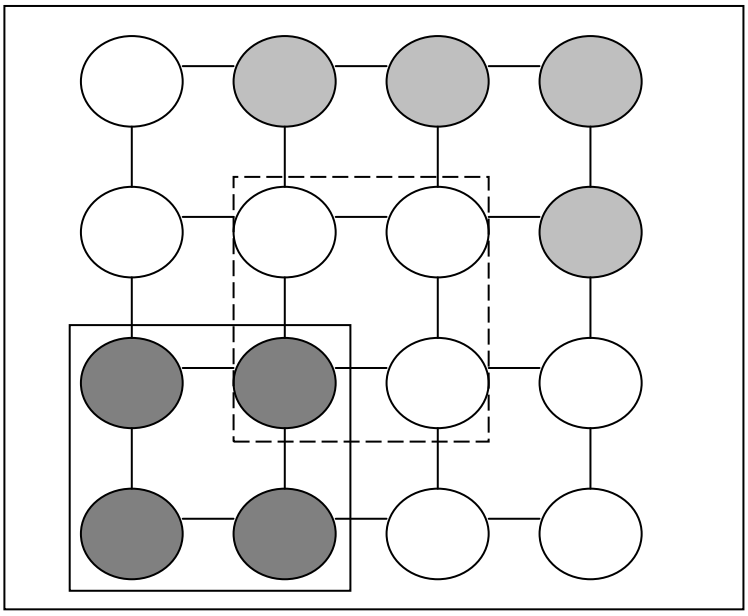


Figure 2.5: The FF and BF algorithms.

- **Adaptive-Scan (AS) Strategy (Ding and Bhuyan, 1993):** The adaptive-scan changes the *orientation* of the sub-mesh being searched for if the required sub-mesh in the original orientation is not available. Thus, the AS strategy has better recognition capabilities than that of the BF and FF schemes.

- ✓ **Example on the AS algorithm (figure 2.6):**

The sub-mesh size equal 4*4 and the job coming requesting a sub-mesh of size 4x1. The AS works just like the FF algorithm; however, it considers rotating the request in case that the orientation requested could not be allocated. In this case, a sub-mesh of size 1x4 will be allocated to the parallel job.

Where: allocated nodes now  allocated nodes in past 

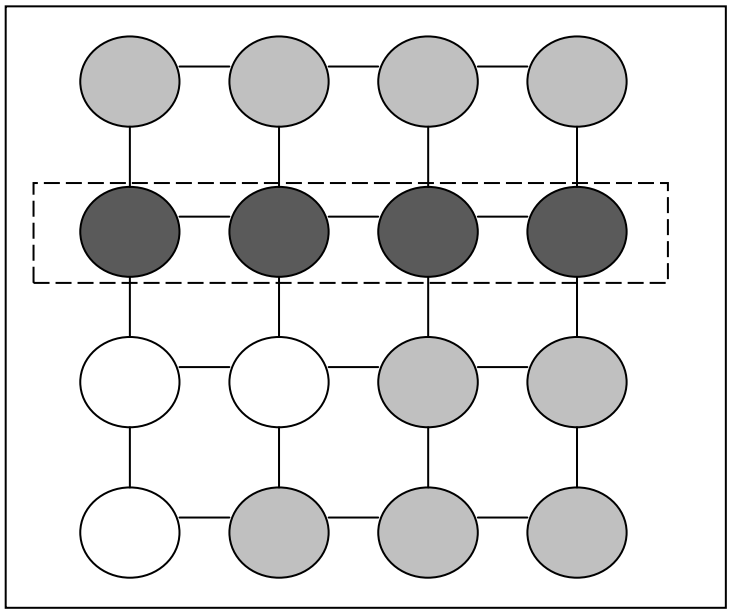


Figure 2.6: The AS algorithm.

2.3 Non-contiguous Processor Allocation Strategies

When using non-contiguous allocation, the contiguity condition is *not* a must (Chang and Mohapattra, 1998); therefore, a job *does not* have to wait for a single sub-mesh of the requested size and shape to be available because it will be executed on a number of different *disjoint* smaller sub-meshes (Chang and Mohapattra, 1998; Suzaki et al, 1996; Srinivasan et al, 2004).

The non-contiguous allocation of requests can successfully solve the drawbacks of contiguous allocation strategies. As experimentally validated, non-contiguous allocation strategies have been found to produce relatively high system utilization and eliminate fragmentation (Chang and Mohapattra, 1998; Suzaki et al, 1996; Srinivasan et al, 2004).

Communication latency is often high in non-contiguous allocation strategies since communication between processors running the same job may be indirect due to non-contiguity (Srinivasan,et al, 2004).

The introduction of wormhole routing (Kumar et al, 2003) encouraged researchers to consider non-contiguous allocation on multi-computers (Chang and Mohapattra, 1998). With wormhole routing, the message communication latency is less dependent on the distance traveled by the message from source to destination (Chang and Mohapattra, 1998); (Kumar et al, 2003).

In non-contiguous allocation schemes, allocation requests of parallel jobs are subdivided into two partitions (Chang and Mohapattra, 1998) if contiguous allocation fails. If allocating either of the two partitions fails, that partition is further subdivided into smaller sub-partions until the allocation successes (Lo (b) et al, 1997). Using wormhole routing has made allocating parallel jobs to non-contiguous processors reasonable in terms of performance even in networks characterized by a relatively

long-diameter, " The diameter is the shortest hop "the maximum of the shortest distance between any two nodes ", a small communication delay happens when an interconnection graph has a small diameter between nodes (Tang, Sanjay and Padubirdi, 1994). Such is the 2-D mesh. So the contiguity condition is relaxed which allowed jobs to be executed without waiting for *sufficient* and *contiguous* set of idle processing nodes to be available in the system (Bani-Mohammad et al, 2007; Bani-Ahmad, 2010; Lo(b) et al, 1997).

When "Wormhole routing" is a special case of *cut-through switching*, it routes the *head* of a *packet* directly from *incoming* to *outgoing* channels of the routing chip. A packet is divided into a number of *flits* (i.e. *flow control digits*) for transmission. The route or the path followed by those flits from source to destination is determined by the *header flits* (or flits). Flits are forwarded through a chosen channel after examining the header flit(s) of a message. As the header flit of a given packet/message advances along a specific route, all following flits follow in through the same route (Seydim, 1998).

There are many important strategies of the non-contiguous allocation such as:

- **Multiple Buddy System (MBS)** expresses the allocation request as a base-4 number, and bases allocation on this expression.

In this strategy, the mesh of the system at hand is divided into non-overlapping square sub-meshes with side lengths that are powers of 2. The number of processors, p, requested by a scheduled job is factorized into a base-4 block. If a required block is unavailable, MBS recursively searches for a larger block and repeatedly breaks it down into four buddies until it produces blocks of the desired size. If that fails, the requested block is further broken into four sub-requests until the job is allocated (Bani-Ahmad, 2010).

- ✓ **Example on MBS strategy (figure 2.7):**

The sub-mesh size equal 4*4 and the job coming is requesting a total of 3 nodes. So, request will be divided into one or more non-overlapping square sub-meshes with side lengths that are powers of 2, thus. In this case, the system sub-mesh that is of size 4x4 is virtually subdivided into 4 blocks, each is of size 2x2 and the request is granted a free sub-mesh of size 2x2.

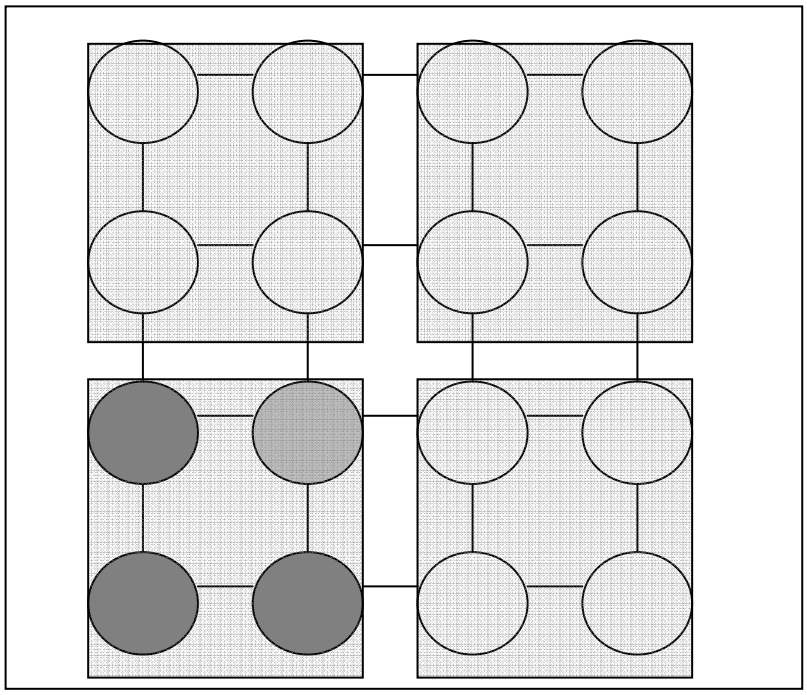


Figure 2.7: The MBS algorithm.

- ❖ In the **Paging allocation strategy**, for instance (Alqadi and Khammash, 2007), the entire 2D mesh is virtually sub-divided into pages or sub-meshes of equal sides' length of 2^i, where i is a positive integer number that represents the index parameter of the paging Approach, that can scan for pages in multiple ways such as snake-line order and row-major order (Bani-Ahmad, 2010).

- ✓ **Example on Paging algorithm:**

Example: Paging – Page size = 1, snake-line order

This type divides mesh into pages with size 2*2 (2^1*2^1), the search manner for free pages is snake-line order as shown in figure 2.8.

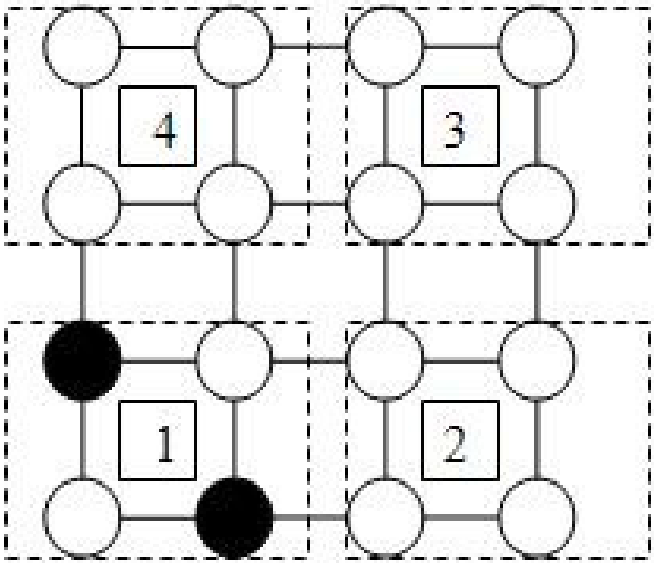


Figure 2.8: Paging with page size=1, snake line order search.

Example: Paging – Page size = 1, row-major order

This type divides mesh into pages with size 2*2 (2^1*2^1).The search manner for free pages is row-major order as shown in figure 2.9.

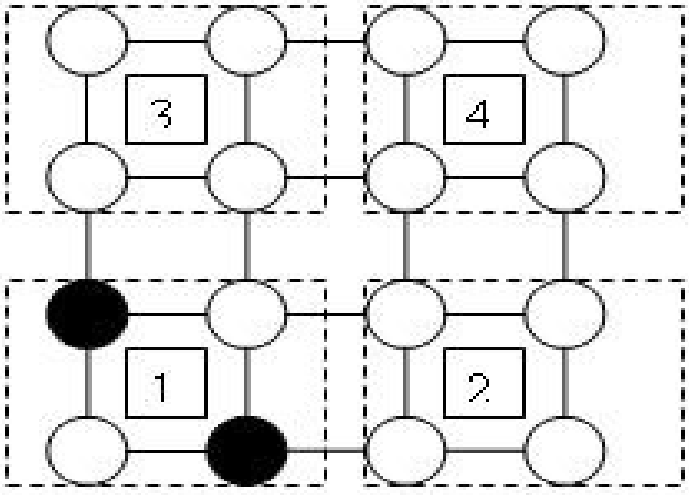


Figure 2.9: Paging with paging size= 1, row-major order search.

A main disadvantage of this strategy is the fragmentation it may cause when some free processors cannot be allocated because they are contained in pages that have been reserved to job (Seydim, 1998).

A partially non-contiguous allocation strategy tries to find a *contiguous* set of processing units of the same size to active job using some contiguous strategy. If it fails, the active job is divided into two sub-requests. The two new sub-requests are then allocated using the same contiguous allocation approach again; this operation will continue recursively until the request is finished. Examples of this allocation strategies are (i) the PALD-FF (***PA***rtitioning at the ***L***ongest ***D***imension with ***F***irst-***F***it), (ii) the PALD-BF (***PA***rtitioning at the ***L***ongest ***D***imension with ***Best***-***F***it) (Bani-Ahmad, 2010) and (iii) the restricted size reduction (RSR) strategy (Bani-Ahmad, 2010), (iv) bounded Gradual-Request-Partioning (BGP) strategy (Bani-Ahmed, 2011a).

The RSR strategy allows jobs to be executed on a reduced size sub-mesh adaptively and partitions it in two sub-blocks of equal size.

In the PALD-FF strategy, First-Fit approach used to find a contiguous group of processing units of the same size and shape of the application at hand. In case of fail, the request at hand is divided into sub-requests after removing one from the longest dimension of the request that is for a given request of size a*b and assuming b>a, the two partion-sizes are a*(b-1)and a*1 after removing one from the longest dimension of the request. The two new sub-requests are then allocated using the first-fit approach again; this procedure continues recursively until the request is fulfilled (Bani-Ahmad, 2010).

Based on the above, the PALD-BF is the same of PALD-FF, but it applies the best fit strategy rather than the first fit, and when using the BGP-BF or BGP-FF, it's the same as the PALD-BF and PALD-FF but the difference is the bound "that’s defining the number of partitions required ".

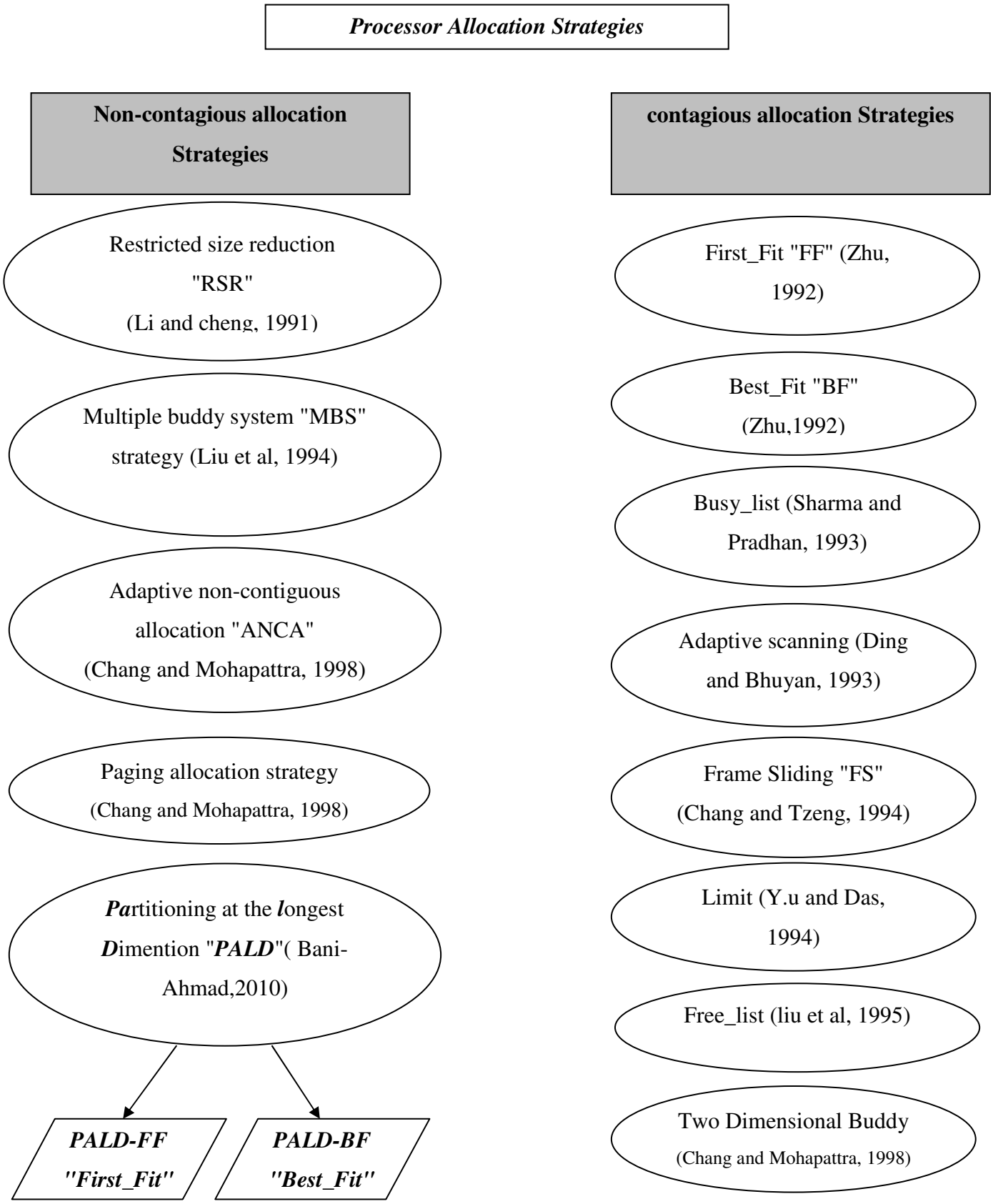


Figure 2.10: Processor Allocation Strategies in Multicomputer Systems

Chapter Three: Methodology and Experimental Setup

3.1 Experimental Objectives

Many processor allocation strategies have been proposed in literature. The amount of communication conducted between the parallel jobs to be allocated is the key performance factor that can highlight the difference between these strategies. The researcher wanted to find out if types of communication patterns affect the performance of these strategies and how.

3.2 Methodology research

In this thesis, the researcher comparatively evaluates number of partially non-contiguous allocation strategies that are proposed in (Bani-Ahmad(a), 2011) and (Bani-Ahmad, 2010) in terms of their performance at high communication loads.

A number of communication styles will be experimented. Examples are: all-to-all and one-to-all communication mechanisms. The allocation strategies of interest will be implemented in the C language, and later integrated with the ProcSimity simulation tool (ProcSimity V4.3 User's Manual, 1997); (Windisch, Miller and Lo, 1995). The processor-allocation strategies will be examined under the scheduling strategy called First-Come-First-Serve as in (Bani-Mohammad et al, 2007); (Bani-Ahmad, 2010).

3.3 Problem Statement

In this thesis, the researcher applied a number of partially non-contiguous allocation strategies. Some of them, the partitioning at longest dimension with best-fit and first-fit, all strategies were explained in previous chapter (Ch. 2). The researcher also evaluated all partially non-contiguous allocation strategies in terms of their performance at high communication loads. A number of communication styles will be experimented (Bani-Ahmad(a), 2011) and (Bani-Ahmad, 2010) in terms of their

performance at high communication loads. Examples on communication patterns are: all-to-all and one-to-all communication mechanisms (see figures 1.2 and 1.3). Other communication patterns are to be examined in our experiments.

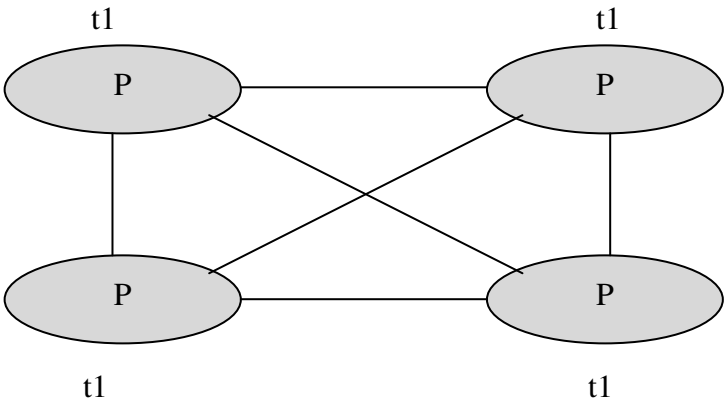


Figure 3.1: One-to-All communication pattern

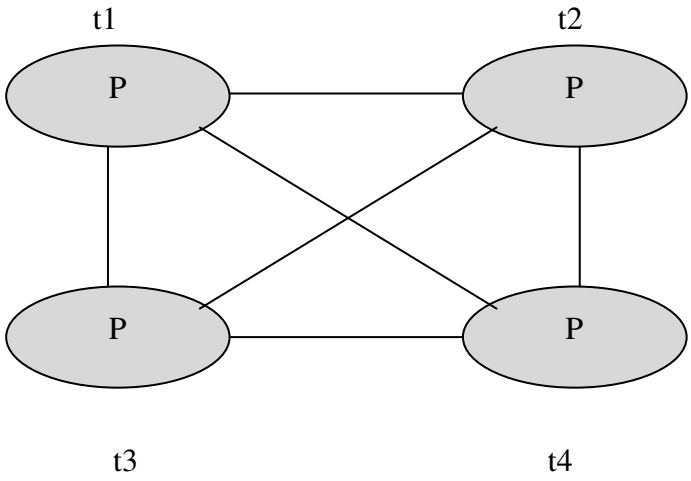


Figure 3.2: All-to-All communication pattern

In brief, this thesis tries to find out the following:

- Behavior of the processor allocation strategy used change as the communication behavior of parallel jobs change.
- Behavior of the processor allocation strategy used change as the communication load caused by the parallel jobs change.

Wide range of communication patterns were tested ,evaluated and compared to what have been done in previous research, which only evaluate one-to-all and all-to-all communication patterns, while current research evaluates FFT "Fast Fourier

Transform" pattern, NAS "Numerical Aerodynamic Simulation" pattern and DQBT "Divide and Conquer Binomial Tree" pattern.

3.4 communication pattern-Overview

A message passing application uses point-to-point and collective operations in order to communicate among the running on nodes, those nodes send messages to the other nodes. Moreover, containing how much data depends on the algorithm implemented by the application, sometimes, it depends on the input and setup of the application. These communications patterns are important for developers and designers of networking and communications software stacks, as well as for purchasers of parallel machines, so it may enable to make decisions on which network topology and technology will best fit their applications (Reisen, 2006)

3.5 Fast Fourier Transform

Some of the important communication patterns are the Fast Fourier Transform (FFT) algorithm, which seems to compute the transform as parallel as possible to do the same task in a shorter time, though it is very important when it comes to parallel computing, limiting the communication overhead so that it would not hide the speedup (Bahn,Yang and Bagherzadeh, 2008).

Our FFT algorithms can be divided into three steps (Bahn, Yang and Bagherzadeh, 2008):

- The first step is processing; where data is rearranged in bit reverse order and classified into p blocks so that N/P data items are sequentially fed to each processing element (PE) from PE[0] to PE[p-1], where p is the total number of PEs and N is the size of data for FFT.
- The second step is actual transformation, during the second step, where the transform is executed; it can be further decomposed into sequential execution

and parallel execution. The sequential FFT refers to the first $\log_2(N/P)$ stages of N-point FFT where N/P -point FFT is performed within each PE since the data that will be transformed reside in the local memory of each PE. Otherwise, the parallel FFT reflects the rest of $\log_2 p$ stages of original N-point FFT where data exchanges needed for the index distance between a butterfly is larger than N/P.

- The third step is post-processing: this step is to make sure of the synchronization of data in data interchanges between processing elements.

Generally, the fast fourier transform (FFT) is defined as the efficient algorithm to compute the discrete fourier transform "DFT" and its inverse. Moreover, there are many distinct FFT algorithms involving a wide range of mathematics. An FFT computes the DFT and resulted in exactly the same result as evaluating the DFT definition directly; the difference between them is that an FFT is much faster than DFT; moreover, many FFT algorithms are much more accurate than evaluating the DFT definition directly (John W. Tukey, 2012).

3.6 Numerical Aerodynamic Simulation

Other important communication patterns are the NAS parallel bench marks (NPB), which is defined as small set of programs designed to help evaluating performance of parallel supercomputers. Benchmarks are derived from computational fluid dynamics (CFD) applications and consist of five Kernels' and three pseudo-applications. The original benchmarks identified in NPB mimic the computation and data movement in CFD applications, the five Kernels' is IS, EP, CG, MG,FT; and the three pseudo applications is BT, SP, LU (NASA Numerical Aerodynamic Simulation Program, 2009); (Bailey,Harris, Saphir et al,1995); (Riesen, 2006).

3.7 Divide and Conquer Binomial Tree

On the other hand, the general idea of the Divide-and-Conquer is recursively partition a problem into subprogram of roughly equal size, if subprogram can be solved independently; there is a possibility to increase the speed by parallel computing (Shimojo et al, 2005).

3.8 ProcSimity software tool

In this thesis, the researcher uses ProcSimity software tool because it is useful to research in the area of processor allocation and processor scheduling for distributed memory multi-computers (Windisch, Miller and Lo, 1995).

The ProcSimity models is considered as a stream of independent user jobs arriving in the system, processor allocation involved with the selection of a set of processors for a newly arrived job. Due to its request for a specific sized block, the allocated processors held by computation for their entire lifetime, then released upon termination of the computation. Processor allocation algorithms consist of a tool and classified into two categories:

A. Contiguous allocation algorithms, that set processors allocated to a specific job from a contiguous region;

B. Non-contiguous allocation algorithms that the processors allocated to a job do not need to be physically adjacent. So a job does not have to wait for a single sub-mesh of the requested size and shape in order to be available because it will be executed on a number of different disjoint smaller sub-meshes. This mechanism depends on the selected strategies, such as Multiple Buddy System (MBS), Paging allocation strategy, the PALD-FF (PArtitioning at the Longest Dimension with First-Fit), the PALD-BF (PArtitioning at the Longest Dimension with Best-Fit), the restricted size reduction (RSR) strategy, BGP_BF and BGP_FF. Processor scheduling in the context of the

current tool refers to discipline scheduling used at the job level. Thus, scheduling controls the next job selection to which processors are to be allocated (ProcSimity V4.3 User's Manua, 1997)

Chapter four: Results and Observations

Results and Observations

Figures 4.1 through 4.10 show the relationship between system load, and more parameters such as the mean job response time and the job service time, mean packet blocking time….etc.

Figures 4.1 through 4.5 are specific for the case of having all-to-all communication pattern. Figures 4.6 through 4.10 are for the case of using one-to-all communication pattern.

The allocation strategies used in these figures are:

(i) Random: the time complexity of this allocation strategy is O(S), where S is the size of the parallel job in hand to be allocated. This algorithm maintains a list of free processing nodes in the system. That's why it takes the same amount of time to allocated and deallocate processing nodes.

(ii) The Multiple Buddy Systems (MBS):

(iii) The First-Fit (FF): the time complexity of this algorithm is O(b), where b is the size of the list in which the set of free idle blocks (or sub-meshes) are maintained.

(iv) The Best-Fit (BF): asymptotically, the time complexity of this algorithm is also O(b), where b is the size of the list in which the set of free idle blocks (or sub-meshes) are maintained.

(v) The Bounded-Gradual Partitioning – Best Fit (BGP-BF): Basically, this allocation strategy uses the BF strategy to allocate the parallel job in hand. Thus, its time complexity is similar to the BF allocation strategy. However, if it is not possible to allocate the parallel job in a contiguous manner, this strategy subdivides the parallel job into two sub-requests as explained before. For a given request of size mxn, this allocation strategy will subdivide this

request into m+n subrequests in the worst case. Thus, the time complexity will be O((m+n)*b). Note that, if the partitioning bound of the BGP-BF used is k, where k is an integer, then the time complexity becomes O(k.b).

The first two strategies are non-contiguous strategies, while the last two are contiguous. Following is a list of abbreviations used in our discussion below:

- MJRT: Mean Job Response Time. Computed as

$$MJRT = \sum_{i=1}^{N} JRT(P_i) / N \quad \text{----------} \quad 4.1$$

Where $JRT(P_i)$ is the response time of the i^{th} job. N is the total number of jobs tested in the all 10 runs of the simulator.

- MJST: Mean Job Service Time, computed as

$$MJST = \sum_{i=1}^{N} JST(P_i) / N \quad \text{----------} \quad 4.2$$

Where $JST(P_i)$ is the service time of the i^{th} job. N is the total number of jobs tested in the all 10 runs of the simulator.

- MPBT: Mean Packet Blocking Time, computed as

$$MPBT = \sum_{i=1}^{M} PBT(p_i) / M \quad \text{----------} \quad 4.3$$

Where $PL(p_i)$ is the packet blocking time of the i^{th} packet sent over the links of the interconnection network (this differs from one packet to another depending on the distance from the source to the destination of this specific packet). M is the total number of messages sent during the all 10 runs of the simulator.

- MPL: Mean Packet Latency, computed as

$$MPL = \sum_{i=1}^{M} PL(p_i) / M \quad \text{----------4.4}$$

Where $PL(p_i)$ is the packet latency of the i^{th} packet sent over links of the interconnection network. M is the total number of messages sent during all 10 runs of the simulator.

- PSU: Percent System Utilization. Where:

$$PSU = \sum_{0}^{T} SU(t)/T \quad \text{----------4.5}$$

T: entire time of simulation SU(t): the ratio of utilization system at moment (t).

The following Figures show results of simulation.

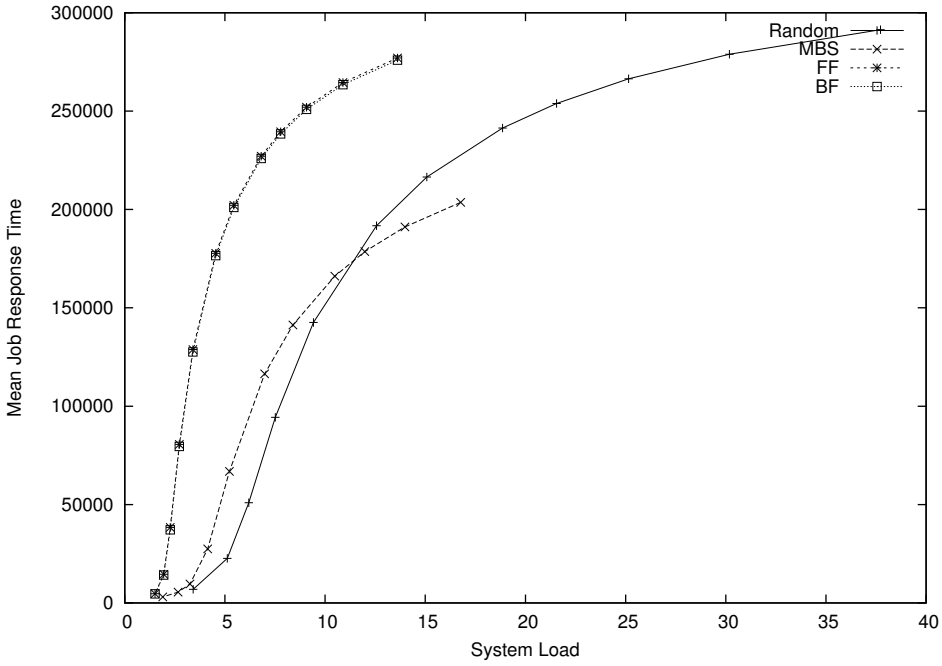


Fig. 4.1: Mean job response time (MJRT) vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Figure 4.1 represents relationship between the system load and mean job response time with the allocation strategies "MBS, FF, BF, Random" under the first come first serve scheduling mechanism and all-to-all communication pattern. The communication load parameter here is the mean number of messages sent by any scheduled job. The number of messages was fixed at 80, because after trying different numbers for messages, this number shows good balance between simulation time and communication load.

Observation 1 (figure 4.1): In general, contiguous allocation strategies show higher MJRT than the non-contiguous allocation strategies.

This is because adjacent allocation strategies require allocated processing units to be contiguous. The random allocation strategy showed high job response time compared to MBS at high system loads because that strategy randomly allocates the required processors.

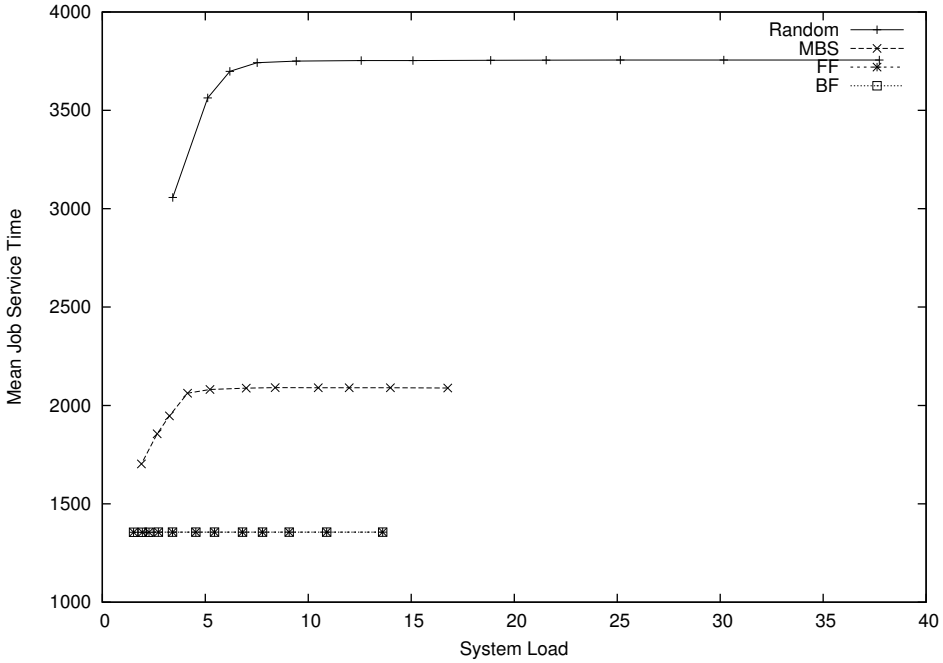


Fig. 4.2: Mean job service time vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Observation 2 (figure 4.2): Noncontiguous allocation strategies showed higher MJST than contiguous allocation strategies.

This is because non-contiguous allocation strategies disperse the set of processors allocated to the parallel job when contiguous allocation fails. This increases the amount of time required for intra-process communication.

Observation 3 (figure 4.2): The MJST is higher when applying the random allocation strategy compared to the case when applying the MBS allocation strategy.

The random allocation strategy usually causes parallel jobs to be more dispersed compared to the MBS strategy. The MBS strategy is partially contiguous; it tries to maintain some level of contiguity between allocated processors.

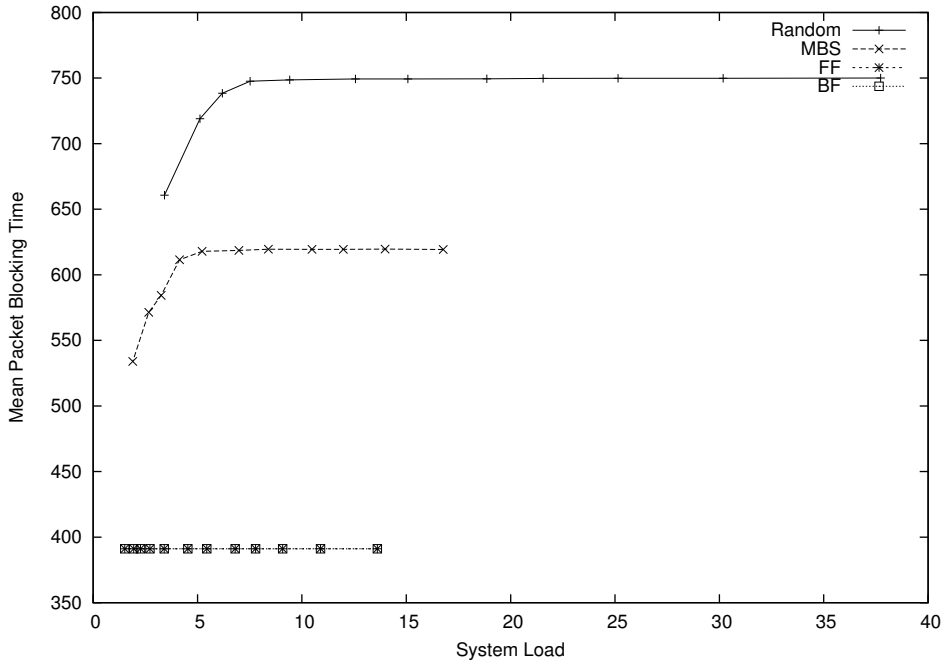


Fig. 4.3: Mean packet blocking time vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

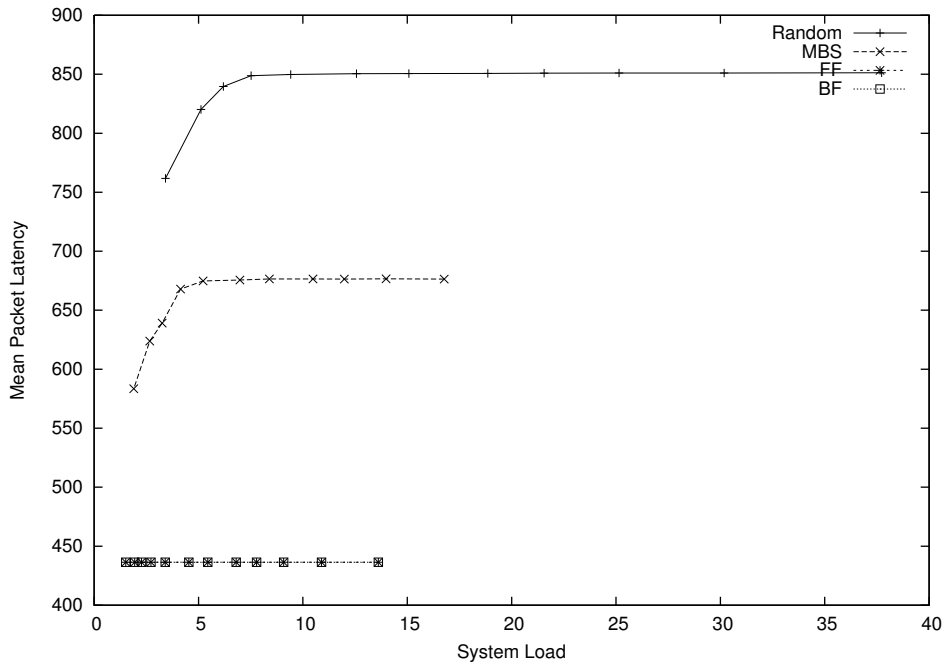


Fig. 4.4: Mean packet latency vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Observation 4 (figures 4.3 and 4.4): noncontiguous allocation strategies showed higher MPBT and MPL values than contiguous allocation strategies

We explain this observation the same way we explained observations 3 and 4: Non-contiguous allocation strategies cause more disperse to allocate parallel jobs producing longer distances between the sources and the destinations of messages. Since each message is blocked for sometime at each intermediate node from source to destination, the mean packet blocking time increases as the number of these intermediate nodes increases. The mean packet latency also increases accordingly.

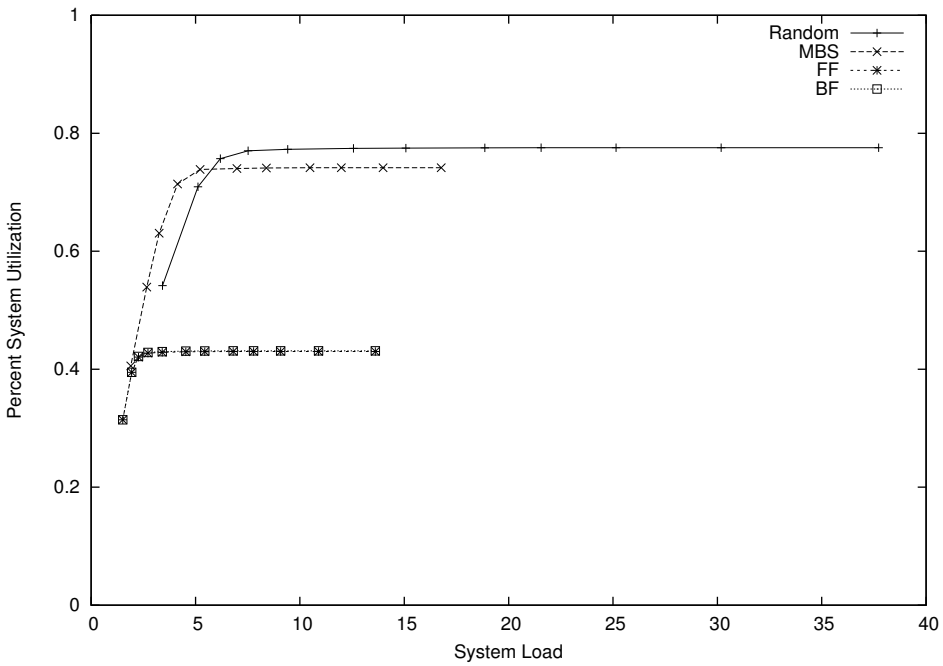


Fig. 4.5: Percent system utilization vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Observation 5 (figure 4.5): Contiguous allocation strategies showed low PSU values compared to noncontiguous allocation strategies

For example, at relatively high system loads, the random allocation strategy produces the highest percentage of system utilization of all experimented allocation strategies in figure 4.5. This is simply because random allocation strategy is allocating any idle processor from the list of free processors. The MBS strategy also allocates processors in a noncontiguous manner. Both strategies successfully allocate the parallel job in hand as long as enough number of free nodes is available. The FF and BF both force the parallel job to wait if no enough contiguous set of processors is available in the system. This, in turn, may produce high waiting time for relatively large parallel requests.

The table shown next summarizes our observations on figures 4.1 through 4.5.

Table 4.1: summarizes on figures 4.1 through 4.5.

(MJRT) vs. system load ,use all-to-all communication pattern "figure4.1"	MJST vs. system load , use all-to-all communication pattern "figure4.2"	MPBT vs. system load, use all-to-all communication pattern "figure4.3"	MPL vs. System load, use all-to-all communication pattern "figure4.4"	"PSU vs. System load, use all-to-all communication pattern "figure4.5"
Best-Fit, First-Fit	Random	Random	Random	Random
Random	MBS	MBS	MBS	MBS
MBS	Best-Fit, First-Fit	Best-Fit, First-Fit	Best-Fit, First-Fit	Best-Fit, First-Fit

Figures 4.6 through 4.10 are similar to the previous figures; however they are specific for the case of using one-to-all communication pattern.

Observation 6 (figures 4.6 to 4.10): Contiguous allocation strategies outperform Noncontiguous allocation strategies in terms of MJST, MPBT, and MPL.

Observation 7 (figures 4.6 to 4.10): Noncontiguous allocation strategies outperform contiguous allocation strategies in terms of MJRT.

Those observations have been explained before when we listed our observations on figures 4.1 to 4.5.

The table shown next summarizes our observations on figures 4.6 through 4.10.

Table 4.2: summarizes on figures 4.6 through 4.10.

(MJRT) vs. system load ,use one-to-all communication pattern "figure4.6"	Mean job service time vs. system load , use one-to-all communication pattern "figure4.7"	Mean packet blocking time vs. system load, use one-to-all communication pattern "figure4.8"	mean packet latency vs. System load, use one-to-all communication pattern "figure4.9"	percent system utilization vs. System load, use one-to-all communication pattern "figure4.10"
Best-Fit, First-Fit	Random	Random	Random	MBS
Random	MBS	MBS	MBS	Random
MBS	Best-Fit, First-Fit	Best-Fit, First-Fit	Best-Fit, First-Fit	Best-Fit, First-Fit

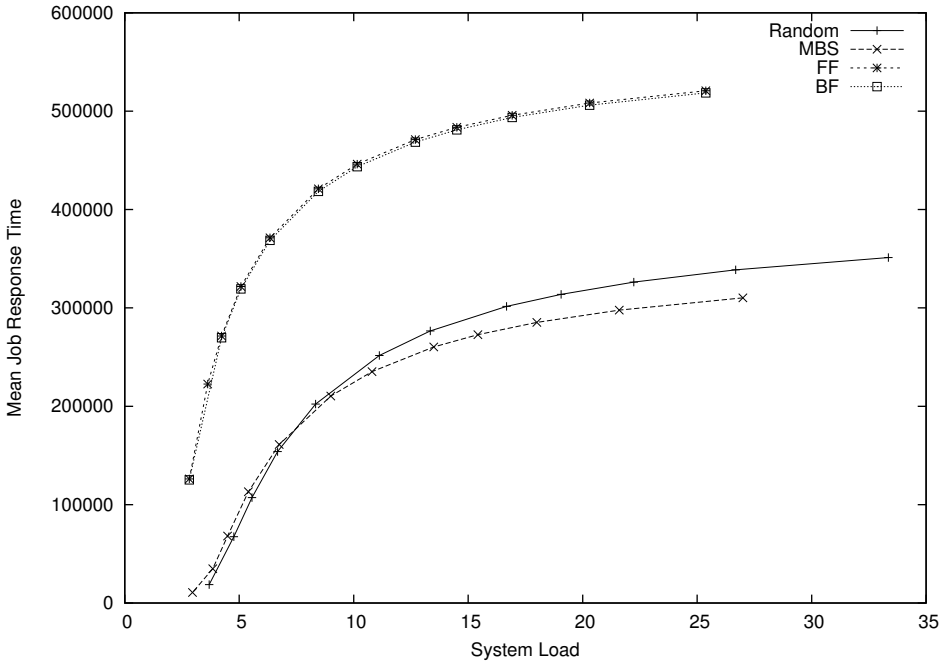


Fig. 4.6: Mean job response time vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

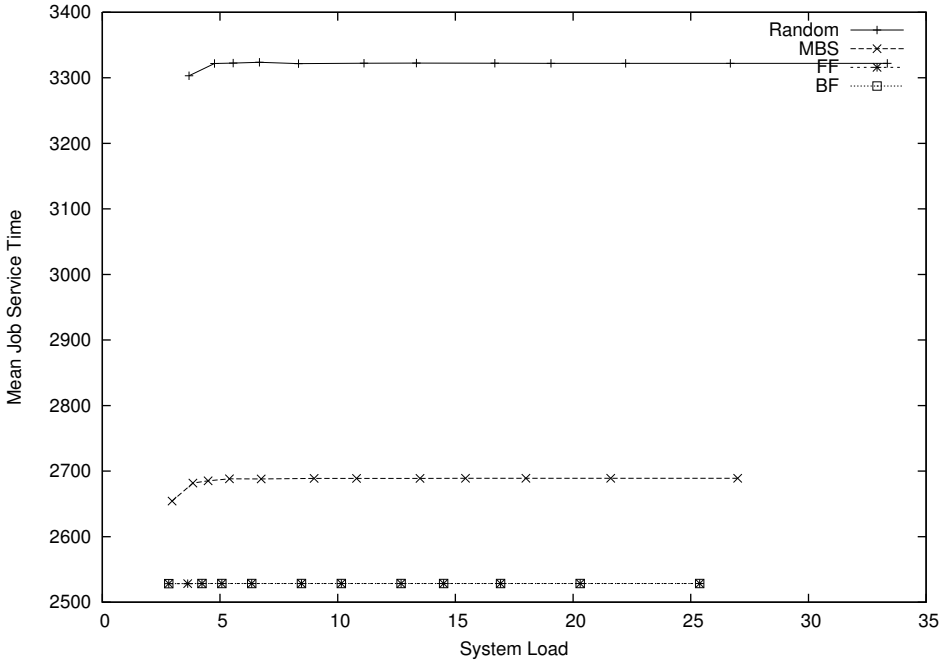


Fig. 4.7: Mean job service time vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

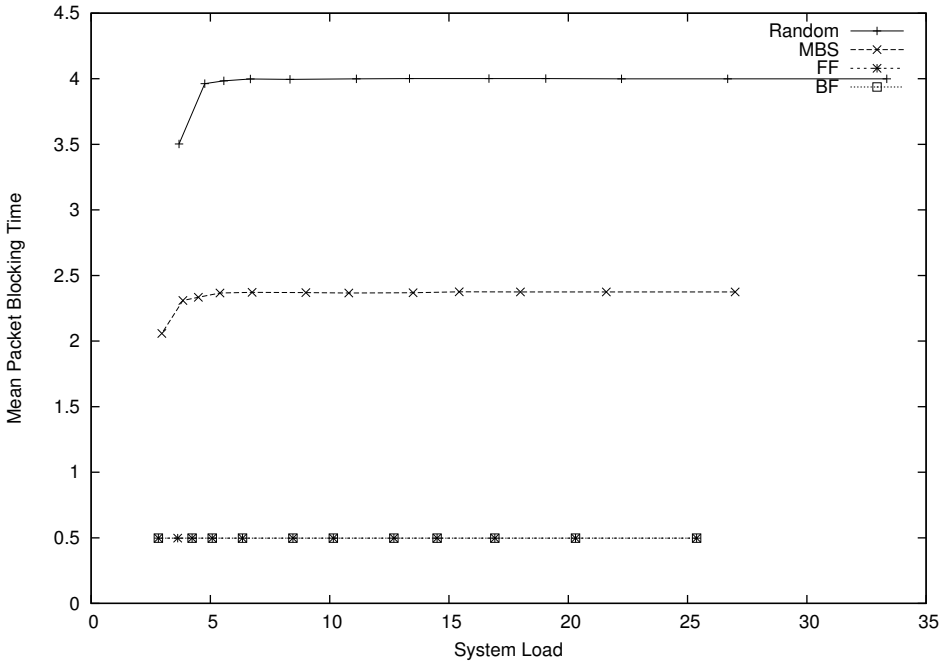


Fig. 4.8: Mean packet blocking time vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

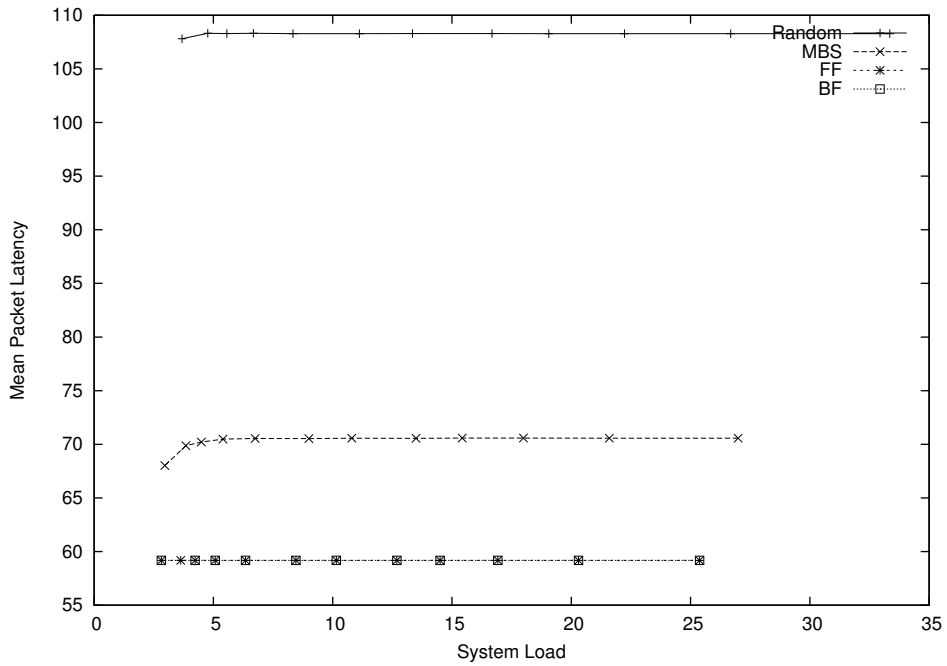


Fig. 4.9: Mean packet latency vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

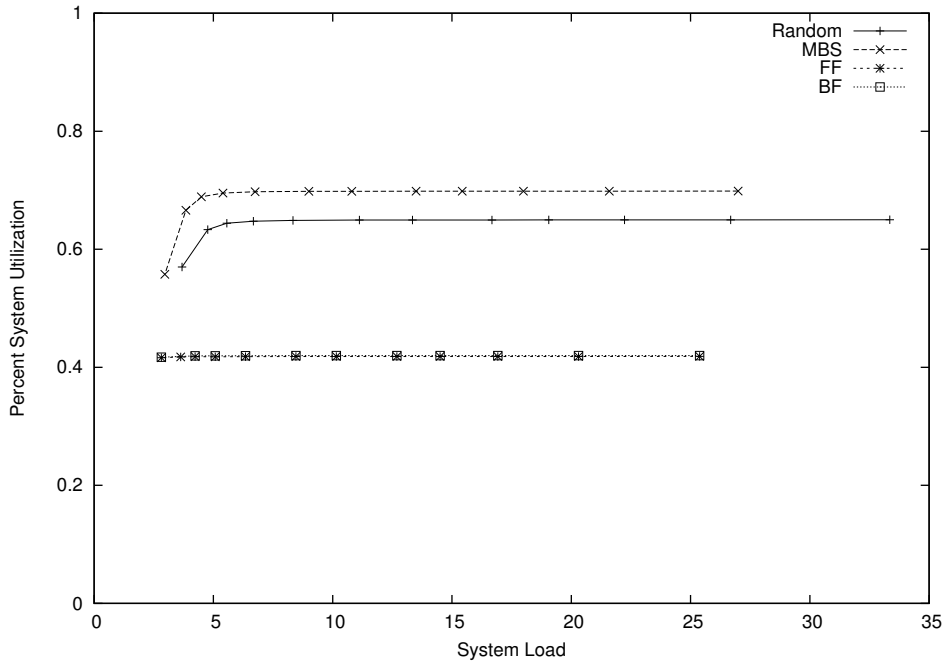


Fig. 4.10: Percent system utilization vs. system load in multiple allocation strategies under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

Observation 8 (figures 4.4 and 4.9): All-to-all communication pattern adds significantly higher communication load to the system compared to the one-to-all communication pattern.

This is because the total number of messages sent over the interconnection network produces more contention and significantly increases the MPL and MPBT.

This observation signifies the importance of our study, as it clearly illustrates the significant impact of communication load on the performance of the parallel system.

Figures 4.11 through 4.25 show relationships between system load and parameters: (i) the mean job response time, (ii) the mean job service time, (iii) the mean packet blocking time, (iv) the mean packet latency, and (v) the percent system utilization. This time, we study the impact of the communication patterns type; i.e.; all-to-all, one-to-all, random, Fast Fourier Transform (FFT), and the NAS multigrid benchmark communication patterns, and (DQBT) Divide and Conquer Binomial Tree on the key performance parameters of the multiprocessor system.

Figures 4.11 through 4.15 are specific for the contiguous allocation strategies. We use the BF allocation strategy as a representative for this group of allocation strategies.

Figures 4.16 through 4.20 are specific for the noncontiguous allocation strategies. We use the MBS allocation strategy as a representative for this group of allocation strategies.

Figures 4.21 through 4.25 are again specific for the noncontiguous allocation strategies. However, we use the BGP-BF allocation strategy as a representative for this group of allocation strategies. In these experiments, the applied partitioning bound is equal to 4.

Next, we present our observations on these figures.

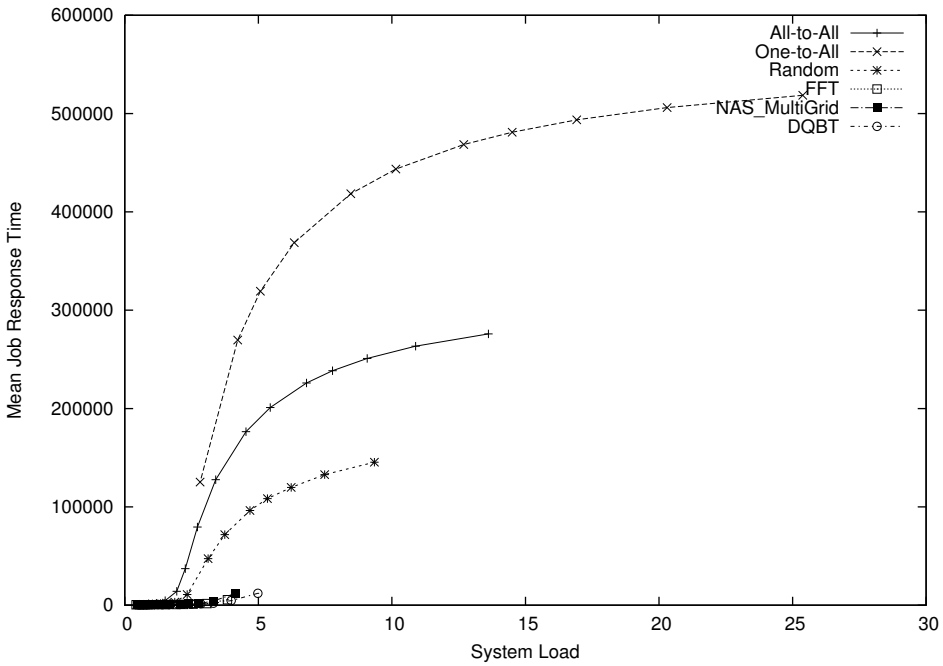


Fig. 4.11: Mean job response time vs. system load in Best Fit allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

Observation 9 (figures 4.11 and 4.12): One-to-all, All-to-all and random communication patterns produces the highest MJRT and MJST of all tested communication patters.

Next we list the tested communication patterns starting with the one caused the highest MJRT: one-to-all, all-to-all, random, NAS-multiglid, FFT, and DQBT. This can be explained as follows, one-to-all and all-to-all communication patterns generate large number of messages to be transmitted over the interconnection network. This increases the service time of allocated parallel jobs and, thus, forces unallocated parallel jobs to wait more in the ready queue of the system. This, in turn, increases the mean job response time.

Figures 4.13 and 4.14 show the effect of communication pattern on the MPBT and MPL values of the system in the case of applying the BF allocation strategy. We notice that system behavior is quiet strange and that the figure is not directly

conclusive. This is because that the number of jobs served during the simulator is relatively low due to the following factors:

- The allocation strategy applied in contiguous. This reduces the number of allocated jobs due to the condition of contiguity, because the Best Fit allocation strategy tries to allocate a sub-mesh, i.e., a contiguous set of processing units, of the same size and shape of parallel requests.
- FCFS is the applied scheduling mechanism. This may prevent many parallel jobs from being allocated if a relatively big parallel job is residing at the head of the ready queue of the system.

Based on the above two factors, we believe that the system behavior is not clear in figures 4.13, 4.14 and 4.15. To remedy this problem and to better capture the impact of communication pattern type, we need to consider applying a non-contiguous allocation strategy (which we do next with the MBS and the BGP-BF allocation strategies).

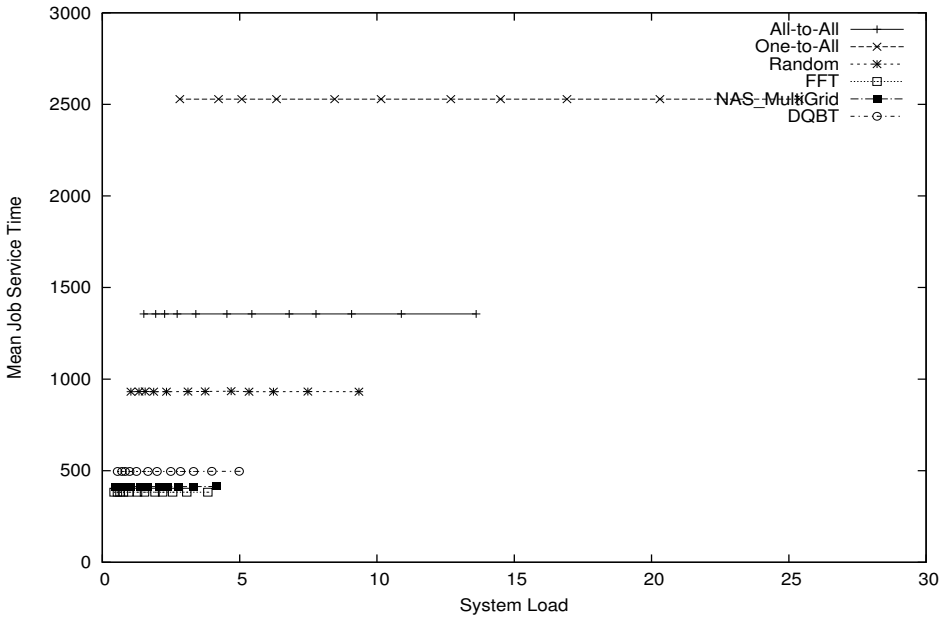


Fig. 4.12: Mean job service time vs. system load in Best Fit allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

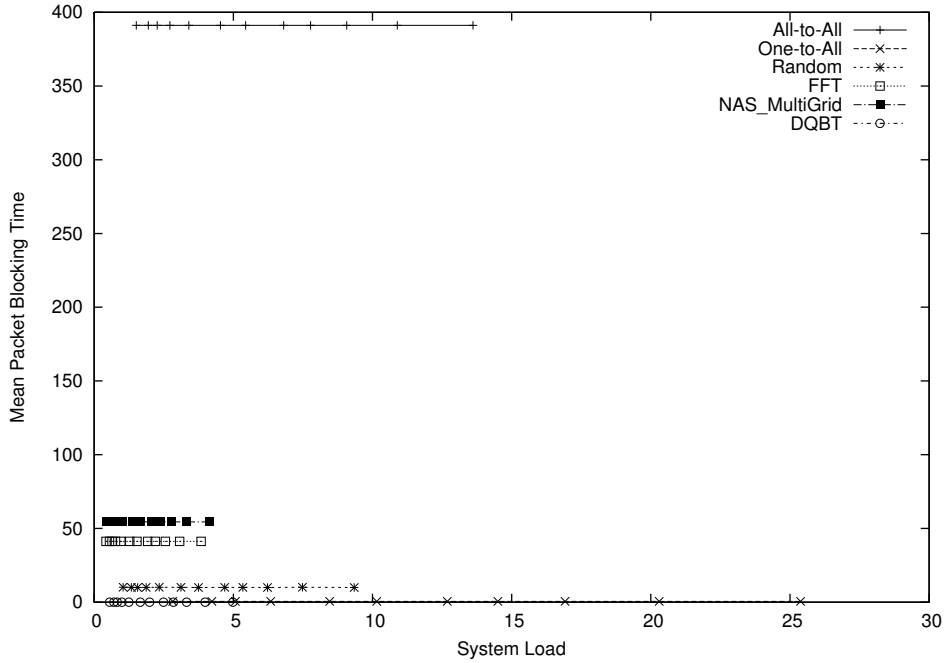


Fig. 4.13: Mean packet blocking time vs. system load in Best Fit allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

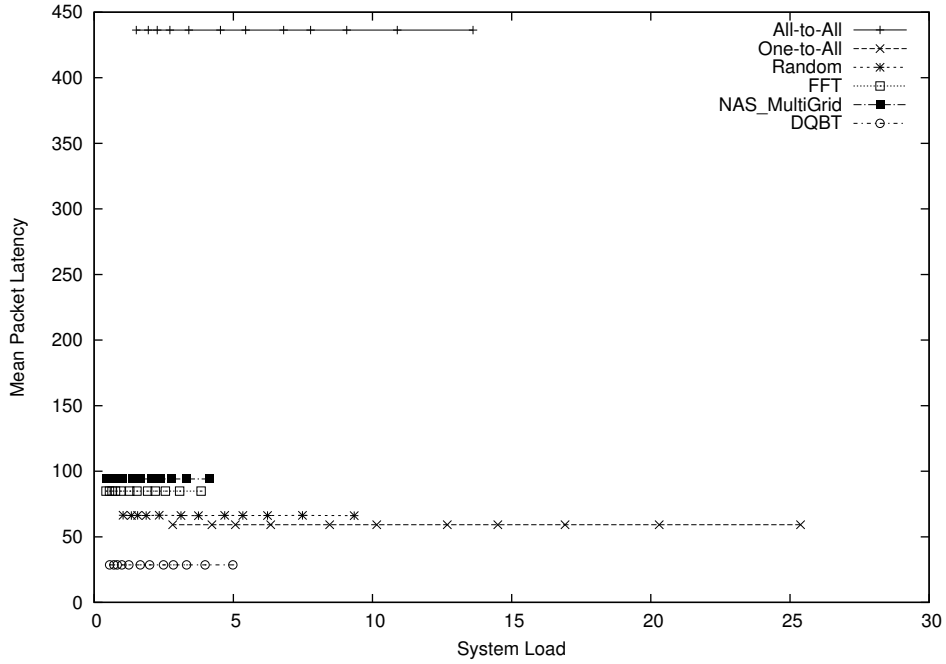


Fig. 4.14: Mean packet latency vs. system load in Best Fit allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

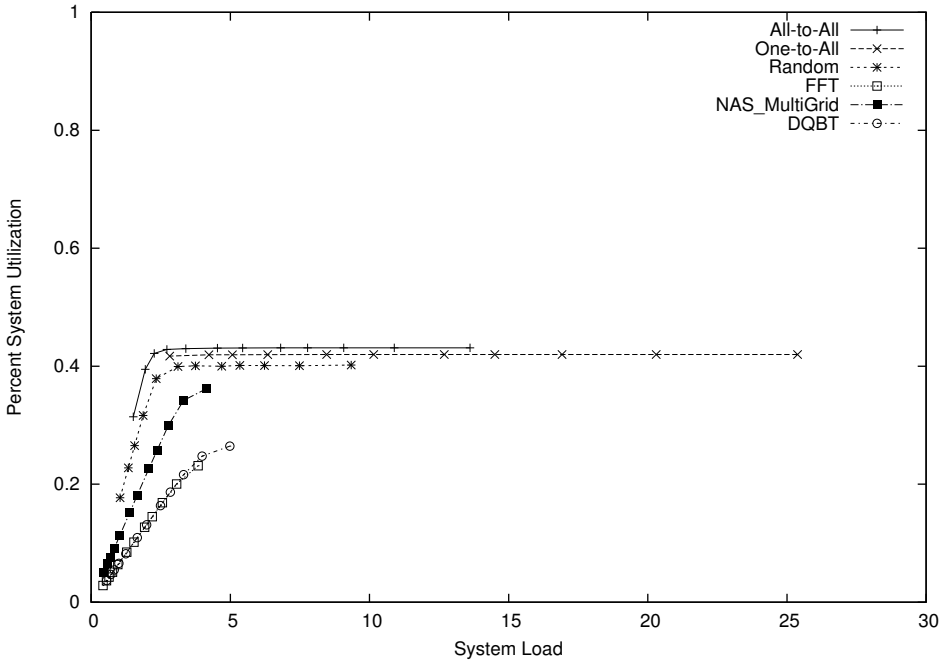


Fig. 4.15: Percent system utilization vs. system load in Best Fit allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

The table shown next summarizes our observations on figures 4.11 through 4.15.

Table 4.3: summarizes on figures 4.11 through 4.15.

Best Fit allocation strategy "mean job response time vs. System load"figure4.11"	Best Fit allocation strategy "mean job service time vs. System load"figure4.12"	Best Fit allocation strategy "mean packet blocking time vs. System load"figure4.13"	Best Fit allocation strategy "mean packet latency vs. System load"figure4.14"	Best Fit allocation strategy "percent system utilization vs. System load"figure4.15"
One-to-All	One-to-All	All-to-All	All-to-All	All-to-All
All-to-All	All-to-All	NAS	NAS	One-to-All
Random	Random	FFT	FFT	random
NAS	DQBT	Random	Random	NAS
FFT	NAS	One-to-All	One-to-All	DQBT
DQBT	FFT	DQBT	DQBT	FFT

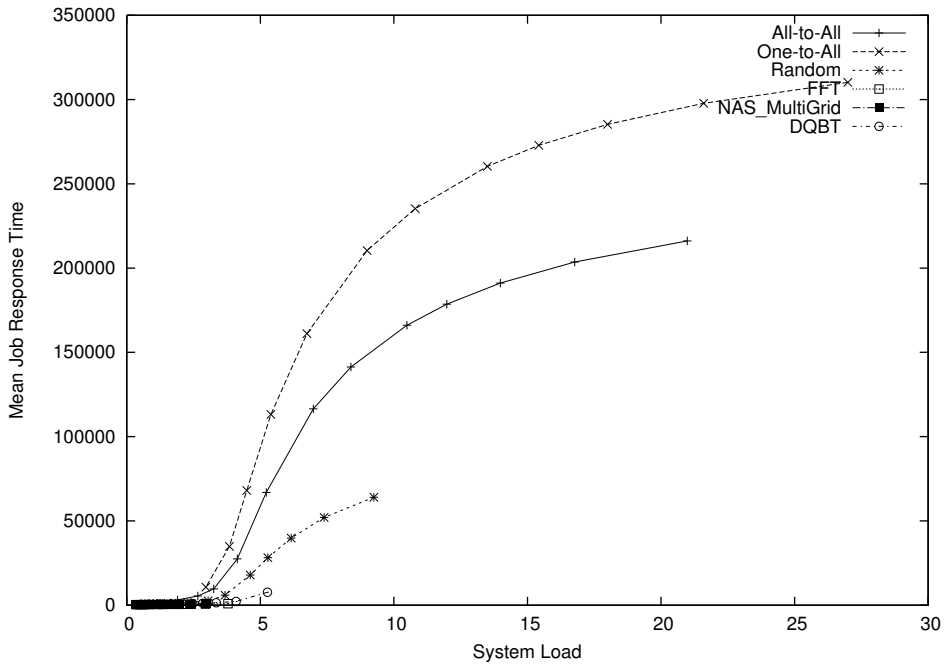


Fig. 4.16: Mean job response time vs. system load in MBS allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

Figures 4.16 through 4.20 are specific for the noncontiguous allocation strategies. We use the MBS allocation strategy as a representative for this group of allocation strategies.

Figures 4.16 and 4.17 showed results similar to figures 4.11 and 4.12 that have been explained before (with the BF allocation strategy). One difference can be observed.

Observation 10 (figures 4.11, 4.12 4.16 and 4.17): the maximum MJRT and MJST values observed with MBS were 500000 and 2500 for the BF algorithm, respectively. as to the MBS algorithm, these values were 300000 and 2700, respectively.

This is expected because the wait time with BF is higher than wait time with MBS. And because of non-contiguity, the service time is higher with MBS compared to the BF algorithm.

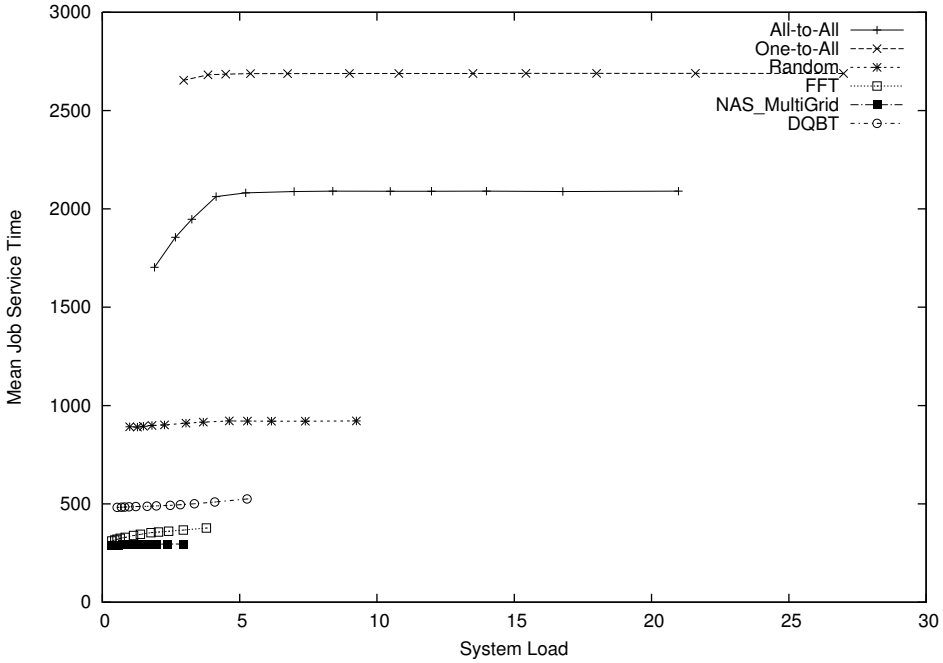


Fig. 4.17: Mean job service time vs. system load in MBS allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

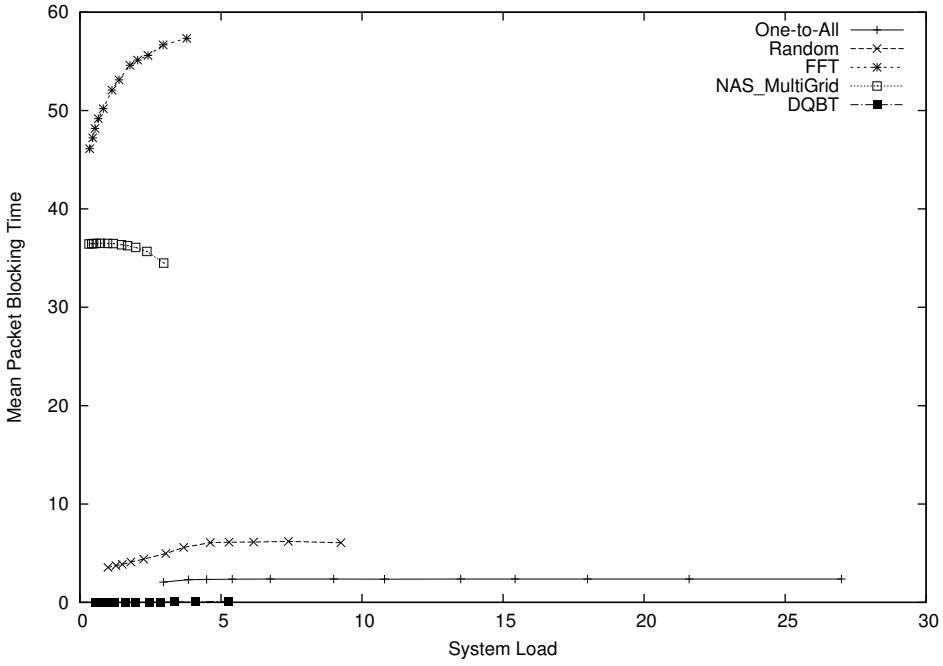


Fig. 4.18: Mean packet blocking time vs. system load in MBS allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

Figures 4.18, 4.19 and 4.20 plot the MPBT, the MPL and the PSU vs system load when using the MBS allocation strategy and multiple communication patterns.

Observation 11 (figures 4.13, 4.14, 4.15, 4.18, 4.19, and 4.20): the behavior of the parallel system can be significantly affected by the communication pattern of the parallel jobs being allocated.

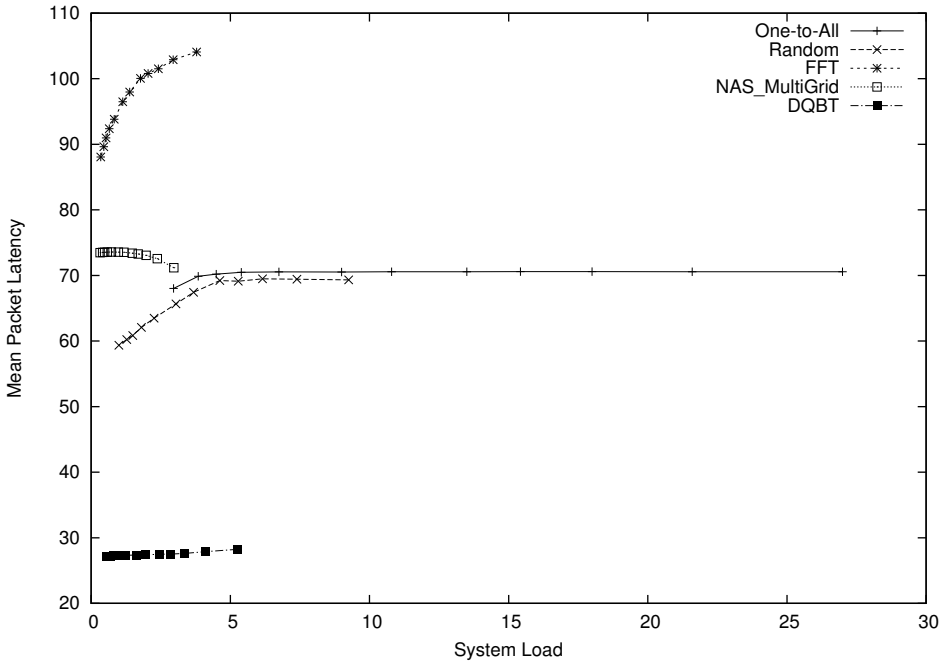


Fig. 4.19: Mean packet latency vs. system load in MBS allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

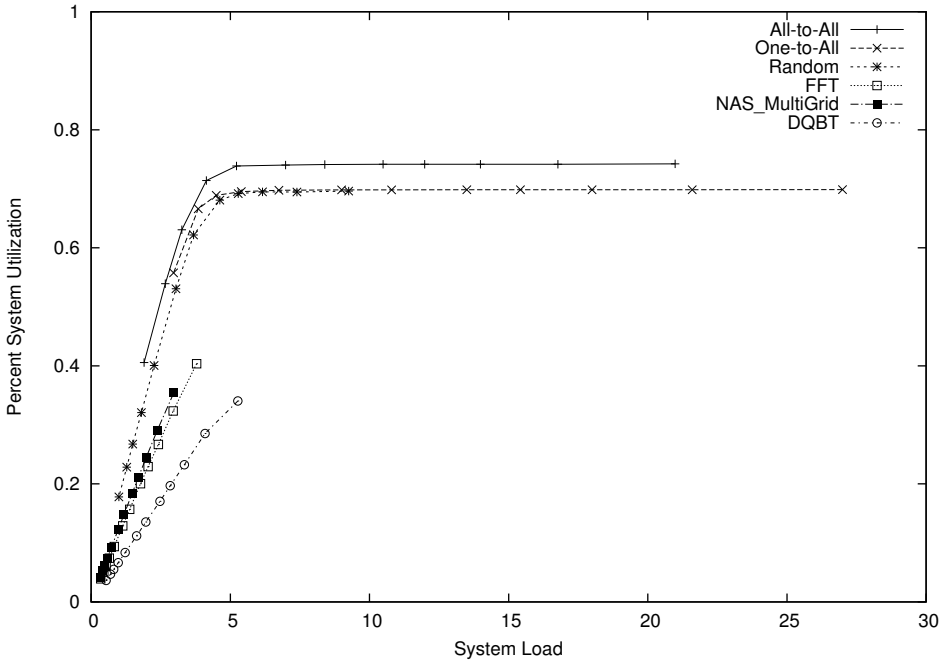


Fig. 4.20: Percent system utilization vs. system load in MBS allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

In figures 4.21 to 4.25, we present the relationships between the system load and more parameters, (i) the mean job response time, (ii) the mean job service time, (iii) the mean packet blocking time, (iv) the mean packet latency, and (v) the percent system utilization., that focus on the BGP-BF allocation strategy.

The table shown next summarizes our observations on figures 4.16 through 4.20.

Table 4.4: summarizes on figures 4.16 through 4.20.

MBS allocation strategy "mean job response time vs. System load"figure4.16"	MBS allocation strategy "mean job service time vs. System load"figure4.17"	MBS allocation strategy "mean packet blocking time vs. System load"figure4.18"	MBS allocation strategy "mean packet latency vs. System load"figure4.19"	MBS allocation strategy "percent system utilization vs. System load"figure4.20"
One-to-All	One-to-All	FFT	FFT	All-to-All
All-to-All	All-to-All	NAS	NAS	One-to-All
Random	Random	Random	One-to-All	Random
DQBT	DQBT	One-to-All	Random	FFT
FFT	FFT	DQBT	DQBT	NAS
NAS	NAS			DQBT

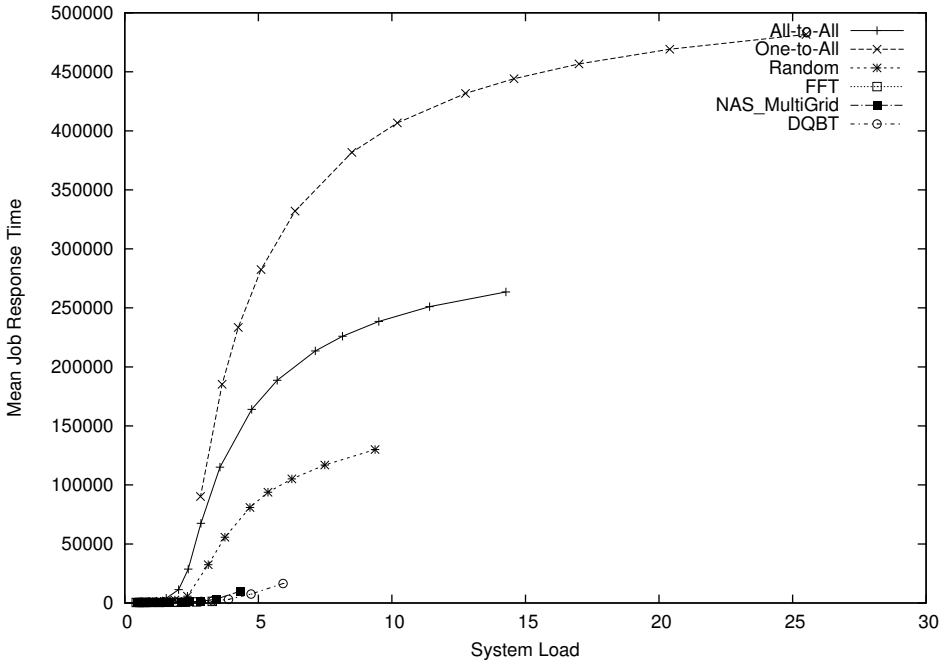
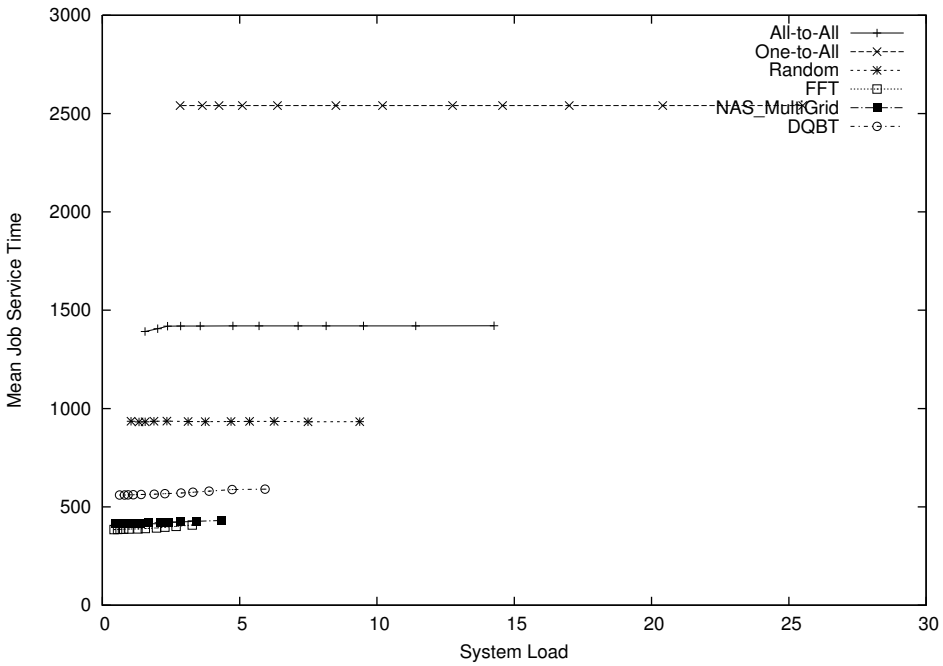


Fig. 4.21: Mean job response time vs. system load in the BGP-BF (partitioning bound = 4) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).



ig. 4.22: Mean job service time vs. system load in the BGP-BF (partitioning bound = 4) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

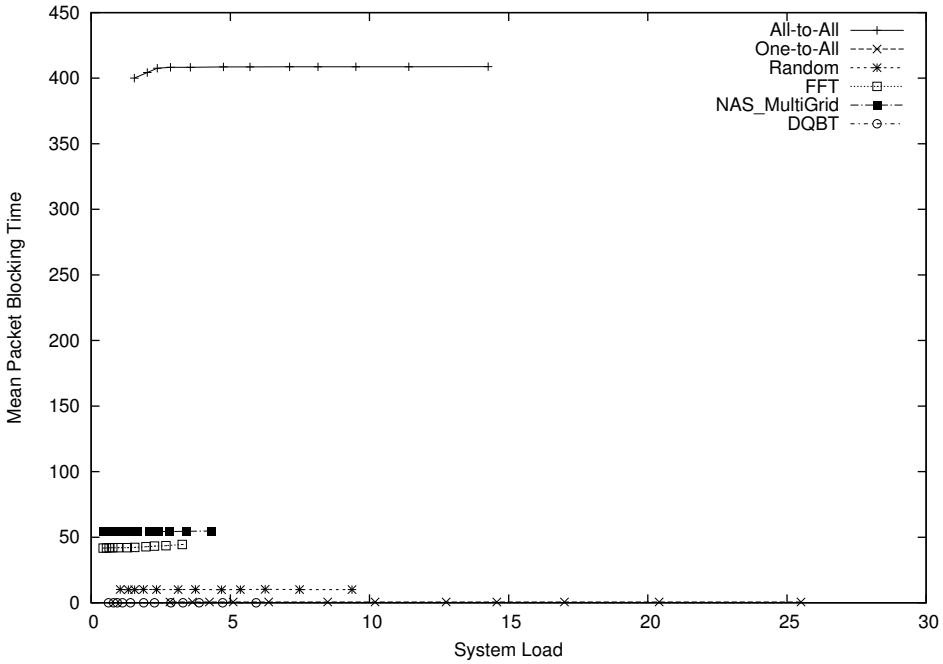


Fig. 4.23: Mean packet blocking time vs. system load in the BGP-BF (partitioning bound = 4) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

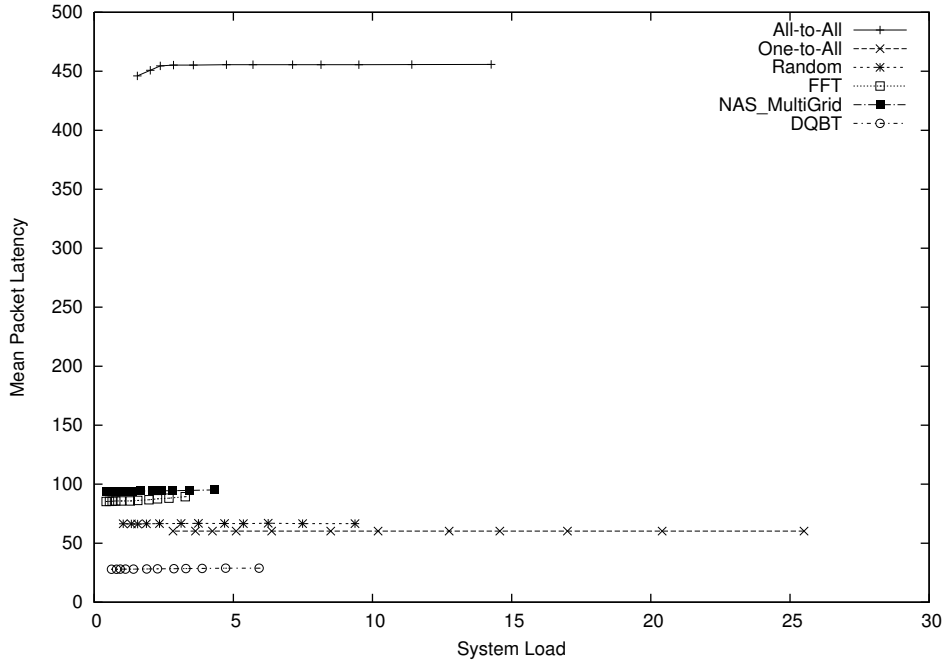


Fig. 4.24: Mean packet latency vs. system load in the BGP-BF (partitioning bound = 4) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

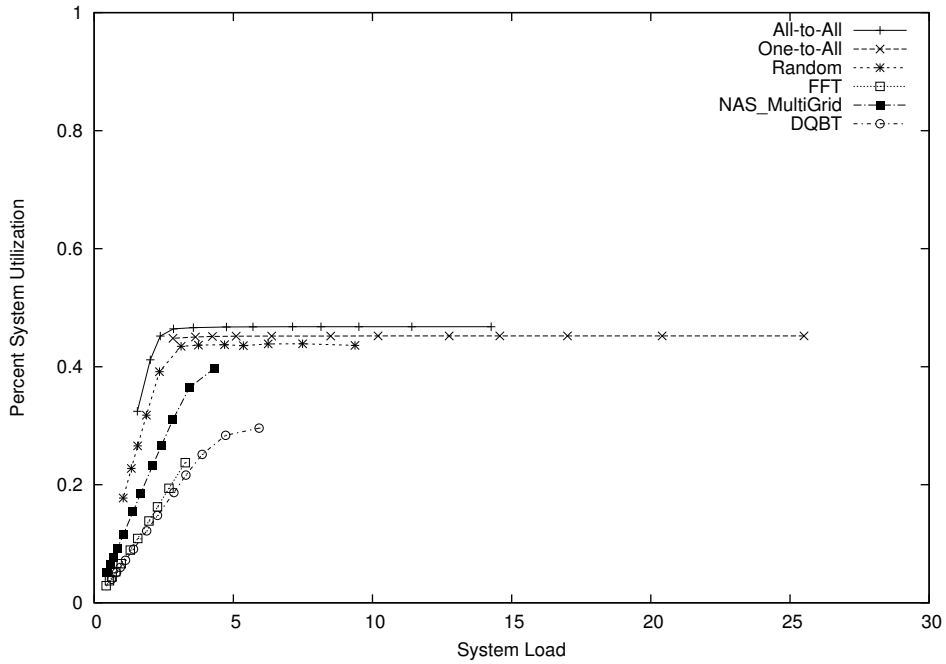


Fig. 4.25: Percent system utilization vs. system load in the BGP-BF (partitioning bound = 4) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

Observation 12 (figures 4.15, 4.20 and 4.25): the behavior of the parallel system can be significantly affected by the communication pattern of the allocated parallel jobs .

For instance, the random and the one-to-all communication patterns produced similar PSU value when applying the MBS allocation strategy. However, the one-to-all allocation strategy outperformed the random communication pattern when using the BF and the BGP-BF strategy.

Next we study the impact of the partitioning bound of the BGP-BF allocation strategy on the different system performance parameters. This will be in figures 4.26 through 4.35. Figures 4.26 to 4.30 are specific for the all-to-all communication pattern. While figures 4.31 to 4.35 are specific for the communication pattern one-to-all.

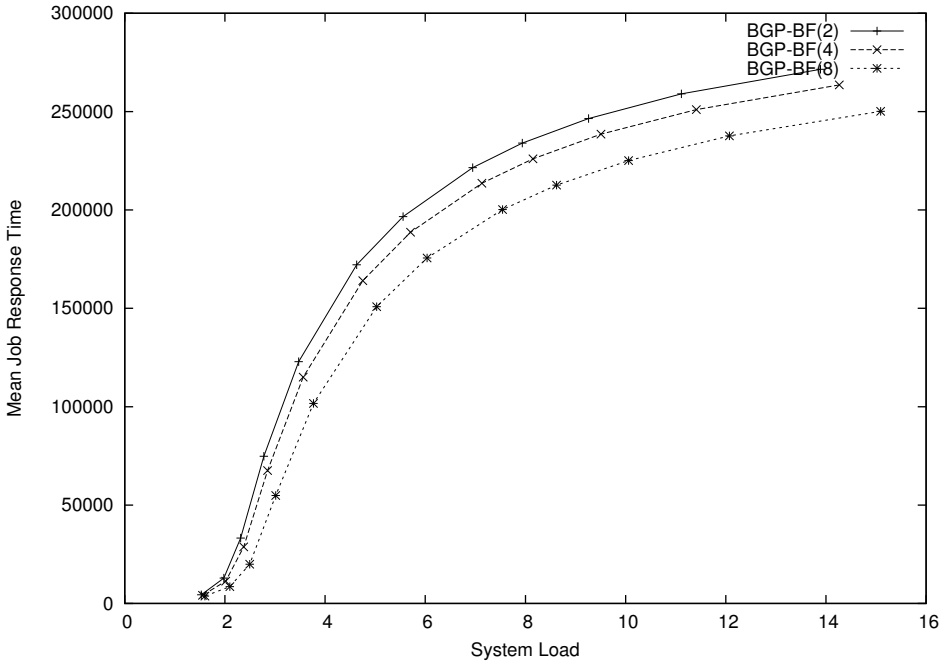


Fig. 4.26: Mean job response time vs. system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Observation 13 (Figure 4.26): In general, when the bound is high, the mean job response time is less.

For example, the BGP-BF (2) produced the highest MJRT value and followed by BGP-BF (4) and BGP-BF (8).

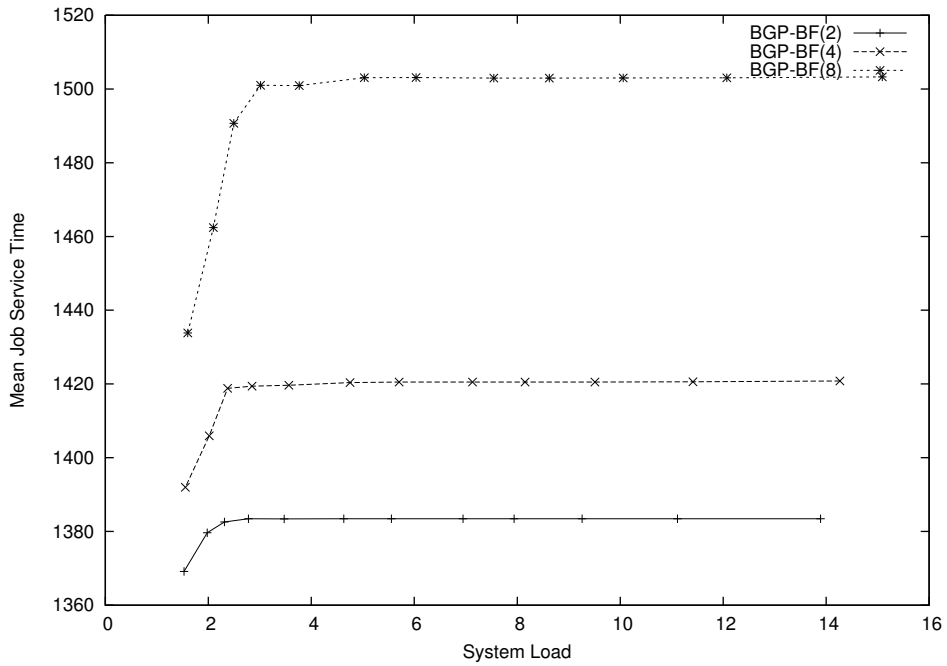


Fig. 4.27: Mean job response time vs. system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Figure 4.27 represents the mean job service time and system load in BGP-BF (partitioning bounds= 2, 4 and 8) allocation strategy.

Observation 14 (figure 4.27): When we increase the partitioning bounded, the mean job service time also increases.

This is because when we increase the partitioning bound, the execution time increases because the level on non-contiguity increases. This results in an increase in the service time of the job.

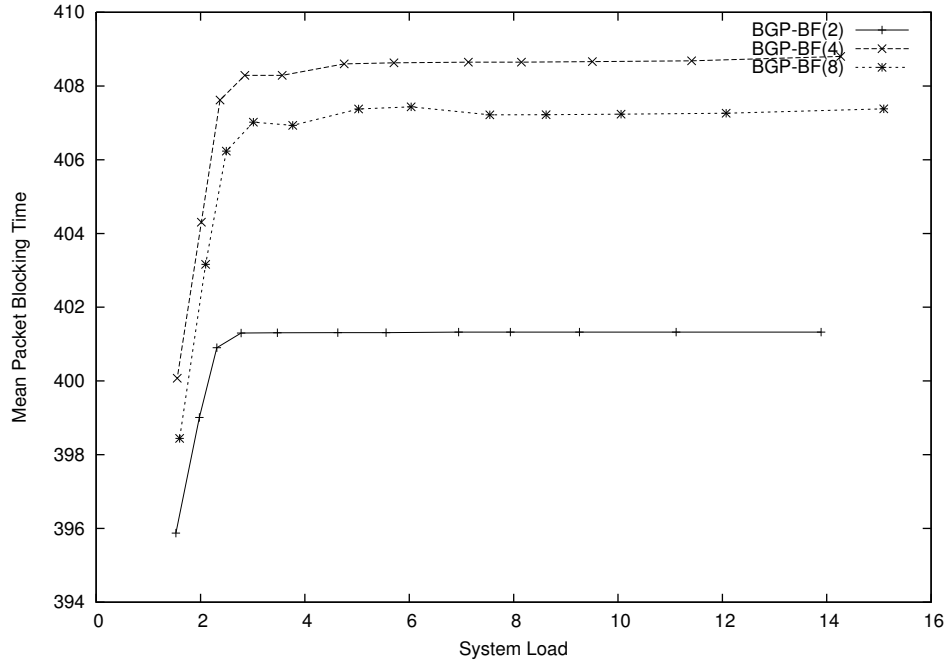


Fig. 4.28: Mean packet blocking time vs. system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Figures 4.28 4.29 plot the relation between the mean packet blocking time and the mean packet latency (y-axis) and the system load (x-axis) when applying the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern.

Observation 15 (figures 4.28 and 4.29): In general, when the partitioning bound is high, the MPBT and the MPL is high.

This is because when the bound is high, the time that message packets spend blocked in network buffers waiting for access to their next channel is high.

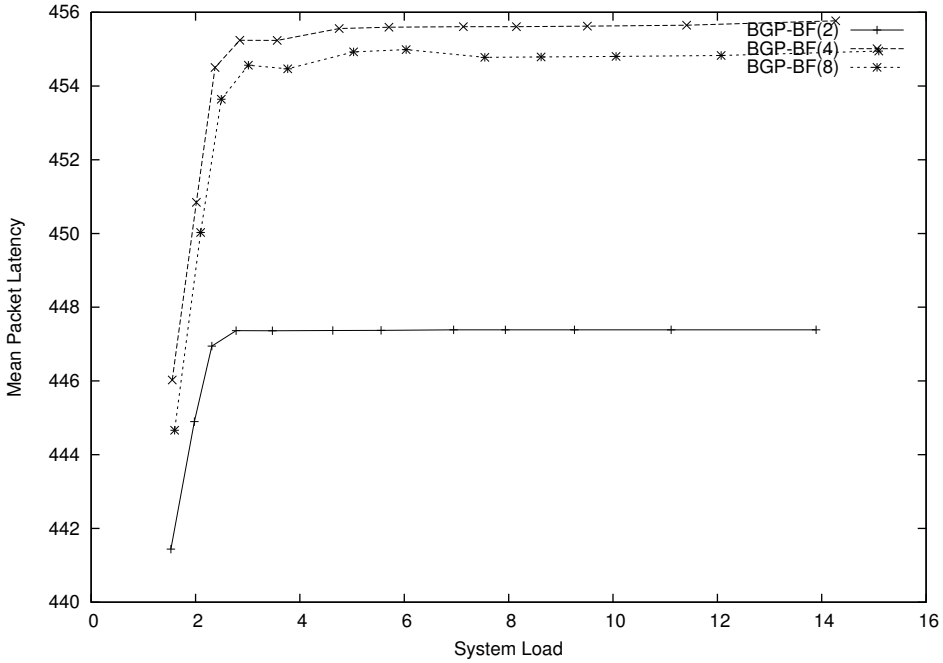


Fig. 4.29: Mean packet latency vs. system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Figure 4.29 represents the relation between mean packet latency and system load in the BGP-BF "partitioning bounds=2, 4 and 8" allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern.

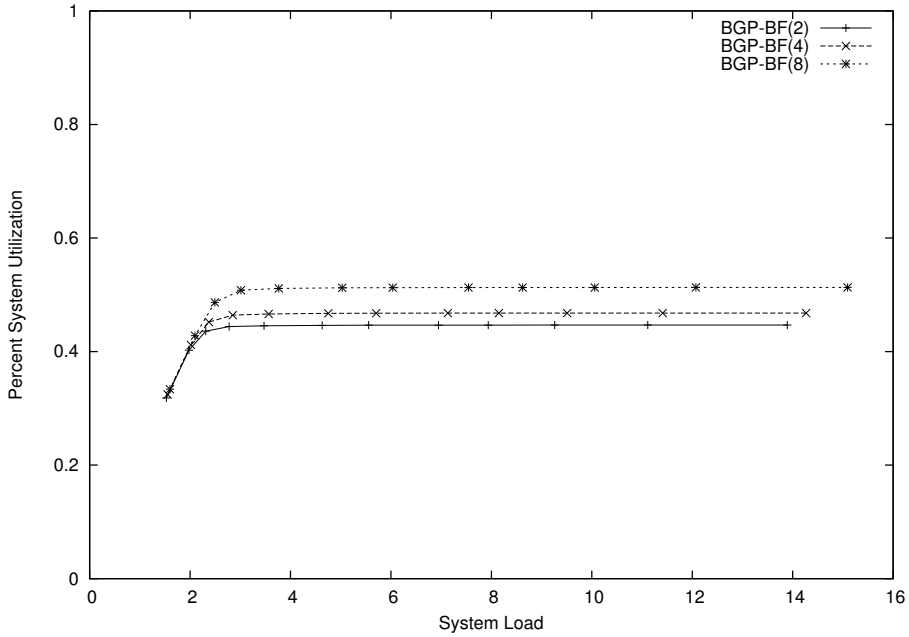


Fig. 4.30: Percent system utilization vs. system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern (message count = 80).

Figure 4.30 plots the percentage of system utilization and system load in the BGP-BF (partitioning bounds = 2, 4, and 8) allocation strategy under the FCFS scheduling mechanism and all-to-all communication pattern.

Observation 16 (figure 4.30): when the partitioning bound increase, the percentage system utilization also increases.

BGP-BF (8) showed the highest PSU values, followed by the BGP-BF (4), BGP-BF (2) respectively. This is because when the bound is higher the processors of the multicomputer system are more efficiently utilized.

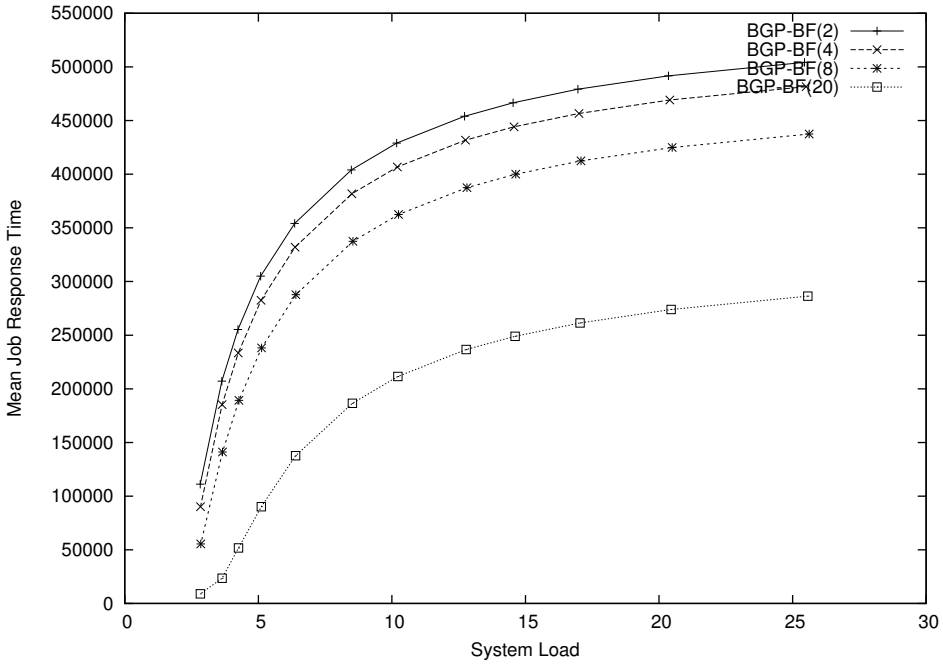


Fig. 4.31: Mean job response time vs. system load in the BGP-BF (partitioning bounds = 2, 4, 8, and 20) allocation strategy under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

Figures 4.31 through 4.35 (one-to-all communication pattern) give similar observations to figures 4.26 through 4.30 (all-to-all communication pattern).

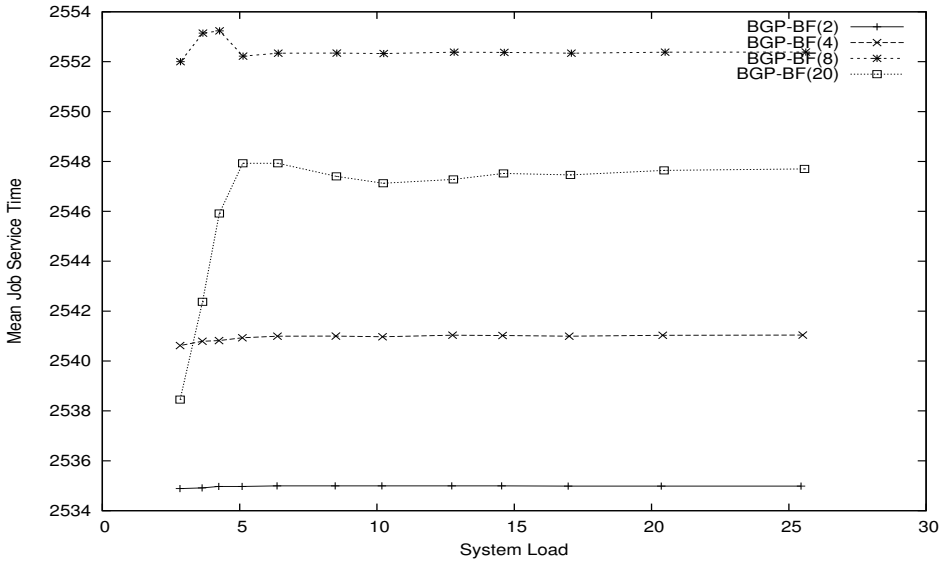


Fig. 4.32: Mean job service time vs. system load in the BGP-BF (partitioning bounds = 2, 4, 8, and 20) allocation strategy under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

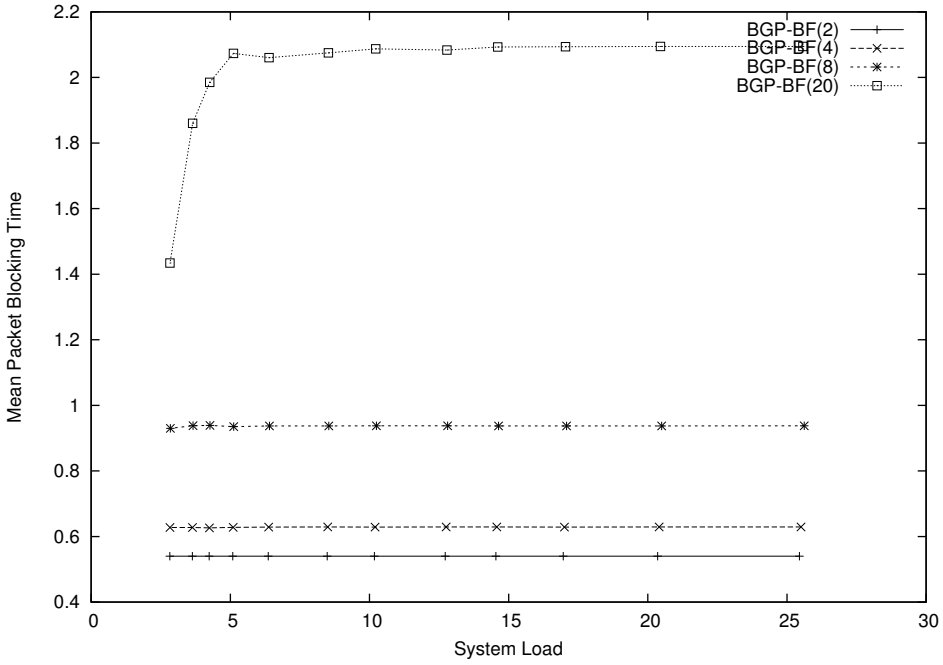


Fig. 4.33: Mean packet blocking time vs. system load in the BGP-BF (partitioning bounds = 2, 4, 8, and 20) allocation strategy under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

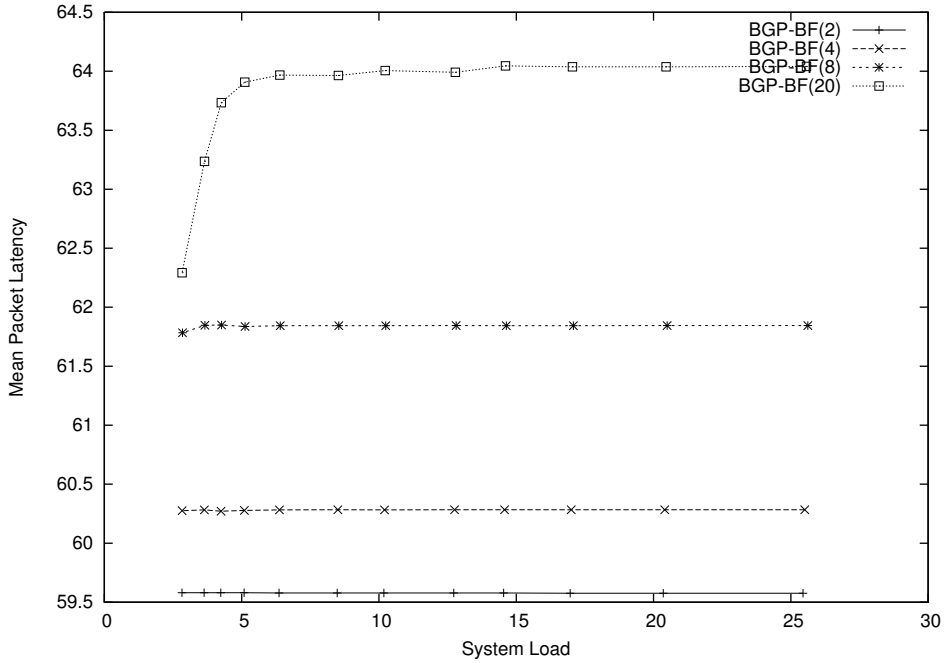


Fig. 4.34: Mean packet latency vs. system load in the BGP-BF (partitioning bounds = 2, 4, 8, and 20) allocation strategy under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

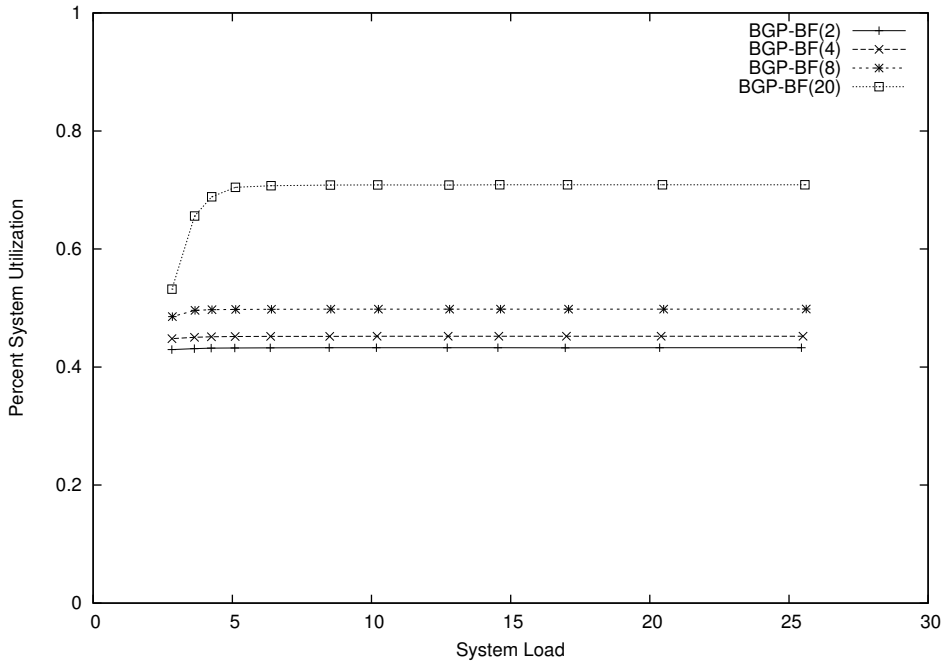


Fig. 4.35: Percent system utilization vs. system load in the BGP-BF (partitioning bounds = 2, 4, 8, and 20) allocation strategy under the FCFS scheduling mechanism and one-to-all communication pattern (message count = 80).

Figures 4.36 through 4.40 represent the relation between the system load and the following system performance parameters: the mean job response, the mean job service time, the mean packet blocking time, mean packet latency and the percentage

of system utilization. Here, we focus on the BGP-BF (partitioning bound=8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns such as all-to-all, one-to-all, random, NAS-multigrid, DQBT.

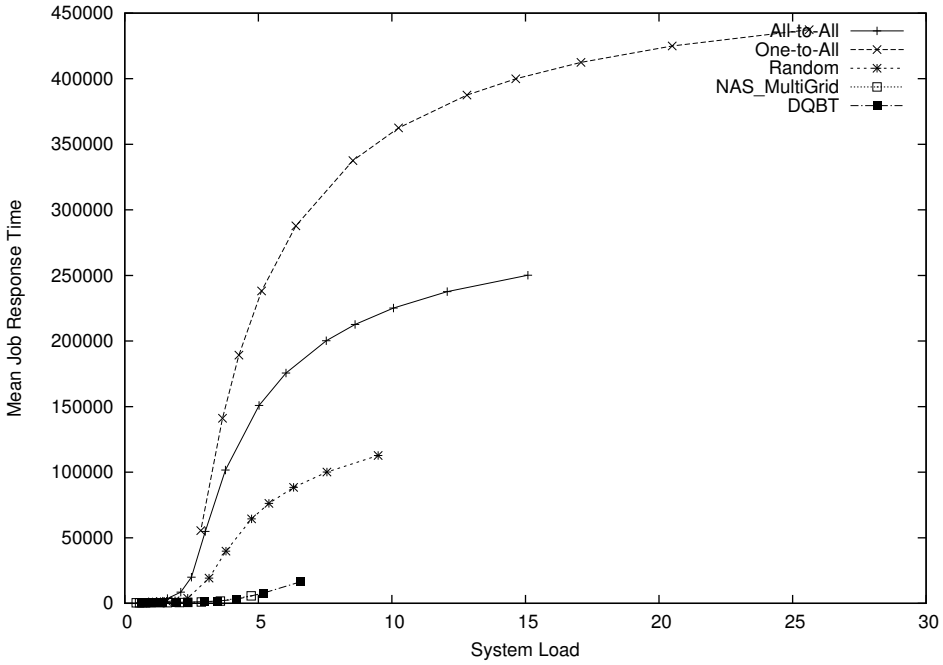


Fig. 4.36: Mean job response time vs. system load in the BGP-BF (partitioning bound = 8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

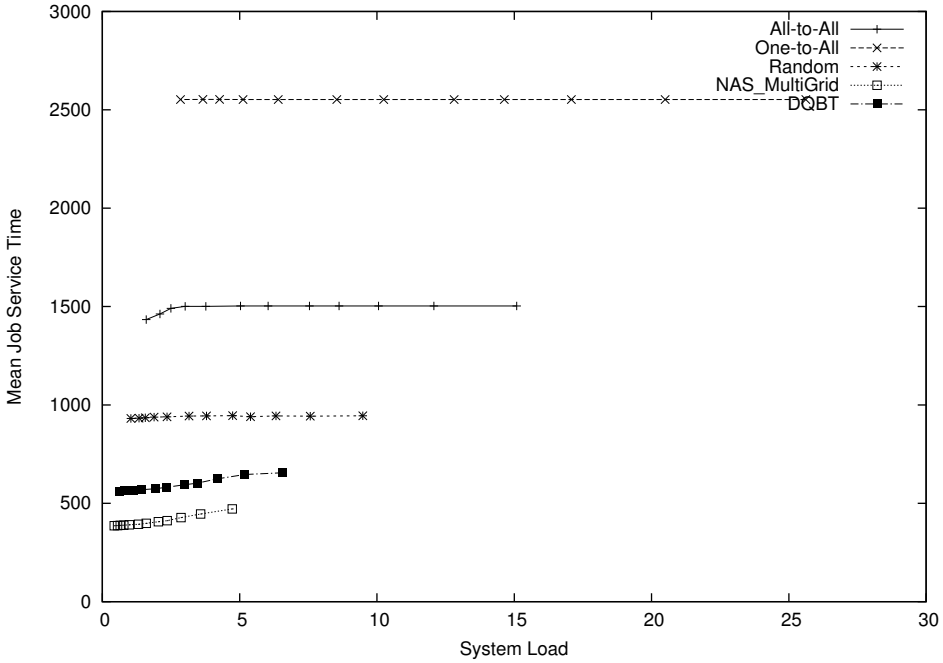


Fig. 4.37: Mean job service time vs. system load in the BGP-BF (partitioning bound = 8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

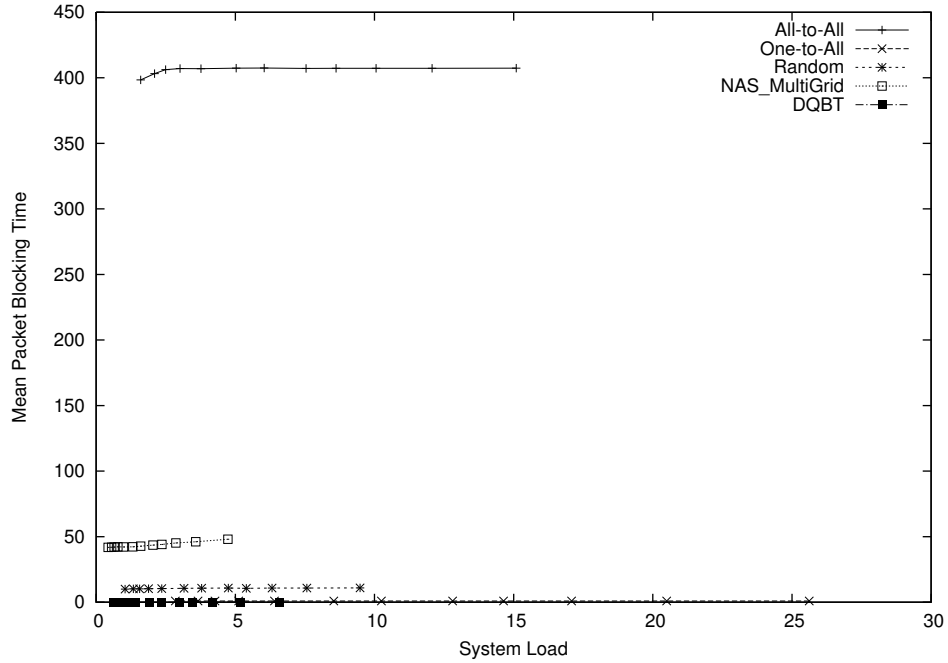


Fig. 4.38: Mean packet blocking time vs. system load in the BGP-BF (partitioning bound = 8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

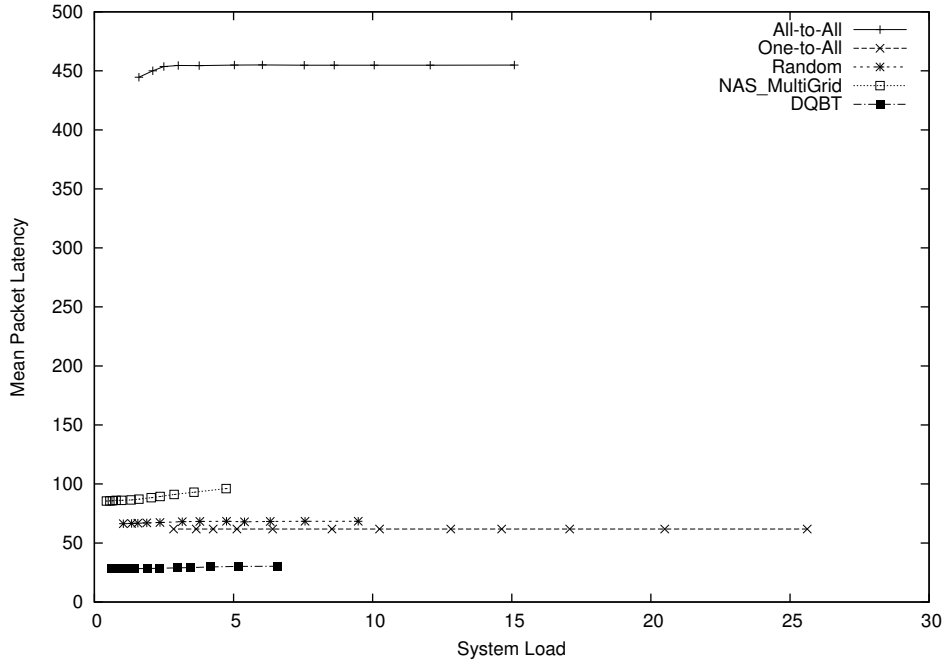


Fig. 4.39: Mean packet latency vs. system load in the BGP-BF (partitioning bound = 8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

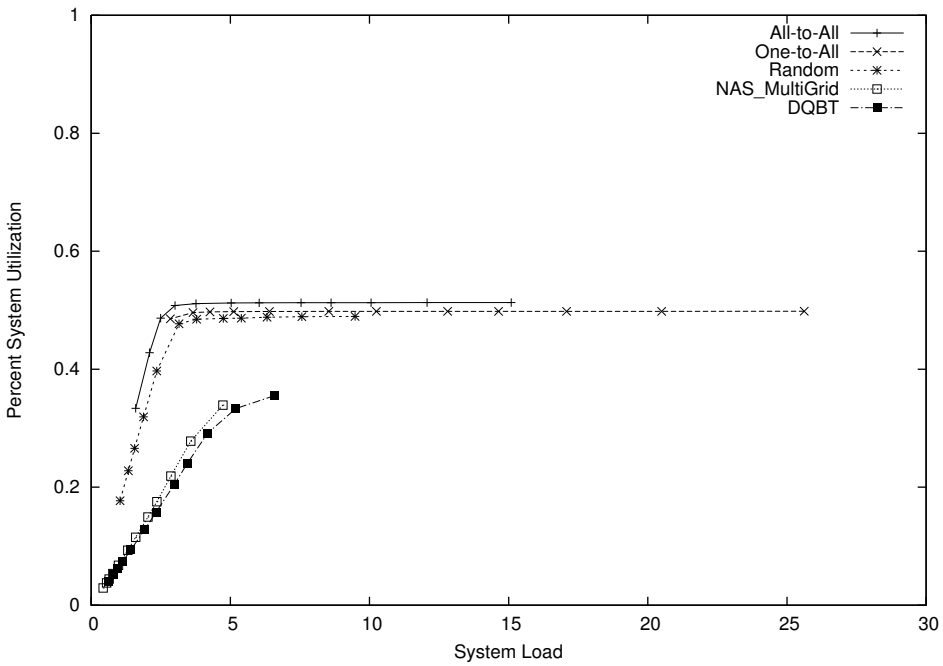


Fig. 4.40: Percent system utilization vs. system load in the BGP-BF (partitioning bound = 8) allocation strategy under the FCFS scheduling mechanism and multiple communication patterns (message count = 80).

The table shown next summarizes our observations on figures 4.36 through 4.40.

Table 4.5: summery on figures 4.36 through 4.40.

BGP-BF(8)"mean job response time vs. System load"figure4.36"	BGP-BF(8)"mean job service time vs. System load"figure4.37"	BGP-BF(8)"mean packet blocking time vs. System load"figure4.38"	BGP-BF(8)"mean packet latency vs. System load"figure4.39"	BGP-BF(8)"percent system utilization vs. System load"figure4.40"
One-to-All	One-to-All	All-to-All	All-to-All	All-to-All
All-to-All	All-to-All	NAS	NAS	One-to-All
Random	Random	Random	Random	random
DQBT	DQBT	One-to-All	One-to-All	NAS
NAS	NAS	DQBT	DQBT	DQBT

Notice how the key system parameters; the MJRT, MJST, MPBT, MPL and the PSU change over changing the communication pattern of the scheduled parallel jobs.

The above discussion and set of observations clearly show that, depending on the performance parameter of most interest, we can dynamically modify the partitioning bound of the BGP allocation strategy.

Figures 4.41 through 4.44 revisit what have been studied in previous studies related to the topic of this thesis. In these experiments, the simulated parallel system is loaded (mean inter arrival time is set to a small value).

These studies focus on two types of communication patterns: the "All-to-All" and "One-to-All". The reasons for that are these communication patterns add significant load on the parallel system. In the following discussion, we regenerate previous studies simulation outcomes to insure that the results related to these types of communication patterns are compatible with ours.

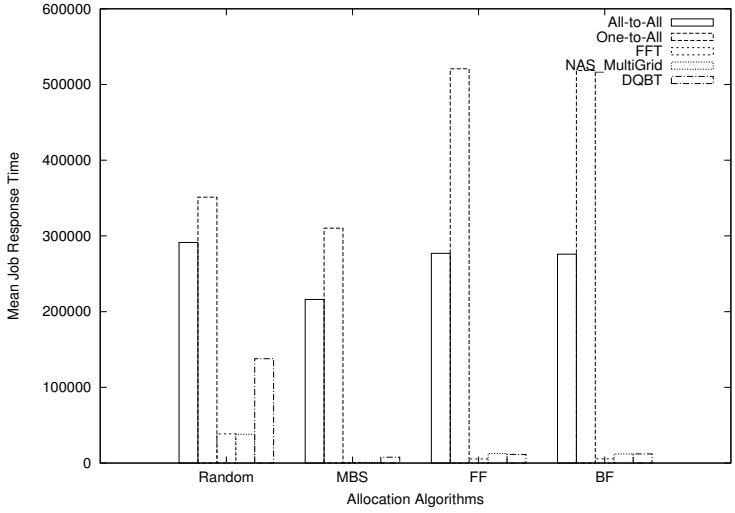


Figure 4.41: Mean job response time VS Allocation Algorithms under FCFS scheduling mechanism and One-to-All, All-to-All communication patterns.

Figures 4.41 and 4.42 represent the Mean job response time and the mean job service time for multiple allocation algorithms for a loaded system. These allocation strategies are: "Random, MBS, FF, and BF". Figure 4.41 clearly shows that the one-to-all communication pattern produces higher MJRT values compared to the Best-Fit allocation algorithm as observed in table 4.3. The figure clearly shows that

noncontiguous allocation strategies have significantly less response time than contiguous allocation strategies, this is also observed in (Bani-Ahmed,2010), (Bani-Ahmed,2011b).

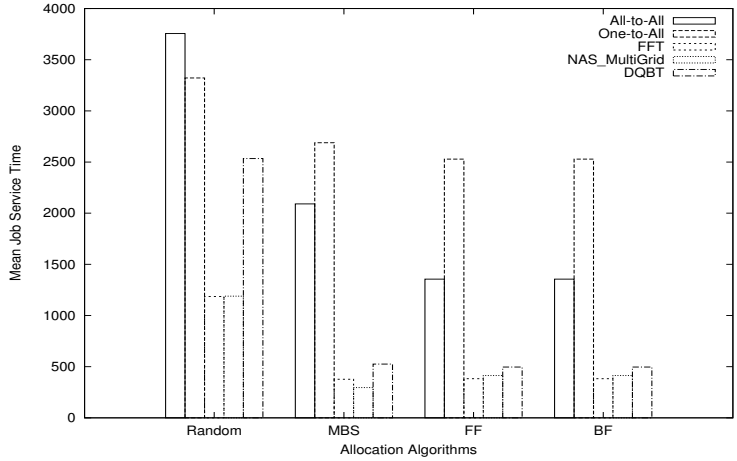


Figure 4.42: Mean job service time VS Allocation Algorithms under FCFS scheduling mechanism and One-to-All, All-to-All communication patterns.

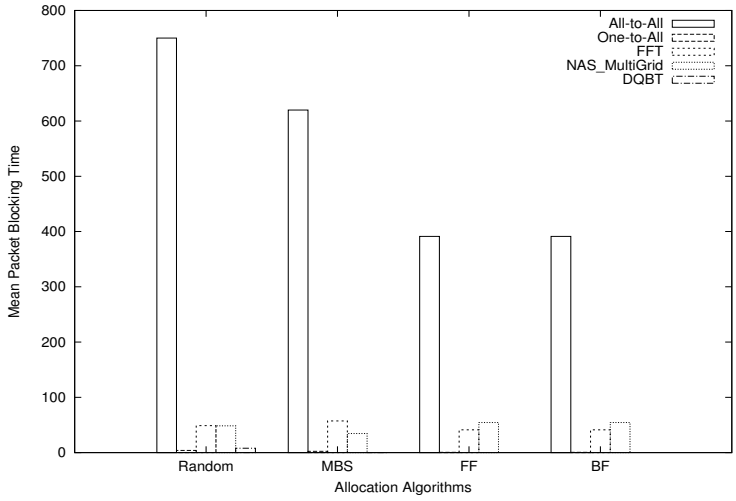


Figure 4.43: Mean Packet Blocking time VS Allocation Algorithms under FCFS scheduling mechanism and One-to-All, All-to-All communication patterns.

Figures 4.43 and 4.44 show the impact of the allocation strategy being used on two performance parameters: the mean packet blocking time and the mean packet latency. MPBT and MPL are the highest when All-to-All communication pattern is involved. This explains why previous studies focus on this communication pattern because it

puts the system in a really heavy communication load (Bani-Ahmed,2010; Bani-Ahmed,2011b; Bani-Mohammed et al,2006) .

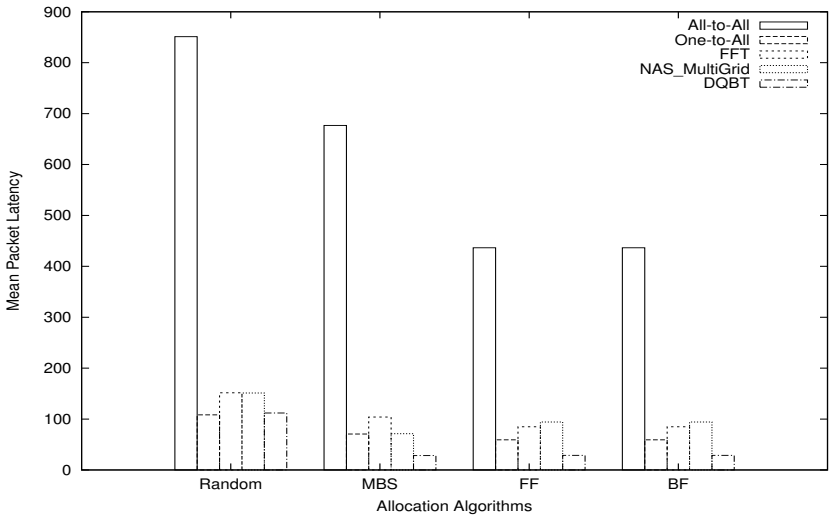


Figure 4.44: Mean Packet Latency time VS Allocation Algorithms under FCFS scheduling mechanism and multiple communication patterns.

The next table shows summary of the figures. In general, table 4.6 clearly shows that the communication pattern is an effective factor that has a significant impact on our choice of which processor allocation strategy to use.

Table 4.6: summarizes on figure 4.44 .

Random	MBS	FFT	BF
All-to-All	All-to-All	All-to-All	All-to-All
FFT/NAS	FFT	NAS	NAS
DQBT	One-to-All	FFT	FFT
One-to-All	NAS	One-to-All	One-to-All
-	DQBT	DQBT	DQBT

Chapter five: Conclusion and future work

Conclusion

- The amount of communication conducted between the parallel jobs to be allocated is a key performance factor that can highlight the difference between allocation strategies. .
- In this thesis, we claim that the type of communication pattern can affect performance of these strategies.
- Comparison of the communication pattern is usually used in literature to evaluate processor allocation strategies.
- We examined wider range of communication patterns in our research, such as (FFT, NAS, Divide and Conquer, Random). Other researches consider only two types of communication patterns; those are the one-to-all and all-to-all patterns.
- We found that the communication behavior of the parallel jobs being allocated can have a significant impact on the performance of the applied processor allocation strategy. So we claim the following:
    - The behavior of the processor allocation strategy used change as the communication behavior of parallel jobs change.
    - The behavior of the processor allocation strategy used change as the communication load caused by the parallel jobs change.

Future work

As a future work of this study, we are working on developing a hybrid processor allocation strategy that combines the advantages of two or more processor allocation strategies studied of this thesis and if the processor allocation module can be informed

ahead of time about the communication pattern and load of the parallel jobs to be allocated.

The other important point that can be discussed in the future is to dynamically change some parameters "numbers use of bound" of the BGP processor allocation strategy that is proposed in literature to make the strategy be adaptive and communication-aware.

References

[2] Alqadi, R. and M. Khammash (2007). "***An Efficient Parallel Gauss-Seidel Algorithm for the Solution of Load Flow Problems***". The International Arab Journal of Information Technology. Vol. 4, No. 2. April 2007.

[3] Bani-Ahmad, S. (2010), "***Sub-mesh Allocation in 2D-Mesh Multicomputer: Partitioning at the Longest Dimension of Requests***". Proceedings of the Fourth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2010). October 25-30, 2010, Florence, Italy.

[4] Bani-Ahmad, S. (2011a). "***Bounded Grand-Request-Based Allocation Strategies in 2D Mesh-based Multicomputers***". International Journal of Digital Content Technology and its Applications. Volume 5, Number 1, January 2011.

[5] Bani-Ahmad, S. (2011b). "***Comparative Evaluation of Request-Partioning-Based Processor Allocation Strategies in 2D Mesh-based Multicomputers***". International Journal of Computer Applications (0975-8887). Volume 26, Number 7, July 2011.

[6] Bani-Ahmad, S. (2011c). "***Processor Allocation with Reduced Internal and External Fragmentation in 2D Mesh-based Multicomputers***". Journal of Applied Sciences. Volume 11, Issue: 6, 2011, pp 943-952.

[7] Bani-Mohammad, S.; Ould-Khaoua, M.; Ababneh, I., and Machenzie, L. (2006), "***Non-contiguous Processor Allocation Strategy for 2D Mesh Connected Multicomputers Based on Sub-meshes Available for Allocation***", Proceedings of the 12th International Conference on Parallel and Distributed Systems (ICPADS'06), vol. 2, IEEE

Computer Society Press, USA, 2006, pp. 41-48.

[8] Chang, C. Y. and Mohapatra, P. (1998), "***Performance improvement of allocation schemes for mesh-connected computers***", Journal of Parallel and Distributed Computing, vol. 52, no. 1, Academic Press, Inc. Orlando, FL, USA, July 1998, pp. 40-68.

[9] Chang, C. Y. and P. Mohapatra (1996), "***An Adaptive Job Allocation Method for the Directly Connected Multicomputer Systems***", Inernational Conference on Distributed Computing Systems, May, 1996.

[10] Chuang, P. J. and N.F.Tzeng (1994),"***Allocating Precise sub-mesh in Mesh-Connected Systems***"IEEE.Trans.on Parallel and Distributed Systems,pp.211-217,Feb,1992.

[11] Ding, J. and L. N. Bhuyan (1993), "***An Adaptive Sub-mesh Allocation Strategy for Two-Dimensional Mesh Connected Systems***", Proceedings of International Conference on Parallel Processing, Vol.11, pp.193-200, Aug.1993.

[12] Duato, J., S. Yalamanchili, and L. Ni. (1997). "***Interconnection Networks***", IEEE Computer Society Press, Los Alamitors, CA, 1997.

[13] Kumar, V., Grama A., Gupta A., and Karypis G. (2003). "***Introduction To Parallel Computing***", The Benjamin/Cummings publishing Company, Inc., Redwood City, California, 2003.

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

[15] Lin, W., V. Lo, K. Windisch, and B.Nitzberg (1994), "***Non-continuous Processor Allocation Algorithms for Distributed Memory Multicomputers***", Proceedings of the 1994 International Conference on Supercomputing, pp.227-236, 1994.

[16] Liu, T., W. K. Huang, F. Lombardi, and L. N. Bhuyan (1995), "***A Sub-mesh Allocation Scheme for Mesh-Connected Multiprocessor Systems***", Proceedings of International Conference on Parallel Processing,vol.II.pp.159-163.Aug.1995.

[17] 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, vol. 8, no. 7, IEEE Press, Piscataway, NJ, USA, July 1997, pp. 712-726.

[18] Mao, W., Chen J., and Watson W. (2005). "***Efficient Subtorus Processor Allocation in a Multi-Dimensional Torus***", Proceedings of the 8th International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), IEEE Computer Society, Washington, DC, USA, 30 November -3 December, 2005, pp. 53-60.

[19] ProcSimity V4.3 User's Manual (1997), University of Oregon, ffp://ftp.cs.uoregon.edu/pub/lo/procsim-man.ps.gz, 1997.

[20] Seydim A. Y. (1998)," ***Wormhole Routing in Parallel Computers***", school of Engineering and Applied Sciences, Southern Methodist University, May 1998.

[21] Sharama, D. D. and D. K. Pradhan (1993), "***A Fast and Efficient Strategy for Sub-mesh Allocation in Mesh-Connected Parallel Computers***", Proceedings of the 5th IEEE symposium on Parallel and Distributed Processing, pp.682-693, Dec.1993.

[22] Srinivasan, T., Seshadri J., Chandrasekhar A., and Jonathan J. (2004). "***A Minimal Fragmentation Algorithm for Task Allocation in Mesh-Connected Multicomputers***", Proceedings of IEEE International Conference on Advances in Intelligent Systems – Theory and Applications – AISTA 2004 in conjunction with IEEE Computer Society, ISBN 2-9599-7768-8, IEEE Press, Luxembourg, Western Europe, 15-18 Nov 2004.

[23] 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 Non-contiguous Processor Partitioning Algorithm***", Proceedings of the Third International Workshop on Applied Parallel Computing, Industrial Computation and Optimization, Springer-Verlag, UK, 1996, pp. 641-650.

[24] Windisch, K.; Miller, J. V.; and Lo, V. (1995), "***ProcSimity: an experimental tool for processor allocation and scheduling in highly parallel systems***", Proceedings of the Fifth Symposium on the Frontiers of Massively Parallel Computation (Frontiers'95), IEEE Computer Society Press, Washington, USA, 6-9 Feb 1995, pp. 414-421.

[25] Yoo, B. S. and Das, C. R. (2002). "***A Fast and Efficient Processor Allocation Scheme for Mesh-Connected Multicomputers***", IEEE Transactions on Parallel & Distributed Systems, vol. 51, no. 1, IEEE Computer Society, Washington, USA, January 2002, pp. 46-60.

[26] Yu, C. and C. R. Das (1994),"***Limit Allocation: An Efficient Processor Management Scheme for Hypercubes***", International Conference on Parallel Processing, vol.II, pp.143-150.1994.

[27] Zhu, Y. H. (1992), "***Efficient Processor Allocation Strategies for Mesh-Connected Parallel Computers***", Journal of Parallel and

Distributed Computing,16,pp 211-337,1992

[28] en.wikipedia.org. (2011), ***Communication latency***. [On-Line], Available: http://en.wikipedia.org/wiki/Latency-(engineering)#Communication -latency.

[29] Jun Ho Bahn, Jungsook Yang and Nader Bagherzadeh (2008)," ***Parallel FFT Algorithms on Network-on-Chips***", Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on, pp.1087-1097.7-9 April 2008.

[30] John W. Tukey, (2012). "***Fast Fourier transform"***. [On-Line], Available: http://en.wikipedia.org/wiki/Fast-fourier-transform

[31] NASA Numerical Aerodynamic Simulation Program, (2009).***" NAS Parallel Benchmarks***",[On-Line], Available: http://en.wikipedia.org/wiki/NAS-Parallel-Benchmarks

[32] David Bailey, Tim Harris, William Saphir, Rob Van der Wijngaart, Alex Woo, Maurice Yarrow (1995),"***The NAS Parallel Benchmark***s ***2.0***",Report NAS-95-020,December,1995.

[33] F. Shimojo, R. K. Kalia, A. Nakano, and P. Vashishta (2005), "***embedded divide-and-conquer algorithm on hierarchical real-space grids: parallel molecular dynamics simulation based on linear-scaling density functional theory***," Comput. Phys. Commun. 167, 151.

[34] K. Wendy Tang, Member, IEEE, and Sanjay A. Padubidri (1994), " ***Diagonal and Toroidal Mesh Networks***", IEEE TRANSACTlONS ONCOMPUTERS,VOL.43,NO.7,July, 1994

[35] Rolf Riesen (2006), "***Communication patterns***", IEEE In Workshop on Communication Architecture for Clusters CAC'06, Rhodes Island, Greece, April 2006.

[36] Singh, K. & Kilarikar, N. (2012). "***Tcp Flow Control Mechanisms For Single Packet Loss***", paper presented in the Proceedings of the National Conference "NCNTE-2012" at Fr. C.R.I.T., Vashi, Navi Mumbai, Feb. 24-25, 2012. 1

[37] Sethu, H.; Shi, H.; Kanhere S. & Parekh, A. (2000)." ***A Round-Robin Scheduling Strategy for Reduced Delays in Wormhole Switches with Virtual Lanes***", Proceedings of the International Conference on Communications in Computing Las Vegas, Nevada, USA, June 26–29, 2000

[38] Montresor, A.; Caro, G. & Heegaard, P. (2004)." ***Architecture of the Simulation Environment***", research paper, Universit`a di Bologna (Italy).

[39] Malmgren, R.; Hofman, K.; Amaral, L. & Watts, D. (2009). "***Characterizing Individual Communication Patterns***", KDD'09, June 28–July 1, 2009, Paris, France.

[40] Massey, G. & McClintock, J. (2006). "***Packet Size Matters in IP Transport***", Copyright 2006 IMAS Publishing (USA), Inc.

[41] Gorinsky, S. & Turner, J. (2004). "***Selecting the Buffer Size for an IP Network Link***", Research paper, Department of Computer Science and Engineering, Washington University.

[42] Ababneh Ismail, (2006). "***An efficient free-list sub-mesh allocation scheme for two-dimensional mesh-connected multicomputers***", Journal of Systems and Software, v.79 n.8, p.1168-1179, August 2006 .

Appendices

Appendix 1 ProcSimity Simulation environments

In this appendix, the researcher discusses the parameters used and he also explains the meaning of all parameters' used in this thesis.

The parameters describe every detail of the conducted simulation, such as architecture of the machine, profile of the job streams, used resource allocation policies and how extensive the simulation should be (ProcSimity V4.3 User's Manual, 1997).

1. A: Architecture

The major goal of the current simulation environment is to enable the composition of complex simulators, without incurring in excessive overhead in terms of both memory and time. To achieve this goal, the architecture of the simulation environment is considered as light-weight as possible. The idea of the simulation is to provide a configuration service, which will enable the construction of simulators by "juxtaposing" together with pre-existing or newly-developed components. In order to allow developers writing their customized and optimized version when needed, each component of the simulator will be interchangeable. Moreover, components will be specified through the object oriented programmatic interfaces, methods of which will describe the expected behavior of a component. These interfaces will be as simple as possible, first, to avoid the burden for developers of implementing complex interfaces; second, to avoid the memory overhead that can be associated with complex implementations (Montresor, et al, 2004).

Common configuration was used for the simulation application; this configuration is stored as files, and can be adjusted directly by editing those files, or by using simulation utility like configuration manager (Montresor, et al, 2004).

Thus, this parameter "architecture" specifies topology and dimensions of the architecture; the first parameter number describes "type of topology", the second parameter number describes the dimensions (ProcSimity V4.3 User's Manua,1997).

These dimensions are expressed by width and height for mesh, or K and n for k-ary n-cube, thus when we write 0 that means use mesh topology, or when we write 1 that means cube architecture. After this, width and height must be specified to explain the mesh architecture, when this statement in this simulation is used, the user must write (0 width height) (ProcSimity V4.3 User's Manua,1997).

1. B: Packet size

The Packet size parameter describes the number of 1-byte flits in each packet sent through the network (ProcSimity V4.3 User's Manua, 1997).

In general, as with all packet-based systems, there is a tradeoff between the size of the packet, the network traffic/congestion and the processing overhead. Smaller packets increase overhead with the additional byte required to populate the packet header. Smaller packets are transported through the network at the same speed but an entire packet can be received more quickly due to its size. The difficulty with smaller packets is summarized in increasing likelihood that a packet will be lost or received out of sequence (Massey & Mcintock, 2006).

On the other hand, larger packets have the opposite problem once they exceed the Ethernet limit of around (1500) bytes. Then, the packet is fragmented and transmitted in two or more parts, before being re-assembled at the opposite end, this complicates the receiver with no real benefit (Massey & Mcintock, 2006), it is recommended to use 8 in this simulation, therefore use the value of this parameter as 8.

1. C: Flow Control Mechanism:

The Flow Control Mechanism that determines the way of flow control which is used in the network switching elements, the ways to be used, such as Virtual- Cut -Through, Wormhole and Store -and -Forward (ProcSimity V4.3 User's Manua, 1997).

To explain the principles' of the Store -and –Forward work, a message can be divided and sent as fixed-length packets in which the first packet including the routing and control information. Each packet individually routed from source to destination, a packet is completely buffered (like mail service) at each intermediate node before it is forwarded to the next node, this is reason why this switching technique is called Store -and –Forward switching. In addition to decrease, the amount of time spent transmitting data and full buffer requirement, Virtual- Cut -through method is introduced in which a packet is stored at an intermediate node only if the next required channel is busy. As soon as enough information is available in the intermediate nodes, forwarding begins even before the entire message is received. Thus wormhole this instead of storing a packet completely in a node and transmitting it to the next node, it's operated by advancing the head of a packet directly from incoming to outgoing channels of the routing chip. A packet is divided into number of flits (flow control digits) for transmission (Seydim, 1998).

In order to use this method, the user should write the number and represent the method used, the numbers are (0, Store -and –Forward), (1, Virtual- Cut –Through), (2, Wormhole) in simulation (ProcSimity V4.3 User's Manua, 1997).

In general, Flow-control Mechanisms have two phases: the *slow start* phase and *congestion avoidance* phase. These mechanisms modify the performance of TCP's sliding window to solve part of the problems related to congestion in networks and their nodes. If TCP works without the *Slow Start* and *Congestion Avoidance*

algorithms, it will start sending as much information as the usable window allows. This could create problems because nodes along the network have to store information in their queues and can run out of memory. When a node cannot store incoming data, segments are lost and transmission errors occur (this is called congestion). Then, the sender has to retransmit the lost segments, and so it does, but without any other flow control, it will fill up the usable window immediately, thus causing congestion once again (Singh & Kilarikar, 2012).

Slow Start provides a method to control primary data flow at the beginning of a communication and through an error recovery according to the received acknowledgements. Every acknowledgement reflects a segment, or a group of segments has left the network without using any resources. Otherwise, new data can be sent. Moreover, *Congestion Avoidance* is considered as algorithm, which adapts data transmission to the resources availability. Then Slow Start has reached a certain threshold, Congestion Avoidance begins slowing down the opening of the window (Singh & Kilarikar, 2012).

Noting that if Virtual- Cut –Through or Store -and -Forward are chosen, the Buffer Size parameter (mention below) will be overridden and set to equal the Packet Size (above). Wormhole is recommended in this simulation, so users should write number (2) to choose it (ProcSimity V4.3 User's Manua, 1997).

1. D: Buffer Size

Buffer Size represents every network switching element buffer, the number of flits that can be conducted in it (that recommended equal to Packet Size for Store -and –Forward or Virtual- Cut –Through flow control, or 1 for Wormhole flow control) (ProcSimity V4.3 User's Manual, 1997).

Moreover, Buffer size indicates the size allocated for temporary storage of data or memory in a particular computer. When a particular program needs to be a continuous flow of data or information, a computer buffer is needed as it allows for faster "reading" or data retrieval. The buffer size depends on the size of the work required in running the programs (Gorinsky & Turner, 2004).

1. E: Routing Delay

The Routing Delay is defined as the number of cycles required for a flit to be routed through a network switch to the output channel. (ProcSimity V4.3 User's Manual, 1997).

1. F: Router Type

Router Type determines the type of routing hardware implemented in the network switches. This should correspond with the architecture type, which means that the user must choose number 0 to Mesh XY Routing when mesh topology is used, and when cube architecture choose number 1 is used to specify *k*-ary *n*-cube Lee Distance Routing w/ virtual channels (ProcSimity V4.3 User's Manual, 1997).

1. G: Allocation Strategy

The seventh parameter is *Allocation Strategy* which is considered a very important parameter because it specifies the processor allocation strategy to be used. The choice of allocation strategy should be compatible with the architecture type chosen. In other words, Allocation Strategy can be applied to this simulation, all strategies are represented by numbers, such as (0, Mesh Random Allocation),(1, Mesh MBS),(2, Mesh Paging),(3, Mesh First Fit),(4, Mesh Best Fit),(5, Mesh Frame Sliding),(10, *k*-ary *n*-cube Random Allocation),(11, *k*-ary *n*-cube Buddy),(12, *k*-ary *n*-cube MBS),(13, *k*-ary *n*-cube Paging),(14, *k*-ary *n*-cube Grey Code),(15, *k*-ary *n*-cube

Partner),(16, *k*-ary *n*-cube Multiple Partner) (ProcSimity V4.3 User's Manual, 1997). All mesh strategies have been used.

In paging strategy3, the *page size* is defined as the size of each page, for meshes, pages are $2^{\text{page-size}}$x$2^{\text{page-size}}$. In addition to the indexing scheme that specifies the order in which free pages are selected through Paging allocation in the mesh. In other words, when use number (0) that means Row-Major order, and (1, Shuffled row-major order),(2, Snake-line order),(3, Shuffled snake-line order),the researcher used page size equal 2 and used Snake-line order Indexing schemes for Mesh Paging Allocation (ProcSimity V4.3 User's Manual, 1997).

1. H: Scheduling Strategy

A scheduling strategy is defined as a class capable of scheduling a task for execution. Butterfly Scheduling comes with a few built-in scheduling strategies. Scheduling strategy takes into account the probability of a dependence violation to clarify the number of scheduled iterations (Sethu, et al, 2000).

Moreover, scheduling strategies can be used at the output link or for access to output queues from the input queues. These scheduling strategies are critical to the fairness achieved in the switch, beyond mere elimination of starvation scenarios (Sethu, et al, 2000).

In this simulation, the *Scheduling Strategy* shows job scheduling strategy, which may be used and represented by a number; such as (0, First Come, First Serve(FCFS)),(1, Last Come, Last Server(LCLS)),(2, Shortest Service Demand First (SSD)),(3, Shortest Hold Time First(SHT)),(4, Smallest Job First),(5, Longest Service Demand First(LSD)),(6, Longest Hold Time First(LHT)),(7, Largest Job First),(8, Scan Up) (ProcSimity V4.3 User's Manual, 1997).

In this simulation, the used Job Scheduling Strategy is First Come, First Serve (FCFS) only.

1. I: Job Size Distribution

The *Job Size Distribution* describes the distribution of the number of processors requested by every job. Mesh jobs require width*height processors and k-ary n-cube jobs simply require a scalar number of processors. There are four types of distribution for each topology; each has its own parameters, as follows:

a. First: Mesh Job Size Distributions *Uniform distribution:* job widths and heights uniformly distributed over range (max $_h$.... min $_h$) and (max $_w$ min $_w$) respectively. Max must not exceed the number of processors in the system represented by "writing" (0 min $_w$ max $_w$ min $_h$ max $_h$).

b. Second: *Exponential distribution* means job widths and heights exponentially distributed with mean width and height of mean $_w$ and mean $_h$, respectively.

c. Third: is the Interval distribution that allows the user to specify the size and probability for any four width/height intervals. ai_w/ ai_h specify the upper bounds of each of the four width/height intervals, respectively, and pi_w/ pi_h specify the probability for each interval and expressed as a decimal fraction. Values should increase at all times, not exceeding the number of processors in the system, and p values for each width and height should total 1.00, and this is presented in this simulation by writing (2 $a1_w$ $a2_w$ $a3_w$ $a4_w$ $p1_w$ $p2_w$ $p3_w$ $p4_w$ $a1_h$ $a2_h$ $a3_h$ $a4_h$ $p1_h$ $p2_h$ $p3_h$ $p4_h$) (ProcSimity V4.3 User's Manual, 1997).

d. Fourth: *workload trace input* that read job stream from the input file and called w*orkload-trace file*. Using a workload trace overrides the parameters communication pattern; which means number of messages per job and system load. Using a workload trace overrides the number of jobs parameter, setting it

to the number of jobs specified in the workload trace file, that is written in simulation as (And the (3 workload-trace file) (ProcSimity V4.3 User's Manual, 1997).

e. Fifth: an *exponential base-2 distribution* job widths and heights distributed from base-2 exponential distribution with mean width and height of mean (w) and mean (h), respectively, that's written as (4 mean $_{w}$ mean $_{h}$) (ProcSimity V4.3 User's Manual, 1997).

The researcher used the Exponential distribution in this simulation.

1. J Execution Time Distribution

The *Execution Time Distribution* is the distribution from which random execution times are generated. There are five possible distributions, each one with its own additional parameter, when writing *0 min max*, this means Uniform distribution the execution times distributed uniformly between *min* and *max this option was used by the researcher.* And when writing 1 *mean* (Exponentional distribution) that means execution times are distributed exponentially and having the mean, *mean* (ProcSimity V4.3 User's Manual, 1997).

The other option is *2 mean std-dev* that means Hypereponential distribution through using this option that means execution times distributed hyperexponentially and having the mean, mean*, and* standard deviation*, std-dev* (ProcSimity V4.3 User's Manual, 1997).

In addition to the other option, where we write *3 mean std-dev* (Normal distribution) execution times distributed normally and having the mean, *mean, and* standard deviation, *std-dev*. When we write *4 mean std-dev* that means (Erlang distribution) this means execution times are distributed by Erlang distribution and having the mean, *mean*, and standard deviation, *std-dev* (ProcSimity V4.3 User's Manual, 1997).

1. K: Mean Inter-arrival Time

Mean Inter-arrival Time is defined as the mean for random generation of Poisson inter-arrival times (ProcSimity V4.3 User's Manual, 1997).

1. L: Mean Time between Sends:

The *Mean Time between Sends* is determining the average time for each process of sending a message in the function of implementing the random pattern of contact. (Simulation recommends using: 0.0) (ProcSimity V4.3 User's Manual, 1997).

1. M: Communication Patterns

Communication Pattern is especially for the specific communication pattern that job executes when message is passing.

The communication patterns used in this simulation (0) No communication: delay for random time interval instead, such as (ProcSimity V4.3 User's Manual, 1997):

(1) Is all to all in each process in a job sends a message to each other process in a job,

(2) n-body:ring/chordal n-body algorithm,

(3) One-to-All a randomly selected process in each job sends message to each other process in the job,

(4) random Communication: each process in a job repeatedly sends a message to another randomly selected process in the job and delays for a random interval based on the Mean Time Between Sends,

(5) FFT that's parallel fast fourier transform algorithm,

(6) NAS multigrid benchmark,

(7) Divide and Conquer Binomial Tree algorithm,

(8) NAS Kernal CG Benchmark this is for mesh topologies

Communication patterns may prove to be useful as an additional class of attribute data, complementing demographic and network data, for user classification and

outlier detection—a point that illustrating the interpretable clustering of users based on their inferred model parameters (Malmgren, et al, 2009).

Communication patterns are important for (Riesen, 2006):

- Purchasers of parallel machines, so they can make decisions about which topology and network technology will best fit their applications.
- Application developers and designers of networks and communication software stacks.

1. N: Mean Messages per Job

Mean Messages per Job specifies the number of messages to be sent by each job in one of several ways (ProcSimity V4.3 User's Manual, 1997):

- If the value has the form "*n*", where n is a real number, then value of the parameter is taken as the average number of messages sent by each job executing a communication pattern. The number of messages for each job is taken randomly from an exponential distribution.
- If the value has the form "*nf",* where *n* is a positive integer, then the value of the parameter is taken to be the exact number of messages to be sent by each job in the simulation.
- If the value has the form "*ni*", where *n* is a positive integer, then the value of the parameter is taken to be the exact number of iterations of each jobs communication patter that the job should execute.

1. O: Debugging Level

Debugging Level is the amount of debugging data consisted in the simulation output, the simulation recommends using :0) (ProcSimity V4.3 User's Manual, 1997).

1. P: Number of Runs

Number of Runs specifies the times that each simulation is duplicated for accuracy and establishing confidence intervals that is recommended and use in simulation is: 10 (ProcSimity V4.3 User's Manual, 1997).

1. Q: Number of Jobs

Number of Jobs explains jobs simulated number in each simulation run, its recommended to use: 1000, (ProcSimity V4.3 User's Manual, 1997).

1. R: Trace File name

Trace File name is the name of the file in which simulation trace data is recorded and used to drive the visualization tool. The filename none indicated not to save trace data, in case the visualization tool is not to be used, (ProcSimity V4.3 User's Manual, 1997).

Appendix 2 Simulation output

The researcher believes that the pattern and intensity of communication between parallel jobs can affect the processor allocation strategy that he has chosen.

When running the simulation, the output can display the statistics report, which explainsin a detailed and ordered manner (ProcSimity V4.3 User's Manual, 1997):

- ***Average Response Time***, the average time from when a job arrives in the waiting queue until the time when it is completed. The researcher used five communication pattern in the simulation, "One-to-All, NAS, FFT, all-to-all, Divide and Conquer".
- ***Average Blocks per Job***, which explain the average number of non-contiguous blocks allocated to a job.
- ***Max Blocks per Job,*** which is considered as the maximum number of non-contiguous blocks allocated to a job.

- ***Average Waiting Queue Time,*** the average time that jobs spend in the system-waiting queue before being allocated.
- ***Average Queue Length***, which reflects the average number of jobs in the system waiting queue.
- ***Average Inter-arrival time***, which mean the average time between job arrivals.
- ***Average Packet Blocking Time,*** which means the average time that message packets spends blocked in network buffers, waiting for access to their next channel.
- ***Average Packeting Latency***, which means the average time for message packets to reach their destination once they are injected into the network.
- ***Average Job Size***, which means the average number of job processors.
- ***System Utilization***, which means the percentage of processors that are utilized over time.
- ***Finish Time***, which means the total time for completion of one simulation run.
- The last result that can be seen is the ***Block Demand***, which means the distribution of block sizes for non-contiguous allocation strategies. For each block size, in ascending order from 0 to 4, the percentage of allocated blocks that were of that size.

Appendix 3 The values uses in simulation

TableA.3.1: Summaries of the values of parameters

Architecture	
0 width height	**Mesh architecture(width*height)**
Packet Size	
8	**Packet Size**
Flow Control Mechanism	
2	**Wormhole**
Buffer Size	
1	**Buffer Size**
Routing Delay	
3	**Routing Delay**
Router Type	
0	**Mesh XY Routing**
Allocation Strategy	
0	**Mesh Random Allocation**
1	**Mesh MBS**
3	**Mesh First-Fit**
4	**Mesh Best-Fit**
17	**BGP-FF**
18	**BGP-BF**
Scheduling Strategy	
0	**First Come, First Serve (FCFS)**
Job Size Distribution	
1 $mean_w$ $mean_h$	**Exponential distribution**
Execution Time Distribution	
0 *min max*	**Uniform distribution**
Mean Interarrival Time	
Mean Inter-arrival Time	**0**
Mean Time Between Sends	
Mean Time Between Sends	**0.0**
Debugging Level	
Debugging Level	**0**
Number Of Runs	
Number Of Runs	**10**
Number Of Jobs	
Number Of Jobs	**1000**
Trace Filename	
Trace Filename	**None**