Robust Resource Allocation Techniques on Homogeneous Distributed System

A thesis report submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science

By

Vivek Kumar Bajaj (10506050)
Ajeet Kumar (10506049)

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008
May 2009

National Institute of Technology Rourkela
Rourkela-769008, Orissa

Certificate

This is to certify that the work in this Thesis Report entitled "**Robustness Resource Allocation Techniques on Homogeneous Distributed System**" submitted by Vivek Kumar Bajaj and Ajeet Kumar has been carried out under my supervision in partial fulfillment of the requirements for the degree of **Bachelor of Technology in Computer Science** during session 2005-2009 in the Department of Computer Science and Engineering, National Institute of Technology, Rourkela, and this work has not been submitted elsewhere for a degree.

Place: NIT Rourkela
Date : May 11, 2009

(Bibhu D. Sahoo)
Sr. Lecturer
Department of CSE
NIT Rourkela

Acknowledgements

No thesis is created entirely by an individual, many people have helped to create this thesis and each of their contribution has been valuable. Our deepest gratitude goes to our thesis supervisor, Mr. *Bibhu D. Sahoo*, Sr. Lecturer, Department of CSE, for his guidance, support, motivation and encouragement through out the period this work was carried out. His readiness for consultation at all times, his educative comments, his concern and assistance even with practical things have been invaluable.

We would also like to thank all professors and lecturers, and members of the department of Computer Science and Engineering for their generous help in various ways for the completion of this thesis. A vote of thanks to our fellow students for their friendly co-operation.

(Vivek Kumar Bajaj, 10506050
Ajeet Kumar, 10506049)
B.Tech. (Comp. Sc.),
2005-2009.

Abstract

Distributed computing systems utilize various resources with different capabilities to satisfy the requirements of diverse task mixtures and to maximize the system performance. Such systems often operate in an environment where certain desired performance features degrade due to unpredictable circumstances, such as higher than expected work load or inaccuracies in the estimation of task and system parameters. Thus, when resources are allocated to tasks it is desirable to do this in a way that makes the system performance on these tasks robust against unpredictable changes.

The system is considered robust if the actual makespan under the perturbed conditions does not exceed the required time constraint. The goal is to maximize the collective allowable error in execution time estimation for the tasks that can occur without the makespan exceeding the constraint. A mapping is considered to be *robust* with respect to specified system performance features against perturbations in given system parameters if degradation in these features is within acceptable limits when certain perturbations occur.

A FePIA four-step procedure is outlined for deriving robustness metric for an arbitrary system. Applying the FePIA procedure, robustness metric for problem assumed in distributed computing is derived.

System model assumed here are homogeneous in nature that means all the machines considered here have same characteristics or have same set of configurations. The applications are assumed to be independent. A set of applications is to be mapped on a set of machines so as to minimize the makespan and maximize the robustness metric. The above assumed problem statement has been implemented using FCFS (first come first service) scheduling technique**.** A large number of simulations have been carried out for different system load to measure the performance of the algorithm at different traffic rates- low, moderate, heavy.

CONTENTS

List of Abbreviations

FePIA	Identifying the performance features, the perturbation parameters, the impact of perturbation parameters on performance features and the analysis to determine the robustness
DAG	Directed acyclic graph
QOS	Quality of service
ETC	Estimated time to compute
FCFS	First come first service

List of Figures

Chapter 1 Introduction

1.1 Introduction

Distributed computing is the coordinated use of a machine set (may be same or different computing capabilities), networks, and interfaces to meet the requirements of widely varying application mixtures and to maximize the system performance or cost effectiveness. Such systems often operate in an environment where certain desired performance features degrade due to unpredictable circumstances, such as higher than expected work load or inaccuracies in the estimation of task and system parameters. Thus, when resources are allocated to tasks it is desirable to do this in a way that makes the system performance on these tasks robust against unpredictable changes.

The act of assigning (*matching*) each task to a machine and ordering (*scheduling*) the execution of the tasks on each machine is known as *mapping*, *resource allocation*, or *resource management* [2]. *Metatask* [2] composed of a number of independent tasks (i.e., no communication between tasks are needed) is considered. Makespan (defined as the completion time for an entire set of tasks) is often the performance feature that needs to be optimized in such systems. Resource allocation is typically performed based on estimates of the computation time of each task on each class of machines. A mapping is considered to be *robust* with respect to specified system performance features against perturbations in given system parameters if degradation in these features is within acceptable limits when certain perturbations occur. *The degree of robustness* [1], [2] is the maximum amount of collective uncertainty in perturbed system parameters within which a user-specified level of system performance can be guaranteed. The system is considered robust if the actual makespan under the perturbed conditions does not exceed the required time constraint, denoted by ζ . The time span of the

tasks that is left with the system to execute when another set of tasks arrives and need to be mapped is termed as *load* on the system.

1.2 Robustness as a performance metric

Different types of standard performance metrics such as makespan, throughput etc are used to evaluate system's performance. For a given distributed computing system, robustness can be used as a quality factor to evaluate performance, since; robustness gives an overall idea of the fault-tolerance capacity of the system.

The term robustness is used to characterize the process working or the algorithmic behaviour whose objective is to reach the alternative set ranking but in the presence of uncertainty.

- Mathematically , Robustness radii,

$$r_\mu(F_j, C) = \min_{C:\, F_j(C)=\tau} | C - C^{est} |$$

- The *robustness metric*, denoted by the $\rho_\mu(\varphi, C)$ for the mapping is simply the minimum of all robustness radii.
- Mathematically,

$$\rho_\mu(\varphi, C) = \min \; r_\mu(F_j, C)$$

1.3 Literature Survey

The work presented in this thesis is built upon the four-step FePIA procedure, where the abbreviation stands for identifying the performance features, the perturbation, the impact of perturbation parameters on performance features, and the analysis to determine the robustness, detailed in [1].The FePIA presents a step-wise method to derive a generalized robustness metric later described in article 2.1) and this is applied to our problem domain. In literature, a number of papers have studied the issue of robustness in distributed systems (e.g.[4],[5],[6]). H.J. Siegel presented a survey of different heuristic algorithms for robust resource allocation for independent tasks.

Reference [4] addresses the problem of matching and scheduling of DAGstructured application to both minimize the makespan and maximize the robustness in a heterogeneous computing system. Due to the conflict of the two objectives, it is usually impossible to achieve both goals at the same time. We give two definitions of robustness of a schedule based on tardiness and miss rate. Slack is proved to be an effective metric to be used to adjust the robustness. We employ ϱ-constraint method to solve the bi-objective optimization problem where minimizing the makespan and maximizing the slack are the two objectives. Overall performance of a schedule considering both makespan and robustness is defined such that user have the flexibility to put emphasis on either objective. Experiment results are presented to validate the performance of the proposed algorithm. Reference [5] investigates the problem of robust resource allocation for a large class of systems operating on periodically updated data sets under an imposed quality of service (QoS) constraint. Such systems are expected to function in an environment replete with uncertainty where the workload is likely to fluctuate substantially. Determining a resource allocation that accounts for this uncertainty in a way that can provide a probabilistic guarantee that a given level of QoS is achieved is an important research problem. First, this paper defines a methodology for quantifiably determining a resource allocation's ability to satisfy QoS constraint in the midst of uncertainty in system parameters. Uncertainty in system parameters and its impact on system performance are modeled stochastically. Second, the established stochastic model is employed to develop greedy resource allocation heuristics. Finally, the utility of the proposed stochastic robustness metric and the performance of the heuristics are evaluated in a simulated environment that replicates a heterogeneous cluster-based radar system. A reference [6] describes the model used for the NewsWire collaborative content delivery system. The system builds on the robustness and scalability of Astrolabe to weave a peer-to-peer infrastructure for real time delivery of news items. The goal of the system is to deliver news updates to hundreds of thousands of subscribers within tens of seconds of the

moment of publishing. The system significantly reduces the compute and network load at the publishers and guarantees delivery even in the face of publisher overload or denial of service attacks.

1.4 Problem Formulation

The work is focused on a distributed computing environment. In simple words we need to specify/find a robust algorithm for allocating resources to distributed computing system. The work consists of comparison between different algorithms of resource allocation on the basis of robustness metric in a distributed computing environment. The algorithm needs to make the system fault-tolerant. Fault-tolerance of the system is a measure of the robustness of the algorithm.

Depending on the Robustness of various resource allocation techniques, FCFS, Round-Robin, we evaluate the performance of these algorithms.

1.5 Approaches

A distributed system is a collection of independent computers (processors) that appear to the users of the system as a single computer. A communication network connects these computers. Each computer has its own memory and other resources. The processors can communicate through message passing over the communication network. The distributed system is modeled as a graph where vertex set represents the set of processors and the edges represent the interconnections (communication links) among the processors. A processor can send and/or receive message to/from processors that are only adjacent to it (i.e. connected by an edge). The communicational links are bi-directional.

The distributed (system) network studied is modeled as an anonymous network by an undirected, connected, simple graph G= (V, E), where the vertex set, V= {v_1, v2… v_n} represents the processors in the network and the edges E represent the bi-directional links among the processors. An edge e in E is represents the bi-directional links among the processors. An

edge e in E is represented by (u, v), if e connects u-v for all (u, v) in G. let G denote the set of all such graphs (networks) [3].

It is assumed that the distributed system is a completely connected graph as a logical topology, i.e. messages are sent directly from one node to another node without intervention from the intermediate nodes. Some nodes may be associated with fault.

Faults come into picture when we talk about Real-time Distributed Systems (RTDS). Faults leads to failures. Failures can be either external or internal. Functional models to failure models are listed in the following table. These failures form a hierarchy with crash fault being simplest and the most restrictive type and Byzantine being least restrictive.

Type of Failure	Description
Crash failure	A server halts, but is working correctly until it halts.
Omission failure *Receive omission* *Send omission*	A server fails to respond to incoming requests. A server fails to receive incoming messages. A server fails to send messages.
Timing failure	A server's response lies outside the specified time interval.
Response failure *Value failure* *State transition failure*	A server's response is incorrect. The value of the response is wrong. The server deviates from the correct flow of control.
Byzantine failure	A server may produce arbitrary responses at arbitrary times.

Different type of failures in RTDS [3]

An occurrence of fault leading to failure in the system brings the concept of robustness into limelight. Value of robustness metric gives an idea of the fault-tolerant capacity of the system.

1.6 Layout of Thesis

There are number of algorithms proposed for resource allocation for Distributed Computing System (DCS). Some of them are briefly give in the previous section. Many consider the makespan and some considers the throughput for representing the allocation scheme. But our work considers the robustness. In this work we consider one of the traditional algorithms First come first serve basis (FCFS) heuristic based algorithms for allocating resources to a homogeneous distributed system in a robust manner and study the performance at different traffic rates of arrival of the tasks**.**

This thesis is divided into five chapters. First one is the Introduction chapter, which is this chapter. It includes introduction to resource allocation problem and literature survey, which discusses different models of resource allocation for DCS. Then the second chapter is the Robustness and Distributed Computing, which includes the mathematical derivation of robustness metric for proposed model used. The third chapter contains the problem statement and the algorithm used to allocate the resources robustly to the system. It analyses the behavior of the algorithm under different conditions. The fourth chapter contains the conclusion and future work of our work

1.7 Conclusion

Distributed Computing System is a loosely coupled system so that the execution of one does not affect the other. A number of faults occurring in the system may lead to failure of completion of a job. We need to make the system fault-tolerant so that a set of tasks may not be completed in expected time but will definitely be completed within the given time-constraint. Thus, the system that completes the execution of the set within the given time-constraint is considered to be robust. Mathematical representation of Robustness metric can be used to quantitatively measure the value of robustness, so that performance of different resource algorithms can be compared. Therefore, Robustness metric can be used as a quality factor for performance.

Chapter 2 Robustness Metric and Distributed System

2.1 Introduction

Parallel and distributed systems may operate in an environment where certain system performance features degrade due to unpredictable circumstances, such as sudden machine failures, higher than expected system load, or inaccuracies in the estimation of system parameters [2]. An important question then arises: given a system design, what extent of departure from the assumed circumstances will cause a performance feature to be unacceptably degraded? That is, how robust is the system? First we need to define and formulate an equation to measure robustness. According to different researchers robustness can be defined in different ways. Robustness is the degree to which a system can function correctly in the presence of inputs different from those assumed in [1]. Robustness guarantees the maintenance of certain desired system characteristics despite fluctuations in the behavior of its component parts or its environment [1]. Robustness should be defined for a given set of system features, with a given set of perturbations applied to the system [1]. So definition of robustness can be customized according to different research investigation. Parallel and distributed computing is the coordinated use of different types of machines, networks, and interfaces to meet the requirements of widely varying application mixtures and to maximize the system performance or cost effectiveness [1], [2]. An important research problem is how to determine a mapping (as described in Chapter 1.1) so as to maximize robustness of desired system features against inaccuracies in estimated system and environment parameters. Research addresses the design of robustness metric for mappings. The mathematical description of a metric evaluates the robustness of a mapping with respect to certain system performance features against perturbations in system components and environmental conditions. The four-step FePIA procedure

employed to derive robustness metrics for distributed systems is described in detail in the next article [1].

Glossary of Notation:

Φ	The set of all performance features
ϕ_i	The *i*-th element in Φ
$<\beta_i^{min}, \beta_i^{max}>$	A tupelo that gives the bounds of the tolerable variation in ϕ_i
$\prod$	the set of all perturbation parameter
π_j	the *j*-th element in $\prod$
μ	a mapping
$r_\mu(\phi_i,\pi_j)$	the robustness radius of mapping μ with respect to ϕ_i against π_j
$\rho_\mu(\Phi,\pi_j)$	the robustness of mapping μ with respect to Φ against π_j
A	the set of application
M	the set of machine

2.2 Design of Generalized Robustness Metric

A general mathematical formulation of a robustness metric that could be applied to a variety of distributed systems by following the four-step derivation procedure. Following are general four-step FePIA procedure for deriving such robustness metric for any desired computing environment.

Step 1

To describe the *robustness requirement* that determines the system performance features that should be limited in variation. For example, the robustness requirement could be that the makespan of the mapping should not increase more than 30% beyond its predicted value (expected in the absence of uncertainty). In that case, the system performance features that should be limited in variation are the finishing times for all machines in the system. Mathematically, let Φ be the set of such system features. For each element $\emptyset_i \in \Phi$, quantitatively describe the tolerable variation in $\emptyset_i$. Let $<\beta_i^{min}, \beta_i^{max}>$ be a tuple that gives the bounds of the tolerable variation in the system feature $\emptyset_i$. For the makespan example, $\emptyset_i$ is the time the i-th machine finishes its assigned applications, and its corresponding $<\beta_i^{min}, \beta_i^{max}>$ tuple could be (0, 1.3 × (predicted makespan value))

Step2

To determine the system and environment perturbation parameters against which the robustness of the system features described in step 1. For the makespan example above, the resource allocation (and its associated predicted makespan) was based on the estimated application execution times. It is desired that the makespan be robust (stay within 130% of its estimated value) with respect to uncertainties in these estimated execution times. Mathematically, let $\prod$ be the set of such system and environment parameters. It is assumed that the elements of $\prod$ are vectors. Let π_j be the j-th element of $\prod$. For the makespan example, π_j could be the vector composed of the actual application execution times, i.e., the i-th element of π_j is the actual execution time of the i-th application on the machine it was assigned.

Step3

To identify the impact of the perturbation parameters in step 2 on the system performance features in step 1. For the makespan example, the sum of the actual execution times for all of the applications assigned a given machine is

the time when that machine completes its applications. Mathematically, for every $\emptyset_i \in \Phi$, determine the relationship $\emptyset_i = f_{ij}(\pi_j)$, if any, that relates $\emptyset_i$ to π_j . In this expression, f_{ij} is a function that maps π_j to $\emptyset_i$. For the makespan example, $\emptyset_i$ is the finishing time for machine m_i, and f_{ij} would be the sum of execution times for applications assigned to machine mi. Note that this expression assumes that each $\pi_j \in \prod$ affects a given $\emptyset_i$ independently.

Step4

To determine the smallest collective variation in the values of perturbation parameters identified in step 2 that will cause any of the performance features identified in step 1 to violate the robustness requirement. Mathematically, for every $\emptyset_i \in \Phi$, determine the *boundary values of* π_j , i.e., the values satisfying the boundary relationships $f_{ij}(\pi_j) = \beta_i^{min}$ and $f_{ij}(\pi_j) = \beta_i^{max}$.

Specifically, let π_j^{orig} be the value of π_j at which the system is originally assumed to operate. However, due to inaccuracies in the estimated parameters and/or changes in the environment, the value of the variable π_j might differ from its assumed value. Figure 1 illustrates this concept for a single feature, $\emptyset_i$, and a 2-element perturbation vector. $\pi_j \in R^2$. The curve shown in Figure 1 plots the set of boundary points $\{\pi_j \mid f_{ij}(\pi_j) = \beta_i^{max}\}$ for a mapping μ. For this figure, the set of boundary points $\{\pi_j \mid f_{ij}(\pi_j) = \beta_i^{min}\}$ is given by the points on the π_{j1}-axis and π_{j2}-axis. The point on the curve marked as $\pi_j^*(\emptyset_i)$ has the feature that the Euclidean distance from π_j^{orig} to $\pi_j^*(\emptyset_i)$, $|\pi_j^*(\emptyset_i) - \pi_j^{orig}|$, is the smallest over all such distances from π_j^{orig} to a point on the curve. An important interpretation of $\pi_j^*(\emptyset_i)$ is that the value $|\pi_j^*(\emptyset_i) - \pi_j^{orig}|$ gives the largest Euclidean distance that the variable π_j can change in *any* direction from the assumed value of π_j^{orig} without the performance feature $\emptyset_i$ exceeding the tolerable variation. Let the distance $|\pi_j^*(\emptyset_i) - \pi_j^{orig}|$ be called the robustness radius $r_\mu(\emptyset_i, \pi_j)$, of $\emptyset_i$ against π_j.

Mathematically, $r_\mu(\emptyset_i , \pi_j) =$ min $| \pi_j^*(\emptyset_i) - \pi_j^{orig} |$

π_j: (fij(π_j) = β_i^{min} V fij(π_j) = β_i^{max})

" $r_\mu(\emptyset_i , \pi_j)$ " is called *to be the robustness metric for the robustness of mapping* μ *with respect to performance feature* $\emptyset_i$ *against the perturbation parameter* π_j.

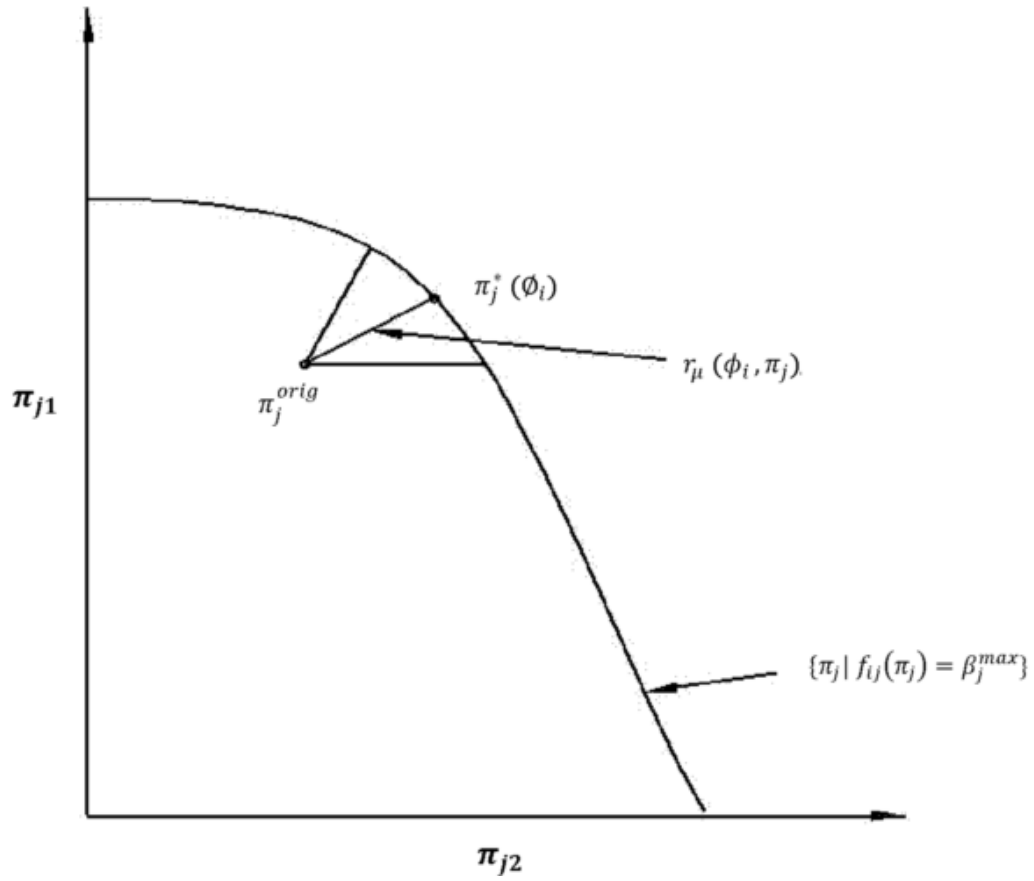

Fig 2.1

The metric definition can be extended easily for all $\emptyset_i$ ϵ Φ. It is simply the minimum of all robustness radii. Mathematically, let $\rho_\mu(\Phi, \pi_j)$ be the *robustness of mapping* μ *with respect to the performance feature set* Φ *against the perturbation parameter* π_j. Then,

$$\rho_\mu(\Phi, \pi_j) = \min_{\emptyset_i \epsilon \Phi} (r_\mu(\emptyset_i , \pi_j))$$

2.3 Derivation of Robustness Metric for Distributed System

The FePIA procedure from [1] is applied to determine the robustness metric for following problem.

- A set of T tasks in the Metatask is required to be allocated to a set of M machines.
- The estimated time to compute (ETC) value for each task on each class of machine is assumed to be known a priori.
- Unknown inaccuracies in the ETC values are expected. So, it is required that the mapping, denoted by μ, be robust against them.
- The finish time of a given machine *j*, denoted by *Fj* , depends only on the actual computation times of the tasks mapped to that machine. The performance feature *(Ø)* that determines if the makespan is robust is the finish times of the machines. That is φ = { F_j | 1≤ j ≤ M }
- The robustness radius [3] for machine *j* of *Fj* against *C* for mapping μ, denoted by $r_\mu(F_j,C)$, is defined as the largest Euclidean distance by which *C* can change in any direction from the assumed point C^{est} without the finish time of machine *j* exceeding the tolerable variation.
- Robustness radius is the minimum Euclidean difference in *C* from the assumed value of C^{est} within which the finish time of machine *j* can reach the tolerable variation.
- Mathematically ,

$$r_\mu(F_j,C) = \min_{C:\, F_j(C)=\tau} | C - C^{est} |$$

- The *robustness metric*, denoted by the $\rho_\mu(\varphi, C)$ for the mapping is simply the minimum of all robustness radii.
- Mathematically,

$$\rho_\mu(\varphi, C) = \min_{F_j \in \varphi} r_\mu(F_j , C)$$

2.4 Conclusion

The system is considered robust if the actual makespan under the perturbed Conditions does not exceed the required time constraint. To measure the robustness of system a metric is defined. There is mathematical description of a metric that evaluates the robustness of a mapping with respect to certain system performance features against perturbations in system components and environmental conditions. With number of research FePIA a four-step procedure for deriving robustness metric for an arbitrary system has been proposed.The first step is to describe the robustness requirement. Second step is to determine the system and environment perturbation parameters against which the robustness of the system features described in step 1. Third step is To Identify the impact of the perturbation parameters in step 2 on the system performance features in step 1. Last step is to determine the smallest collective variation in the values of perturbation parameters identified in step 2 that will cause any of the performance features identified in step 1 to violate the robustness requirement. Then FePIA four-step procedure is employed to derive robustness metric for distributed systems.

Chapter

3 Performance of Robust Resource Allocation Technique

3.1 Introduction

The derivation of the robustness metric as described is chapter 2 is for a system that maps a set of independent applications on a set of machines. In this system, it is required that the makespan (defined as the completion time for the entire set of applications) be robust against errors in application execution time estimates.

System model assumed here are homogeneous in nature that means all the machines considered here have same characteristics or have same set of configurations. The applications are assumed to be independent, i.e., no communications between the applications are needed. The set A of applications is to be mapped on a set M of machines so as to minimize the makespan and maximize the robustness metric ($\rho_\mu(\varphi, C)$). Because larger the robustness metric better is the mapping. Each machine executes a single application at a time (i.e., no multi-tasking), in the order in which the applications are assigned. Assumed problem can be implemented by number of traditional algorithms like FCFS, Round Robin, and Min-Max etc. In this Thesis, it has been implemented using FCFS approach.

3.2 Problem Statement

A set of T tasks in the metatask (set of independent tasks) where there is no communications between the tasks are needed is required to be allocated to set of M of machines while minimizing the makespan and maximizing the robustness metric ($\rho_\mu(\varphi, C)$).

3.3 Algorithm

The above assumed problem statement has been implemented using FCFS (first come first service) scheduling technique.

Steps:

1. A number of independent tasks were generated with random time span (0-1 sec) using Poisson distribution.
2. These tasks were mapped to a set of machines (count = 3) at each time interval with a maximum mapping size of 10 and time constraint τ was set to be 5.The tasks are mapped to the machine on first come first serve basis.
3. For each machine the robustness radii is calculated.
4. Minimum of the robustness radii among the individual machines is robustness metric of the whole system for a mapping.
5. Steps 1-5 are repeated for a number of mappings.
6. Plot of robustness vs. load, robustness vs. no of tasks, robustness vs. time span of tasks, time span of tasks vs. no of tasks are drawn to understand the behavior.

   The above mentioned algorithm was implemented for different traffic rates- heavy, moderate, low traffic and nature was studied.

3.4 Graphical Results

Heavy Traffic:

Robustness vs. load:

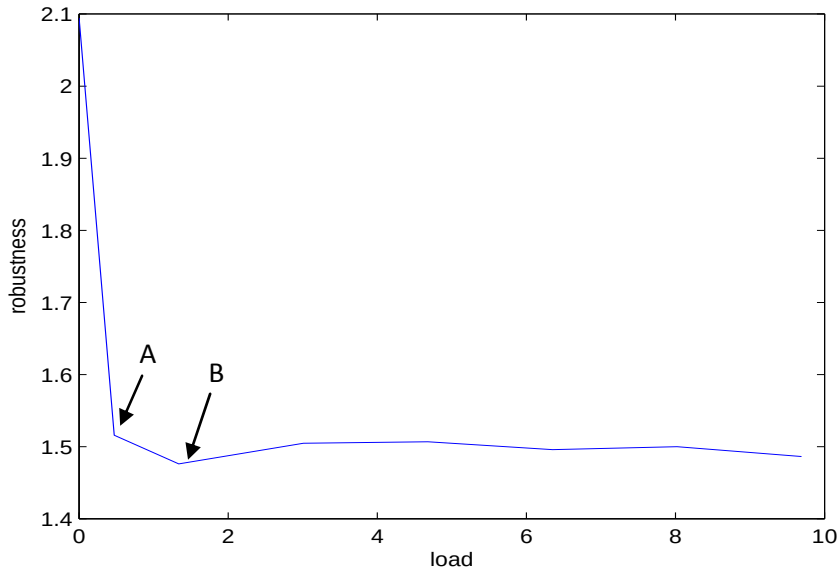

Fig 3.1

Robustness vs. estimated time span of tasks:

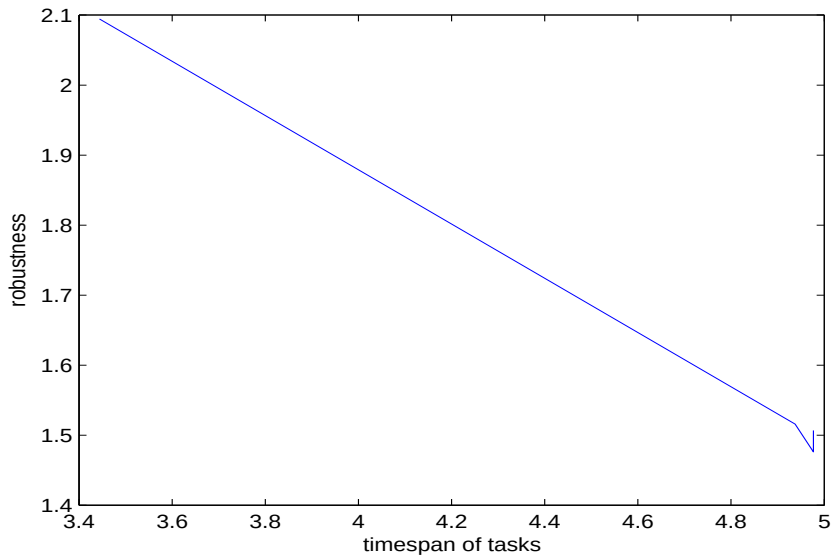

Fig 3.2

Robustness vs. no of tasks:

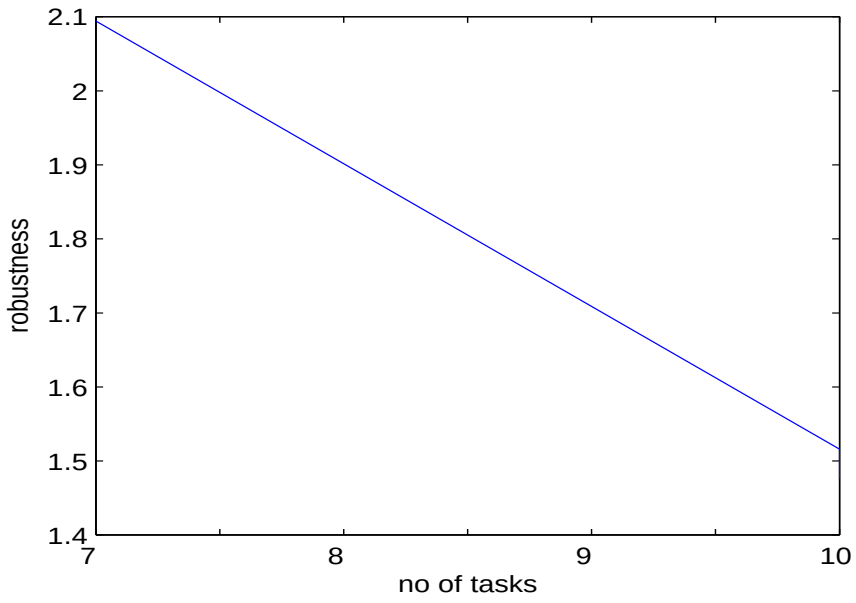

Fig 3.3

Estimated Time span of tasks vs. no of tasks:

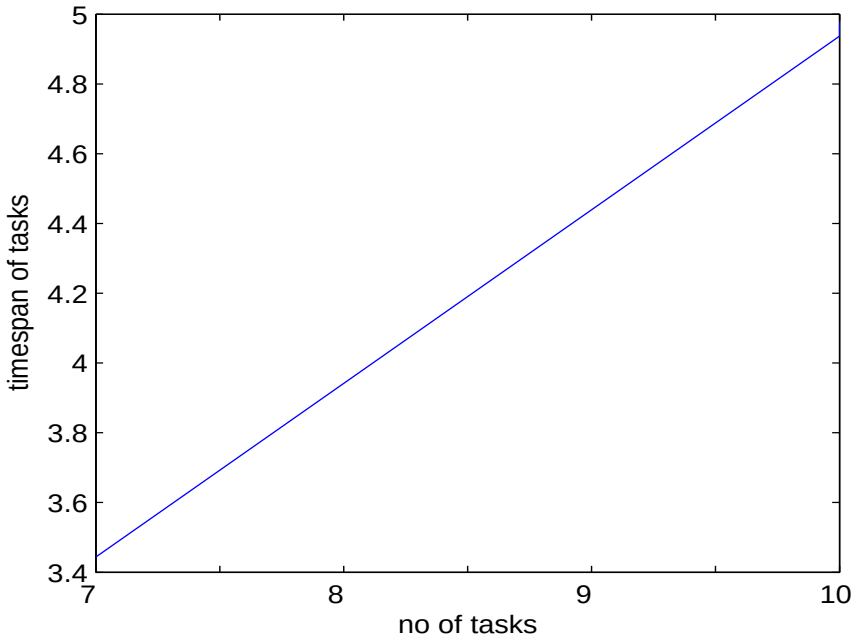

Fig 3.4

Moderate Traffic:

Robustness vs. load:

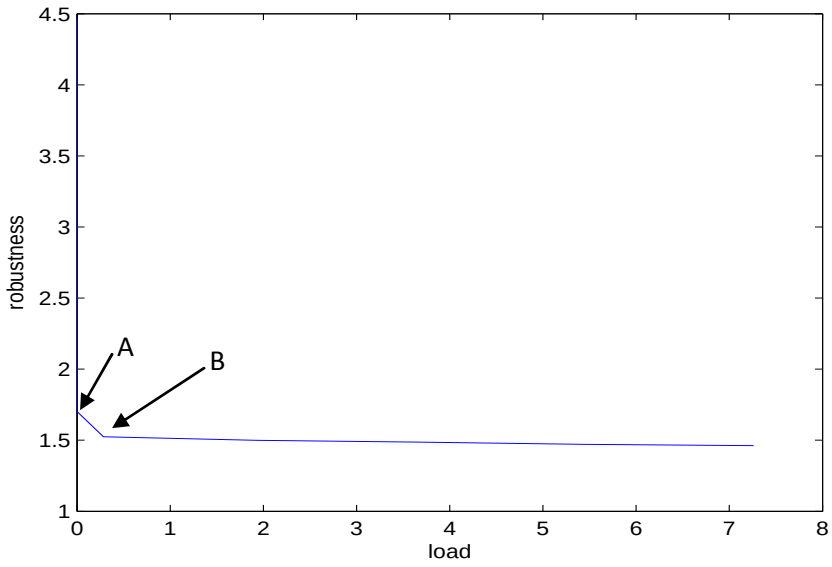

Fig 3.5

Robustness vs. estimated time span of tasks:

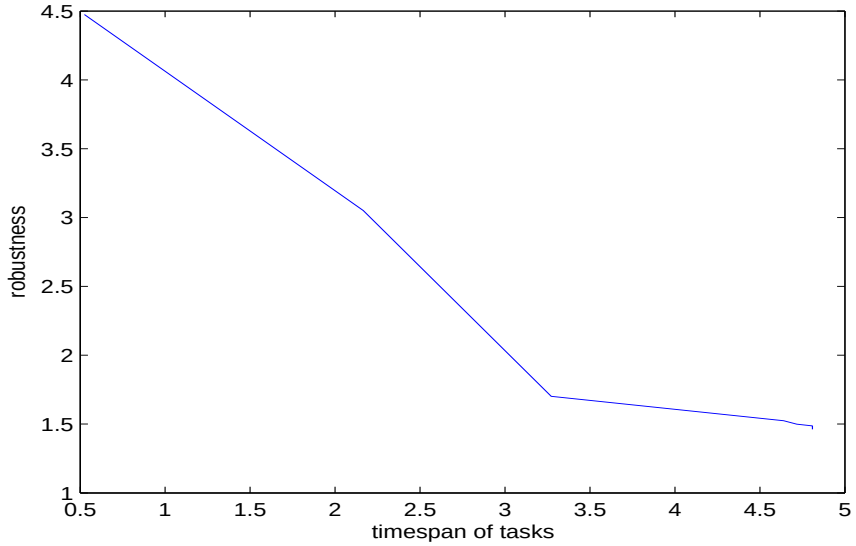

Fig 3.6

Robustness vs. no of tasks:

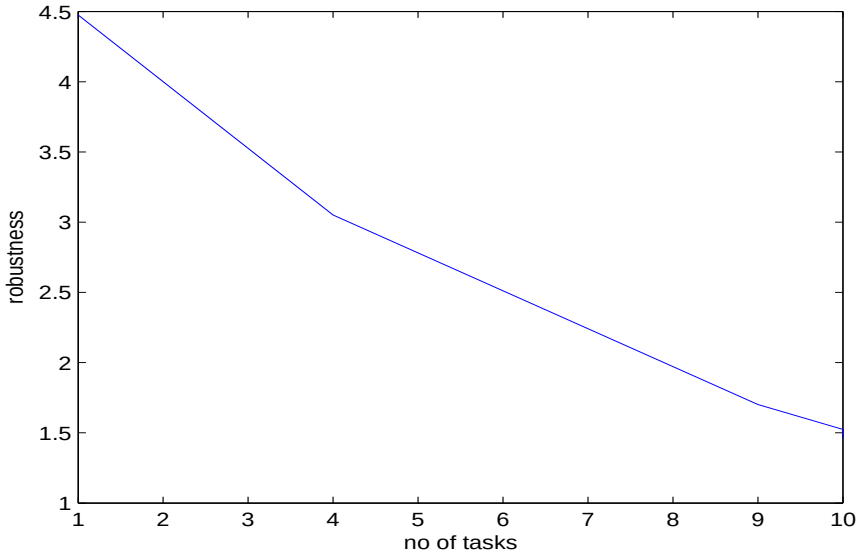

Fig 3.7

Estimated Time span of tasks vs. no of tasks:

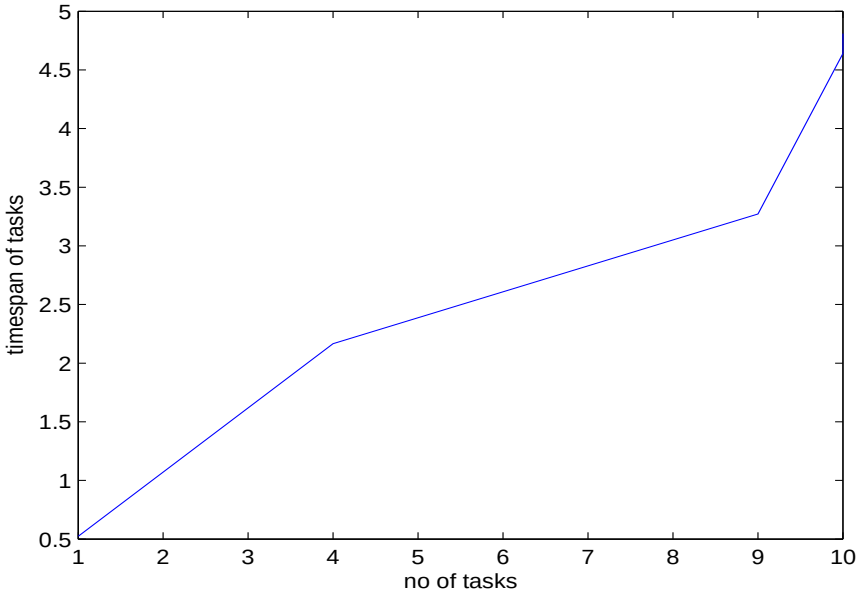

Fig 3.8

Low Traffic:

Robustness vs. load:

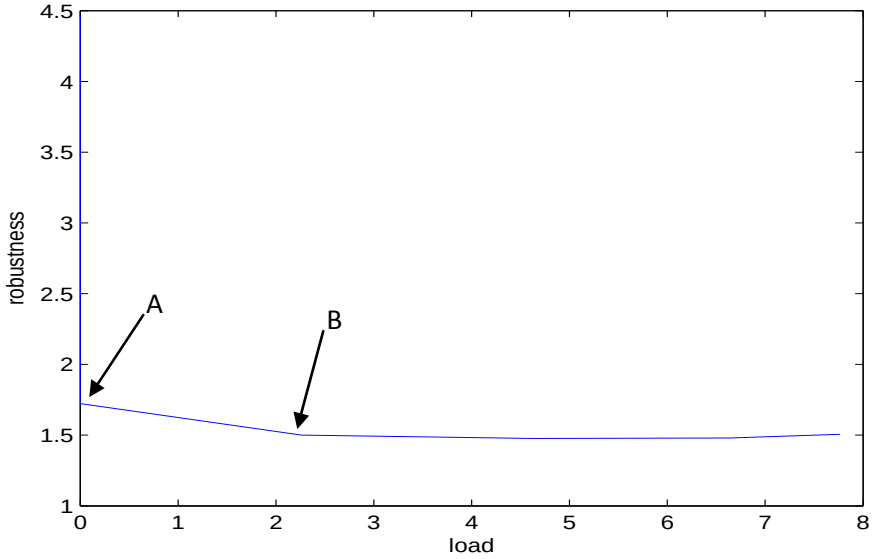

Fig 3.9

Robustness vs. estimated time span of tasks:

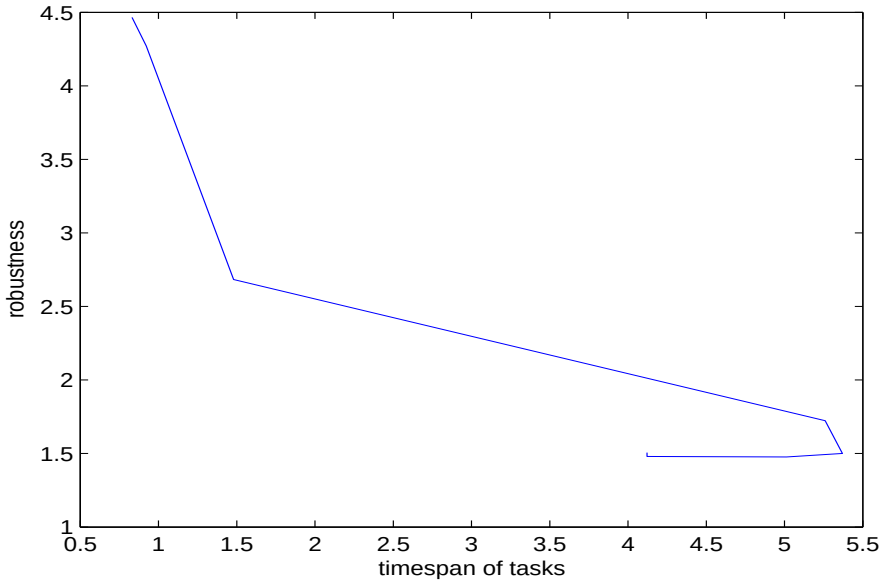

Fig 3.10

Robustness vs. no of tasks:

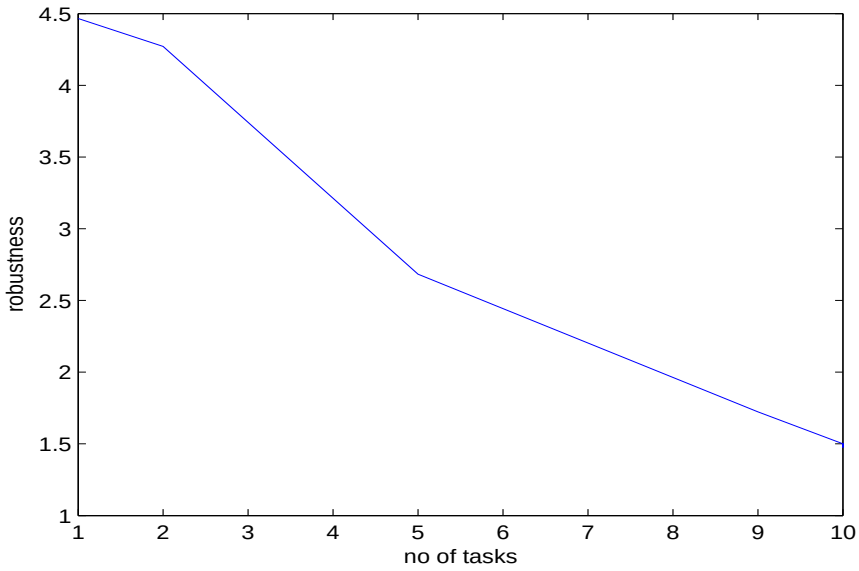

Fig 3.11

Estimated Time span of tasks vs. no of tasks:

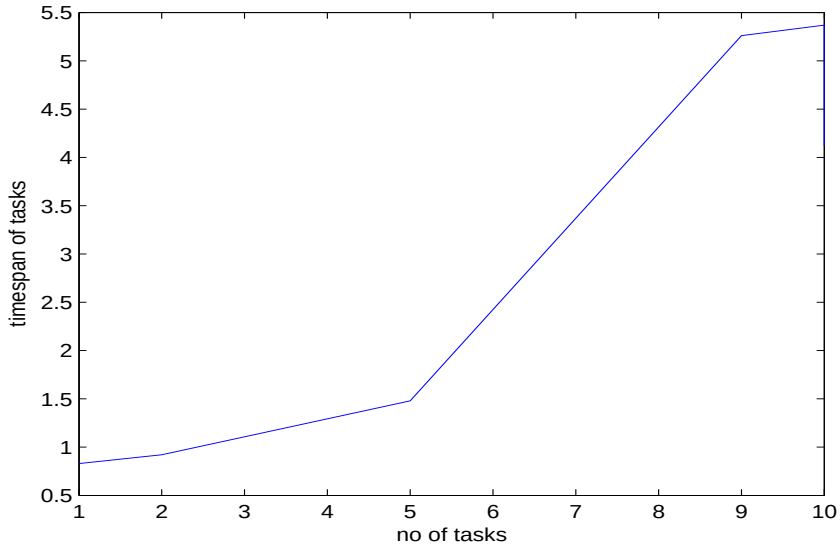

Fig 3.12

3.5 Conclusion

Experimental results are in accordance with the mathematical formula used. Robustness depends on the load of the system, the load to be mapped and the no of tasks being mapped. Experiment was conducted at different rates of traffic. It was found that after a long duration of time the robustness of the system saturates at around 1.5.

Robustness decreases with the increase in current load of the system and the load to be mapped. The graph was plotted after averaging out the values of robustness of a large no of simulations.

While studying the behavior of the plot of robustness vs. load for all three traffic rates, we conclude that

- There is a sharp decrease in value of robustness from A to B with small increase in load (for heavy traffic).
- The decrease in robustness from A to B is relatively slow for moderate traffic is relatively slow as compared to heavy traffic. And the value of robustness is also slightly high.
- A far slow rate in decrease of robustness is observed from A to B for low traffic case.
- This variation in rates of robustness change is due to the traffic rates. Thus, robustness depends also on the load mapped already to the system. It doesn't depend on estimated makespan of a task but on the actual makespan of the task.
- In spite of all this, the robustness value saturates at around **1.5** for all the cases, thus proving that the method is consistent and gives a sub-optimal solution for robust allocation of resources on Homogeneous Distributed System.

Chapter 4 Conclusion and Future Work

4.1 Conclusion

The Robust Resource Allocation problem can be solved using various approaches, among them, FCFS algorithm approach is suitable for sub-optimal solutions and this work achieves the sub-optimal solutions. In this thesis, the Robust Resource Allocation problem is implemented using FCFS algorithm approach.

In this thesis, we have implemented basic FCFS algorithm. The performance of the algorithm is studied at different system load and traffic at the gates. For all the cases, the robustness of the system saturates at around 1.5. The applied algorithm works consistently for the model used under varying conditions and provides sub-optimal solutions.

4.2 Future Work

In this thesis report, the Robust Resource Allocation problem in Homogeneous Distributed System is solved using FCFS algorithm. The algorithm is proved to be successful in terms of serving the first arrived process first and thus reducing starvation. The problem is analyzed and solved by considering the parameters: Current load on the system, Load to be mapped, number of tasks to be mapped, the time-constraint.

In future, the performance of other algorithms (like round robin, min-max etc) can be evaluated under various parameters. The performance of these algorithms can be compared with the traditional algorithms such as FCFS algorithms. The algorithm has to be verified for different real-time traffic trends. Also, faults have not be considered while implementation of the

problem. The simulation can be carried out by taking into consideration, occurrence of number of faults like crash fault and the performance of the system can be observed.

The model used is a homogeneous one; the algorithm can be modified and applied to a heterogeneous model, thus providing more real-time solutions. Various other algorithms can be developed using different approaches like round-robin, min-max, etc. (as already mentioned above), so that a more optimal output may result.

.

References

[1] Shoukat Ali, Anthony A. Maciejewski, Howard Jay Siegel, and Jong-Kook Kim, Definition Of Robustness Metric For Resource Allocation, IEEE Trans. Parallel Distributed Systems 15(7), pp.630-641, July 2004.

[2] Prasanna Sugavanama, H.J. Siegel, Anthony A. Maciejewski, Mohana Oltikar, Ashish Mehta, Ron Pichel, Aaron Horiuchi, Vladimir Shestak, Mohammad Al-Otaibi, Yogish Krishnamurthy, Syed Ali, Junxing Zhang, MahirAydin, Panho Lee, Kumara Guru, Michael Raskey, Alan Pippin , Robust static allocation of resources for independent tasks under makespan and dollar cost constraints, Journal of Parallel and Distributed Computing,Vol.67, pp. 400-416, 2007.

[3] Aser Avinash Ekka, Fault Tolerant Real Time Dynamic Scheduling Algorithm For Heterogeneous Distributed System, Mtech(Research) Dissirtation, Department of CSE, NIT Rourkela, 2007.

[4] Zhiao Shi, Emmanuel Jeannot, Jack j. Dongarra, Robust Task Scheduling in Non-deterministic Heterogeneous Computing Systems, Cluster Computing, 2006 IEEE International Conference, 2006, pp.1-10, DOI: 10.1109/CLUSTR.2006.311868.

[5] Vladimir Shestak, Jay Smith, Robert Umland, Jennifer Hale, Patrick Moranville, Anthony A. Maciejewski, and Howard Jay Siegel, Greedy Approches to Static Stochastic Robust Resource Allocation For Periodic Sensor Driven Distributed Systems, *2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06)*, pp. 3-9, Las Vegas, NV, 2006.

[6] Werner Vogels, Chris Re, Robbert Van Renesse, Ken Birman, A Collaborative Infrastructure for Scalable and Robust News Delivery, Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops(ICDCSW'02) , pp. 655-659, 2002.

[7] Mukesh Singhal, Niranjan G. Shivaratri, Advanced Concepts in Operating Systems, Tata McGraw Hill Publication 2007

[8] George Coulouris, Jean Dollimore, Tim Kindberg, Distributed Systems, Pearson, 4th edition.