Matchmaking on Distributed Systems with Limited Knowledge of Resources

By

Jose Orlando Melendez, B.Eng. (SCE)

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirement for the degree of

Master of Applied Science in Electrical Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering

Department of Systems and Computer Engineering

Faculty of Engineering

Carleton University

Ottawa, Ontario, Canada

August 2010

Your file Votre référence
ISBN: 978-0-494-71518-5
Our file Notre référence
ISBN: 978-0-494-71518-5

Abstract

The problem of matchmaking and assigning requests to resources is an important problem in the context of resource management on distributed systems such as grids and clouds. Most existing research on matchmaking strategies uses *a priori* knowledge of the local scheduling policy used at a resource in the cloud/grid. However, exact knowledge of the scheduling policy used at every resource is unlikely to be available on real heterogeneous systems deploying a variety of different resources. This research presents matchmaking strategies that do not use detailed knowledge of the scheduling policies used at resources. The research shows how an improvement in system performance as well as revenue earned by the service provider can be attained through the deployment of these strategies. A simulation based performance analysis is performed. A thorough analysis of the simulation results leading to important insights into system behaviour and performance is presented.

Acknowledgements

Firstly, I would like to thank my supervisor Professor Shikharesh Majumdar for his guidance, continuous support, and encouragement during the thesis. He has helped me in developing skills required to solve problems which I will find uses for in my future endeavours.

I would like to thank Navdeep who helped me get started with the GridSim toolkit.

Finally I would like to thank my friends and family who gave me encouragement and support that was indispensible.

Table of Contents

List of Tables

List of Figures

List of Symbols and Acronyms

λ	Arrival Rate
A	Rate of Accepted Requests
AppLeS	Application Level Scheduling
AR	Advance Reservation
AS	Any Schedulability
B	Blocking Ratio
BPEL	Business Process Execution Language
EDF	Earliest Deadline First
ESF	Earliest Start time First
FF	First Fit
GASS	Grid Access to Secondary Storage
GHS	Grid Harvest Service
GIS	Grid Information Service
GRAM	Grid Resource Allocation Manager
GrADS	Grid Application Development Software
GridARS	Grid Advance Reservation-based System
GSI	Grid Security Infrastructure
GUR	Generic Universal Remote

JVM	Java Virtual Machine
LF	Least Fit
LHC	Large Hadron Collider
MF	Most Fit
MLF	Minimum Laxity First
NF	Next Fit
NWS	Network Weather Service
OD	On Demand
QoS	Quality of Service
R	Revenue Rate
RMS	Resource Management Systems
SLA	Service Level Agreements
UMR	Uniform Multi-Round
VM	Virtual Machine
VO	Virtual Organization

Chapter 1: Introduction

1.1 Background

The grid is the evolution of distributed systems and is rapidly growing in popularity [FOS-99]. The grid arose from the need to share computers, software, data, and various other types of resources that may be required by collaborating institutions in order to solve science and engineering problems [FOS2-01]. One such recent example is the Large Hadron Collider (LHC) grid [CER]. This is a grid that is used by the particle physics community in order to perform computations on the vast amount of data that are produced by the experiments. This grid is spread across the entire globe and uses resources from a large number of different scientific institutions from around the world. A grid can be seen as a collection of distributed networks. However a distributed network alone could not manage the requirements of the LHC grid as it is missing some features that are required by grids [CER].

The sharing of resources is controlled by a "Virtual Organization" (VO). The VO is a collection of resources and/or institutions that have the same set of rules for access to resources. Rules can consist of which resources can be shared, who is allowed to share and the conditions under which the sharing is permitted [FOS2-01]. General distributed systems do not address these requirements. The grid is designed to meet these requirements and is therefore expected to perform the following [FOS-02]:

1. *Coordinate resources that are not subject to centralized control*
2. *Use standard, open, general-purpose protocols and interfaces*
3. *Deliver nontrivial qualities of service*

Within a grid, resources are distributed in nature where there can be several domains, therefore non-centralized coordination is required in order to address issues of security, policy, payment, membership, etc. that arise in these settings [FOS-02]. The protocols and interfaces are required to be standard and open to ensure that various types of different resources and domains are capable for being used in a grid. Otherwise application specific grids will need to be developed [FOS-02].

The goal of grids is to allow access to resources in much the same way one can connect an electronic device into a socket to receive power. In fact the name grid is derived from the power grid system, as the infrastructure of the grid is similar to that of the power grid [FOS-99]. There are inter-connected nodes through which the resource power is shared and distributed.

The grid was first used mostly by scientific institutions interested in performing research or experiments on voluminous amount of data. However, in recent times a more business approach to the grid has seen a rise in interest on a phenomenon known as cloud computing [BUY-09]. Enthusiasts of cloud computing envision computing to become as much as a utility as water or electricity. This vision is similar to that of the grid. However there are some distinctions that make cloud computing an extension to the grid. As discussed in [BUY-09] the authors define the cloud as an extension of the combination of both clusters and grids, containing next-generation data centers with "*virtualized*" nodes

that are dynamically "*provisioned*" on demand as a personalized resource collection to meet a specific service-level agreement. The service-level agreements are established through a "*negotiation*" and are accessible via web services. Therefore it can be seen that grids have and will continue to garner attention and interest from both academic and business institutions.

1.2 Types of Grids

This section discusses a range of grid types.

1.2.1 Resources

Grids can be categorized by the type of resource that is shared. It is possible, however, for a grid to belong to more than one category when multiple types of resources are shared within the same grid. The most basic forms of grids are those that contain computing, data, and storage resources [ITT]. Computing resources share the computing power of computers; one such example is the TeraGrid which has a combined total of a petaflop of computing capability [TER]. Data grids share data resources; as mentioned earlier the LHC experiments produce a large amount of data that are to be shared among institutions. Storage grids allow users to access a large amount of storage space through sharing of storage devices. A well known storage grid is the Amazon S3 [AMA2]. Shared resources can also include other types of equipment such as lightpaths in lambda grids as discussed in [FAR-05].

1.2.2 Distribution

Another way of classifying a grid is through the geographical distribution of the corresponding resources [ITT] such as local or in a Virtual Organization. Internet distribution specifies a grid that contains resources that are connected through the internet. As such any computer or device with an internet connection can be part of the grid at any point in time. A Virtual Organization is a grid that contains within it several institutions or corporations that are seen as one large distributed organization. A local grid is such that it is contained within a single organization or corporation and can only be accessed by entities that are local or within the organization.

1.2.3 Service

Although grids are generally service independent, there are grids that can offer a specific type of service. For example a render farm, used in rendering 3D animations, can offer it's rendering service while using specific rendering software package [ITT].

1.3 Workload of Grids

Grids are subjected to tasks or requests. This section discusses at the types of requests that are typically seen by a grid.

1.3.1 On Demand

An On Demand (OD) request is the most basic form of request that is received by a grid. When an On Demand request is received the grid performs the request on a best effort basis meaning that there are no guarantees regarding the response time.

1.3.2 Advance Reservation

Advance Reservation (AR) requests are characterized by a deadline that must be met. ARs are also associated with an earliest start time, latest end time (deadline), and execution time. ARs are more beneficial as part of a system that is constrained by QoS guarantees [FOS2-99]. The deadline is used to achieve a QoS guarantee. The ability to reserve a window of computation time is useful when co-allocation of resources is required in order to perform requests [FOS2-99].

1.4 Grid Components

There is no one grid solution or basic model for a grid, instead a grid consists of a mix of several of the following components in order to perform its functions. These are the most commonly used core components.

1.4.1 Scheduling Component

Schedulers perform scheduling as described in Section 1.4.3.2 for Advance Reservations that may require co-allocation of resources [JAC-03]. There can be several levels of schedulers within a grid system. In a grid system where clusters of resources are present it is possible to have schedulers for each cluster as well as a high-level (meta-scheduler) that interacts with the schedulers for each cluster. The scheduler component is not always a necessary component partly because different types of schedulers are needed for different applications. The Globus toolkit, an open source software toolkit used for developing grid systems and applications, is such an example where there is no scheduler

component present. The AppLeS (Application Level Scheduling) parameter sweep template is an example of a commercially available scheduler component [CAS-00].

1.4.2 Data Virtualization and Data Management

A request is often associated with some type of data as any computational work requires some form of input(s) in order to perform the request. Since a grid system is geographically distributed it is not uncommon for data and computing power to be distributed as well. Therefore a component is required for seamless access of distributed data and the performing of data management [JAC-03]. This component allows users to access distributed data easily and efficiently as though the data is contained at a central server. Within the Globus toolkit this component is known as Grid Access to Secondary Storage (GASS) [BER-09, FOS-01].

1.4.3 Resource Management

A core functionality of a grid system is resource management, since it is composed of distributed resources which can dynamically join or leave the grid [JAC-03]. Resource management allows the resource providers to monitor, manage, and understand what is occurring within their system at a resource level. Within the Globus toolkit the Grid Resource Allocation Manager (GRAM) component performs these services [BER-09, FOS-01].

1.4.3.1 Matchmaking

This component, which is often called a broker, handles matchmaking. Matchmaking is the process by which requests are matched to available resources [JAC-03, BER-09]. Matchmaking may use different criteria. However, the most common criteria for matchmaking concern performance and efficiency. For example, incoming requests may be matched to the resource which results in the shortest turnaround time when performance is the focus. In this research matchmaking is made such that the number of requests that are rejected by the system is minimized.

1.4.3.2 Scheduling

Scheduling concerns the order in which requests are to be executed on a resource. The ordering of requests has an impact on the number of requests that can be scheduled at a given time. This notion becomes important within a grid when there are QoS constraints to meet. There are two types of scheduling; first there is task scheduling where individual requests or tasks are submitted to the grid and secondly there is batch scheduling where a set of tasks are submitted to the grid [JAC-03]. Scheduling may also be performed at different levels [BER-09].

1.4.4 Security

A grid provides access to a wide variety of resources some of which may contain sensitive information; hence it is important that information security not be compromised by the grid system [JAC-03]. A security component is used to ensure that the system

maintains an appropriate level of security for all users. This is handled by the Grid Security Infrastructure (GSI) within the Globus toolkit [BER-09, FOS-01].

1.4.5 Charge-back

Charge-back is used for businesses to track users' access of computing and data resources in order to charge users for using the grid system assets. Charge-back is commonly seen in utility grids and commercially available cloud systems such as the Amazon S3 that includes a data storage service [AMA2], Amazon's EC2 provides users with computational resources [AMA] and Google's App Engine service [GOO] provides web application developers with computing resources for running their applications.

1.5 Motivation for the Thesis

As mentioned earlier grids (and also clouds) comprise several distributed resources. These resources are heterogeneous in nature which means that they can contain both different types of hardware as well as running different types of software or operating systems.

Both matchmaking and scheduling are important in attaining QoS on such systems. Most existing matchmaking strategies are based on *a priori* knowledge of the scheduling policies used at the resource in the grid or cloud. However, it may be hard for a matchmaking broker to be aware of the scheduling policies used at all the resources in a large heterogeneous system where resources are added and removed dynamically.

The scheduling policy used at the resources may be unknown or difficult to use during matchmaking for several reasons. The exact configuration of the grid or cloud

may not be known at the time of system design or deployment and is likely to change many times during the lifetime of the system. Moreover, even if the local scheduling policy is known, executing the scheduling algorithm at the broker to determine the schedulability of an AR request on a given resource is likely to be a complex operation that requires a significant programming effort at the broker level and will substantially increase the run time CPU overhead for this broker. The problem gets magnified in larger heterogeneous systems deploying resources with diverse scheduling policies. Resources where the scheduling policy is unknown may also be incorporated into the system for various reasons: (i) including legacy resources that may be readily available; (ii) using newly acquired resources for which the scheduling simulator may take a considerable time to develop; and (iii) utilizing a set of resources acquired temporarily from another organization such as a specific department in an enterprise cloud or for dynamically handling of spikes in demands for computing resources (as described in [KIM-09]).

An example system is shown in Figure 1-1. This figure contains the following components: a source (representing a stream of requests from multiple users) which submits requests, a broker that performs matchmaking and matches a request to a resource, and a pool of resources which are capable of performing the request. In this example it is the broker's duty to perform matchmaking in such a way that the required level of QoS is met. This may require a QoS for each request in the form of a deadline. This type of QoS requirement is mostly seen in a system where AR requests are present; a best effort delivery is performed in a system comprising only On Demand requests. This thesis concerns matchmaking strategies that can be deployed at the broker. All the

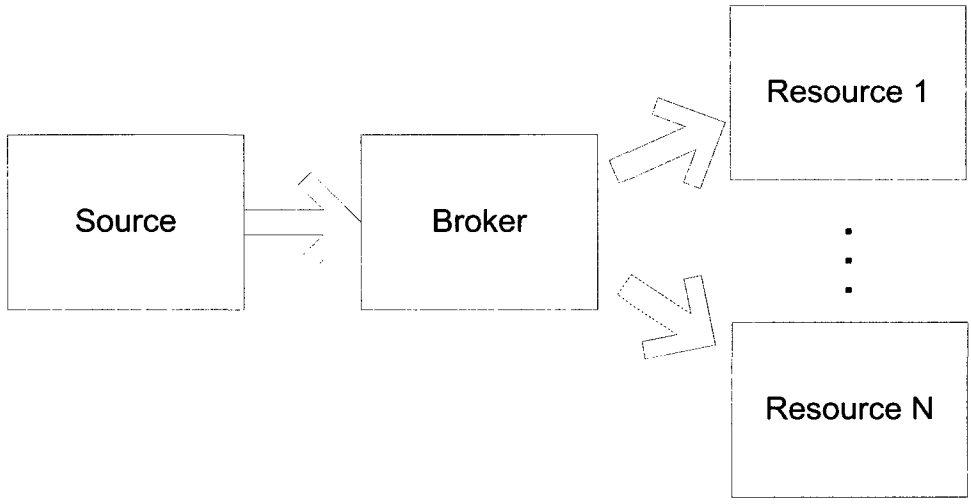

Figure 1-1: Structural view of a grid system

details of all the resources in the system may not always be known to the broker [MAJ-09]. Because of the difficulty associated with having exact *a priori* knowledge of the scheduling policies used at every resource this research focuses on matchmaking without using an exact *a priori* knowledge of the scheduling policies used at the resources.

1.6 Goals of the Thesis

The goal of this thesis is to devise a matchmaking technique that is capable of determining schedulability of an Advanced Reservation request and effectively allocate the request to an appropriate resource without using detailed *a priori* knowledge of the scheduling policies of the resources in the grid or cloud.

The matchmaking strategy devised in this thesis is compared with a traditional matchmaking strategy that uses *a priori* knowledge of the scheduling policies used at the resources. Performance of the matchmaking strategies is evaluated using several metrics

that are measured through system simulation. Various workload and system parameters are varied during the simulation. An important goal of the thesis is to perform an in depth analysis of the performance of the matchmaking strategies and determine the impact of these workload and system parameters on performance.

Systems deploying brokers, one using the matchmaking strategy devised in this thesis and the other using a traditional matchmaking strategy are simulated. Hybrid matchmaking strategies that combine the matchmaking strategy that does not use *a priori* knowledge of resource scheduling policies with traditional matchmaking strategies that are based on the resource scheduling policies are also investigated. One of the goals of the performance analysis is to determine which hybrid strategy performs the best and to understand what improvements in performance can be attained using a system comprising multiple brokers.

1.7 Contributions of the Thesis

The main contributions of the thesis include:

- A matchmaking technique that does not use detail *a priori* knowledge of the underlying scheduling policies at the available resources.
- Insights into the effects of various system and workload parameters used in the experiments/simulations on system performance and revenue earned by the service provider.
- Investigation of allocation algorithms that are used by the matchmaking strategy devised in this thesis.

- Demonstrating the effectiveness of hybrid matchmaking strategies that combine the matchmaking strategy that does not use *a priori* knowledge of resource scheduling policies with traditional matchmaking strategies that are based on the knowledge of resource scheduling policies.

Some of the research results are captured in a paper published recently [MEL-10]. Although most of the discussions in the thesis refer to grids, the matchmaking techniques are applicable to other types of distributed systems such clouds.

1.8 Outline of the Thesis

The remainder of the thesis is outlined as follows. Chapter 2 gives an overview of the concepts and previous work done in matchmaking and scheduling on grids and clouds. Chapter 3 describes the matchmaking strategies devised in this thesis and the simulation model developed for performance analysis. Chapter 4 examines the performance results obtained through simulation of the various strategies. Chapter 5 concludes the thesis as well as provides directions for future research.

Chapter 2: Overview of Matchmaking and Scheduling on Grids and Clouds

This chapter discusses the existing work on matchmaking and scheduling in the context of grids and clouds. First the workload parameters of the requests found within a grid are introduced in Section 2.1. Section 2.2 discusses various scheduling strategies for a grid environment. Section 2.3 provides a literature survey of methods used in grids to handle Advance Reservation requests whereas Section 2.4 provides a literature survey of resource management techniques used within cloud computing. Section 2.5 presents the Any Schedulability criterion that is used in this research.

2.1 Workload Parameters

This section describes the various workload characteristics that are used to define an AR request. For an OD request the service time is usually sufficient to define the request.

Each AR request has an end time which is the request's deadline. The work that is to be performed by the request must be completed before its end time. The earliest start time of the request denotes the earliest time at which the request is capable of executing. This characteristic is useful in situations where data is required by the request but is only available at some future point in time. The service time associated with the request specifies the amount of time required by the request to perform its task. AR requests are

The heterogeneity of resources also plays an important role in scheduling. Heterogeneity in resources may result from resources using different types of operating systems with their own scheduling policies used at the different resources.

The changing availability of resources also poses its own set of problems. Since grids are expected to be dynamic systems, resource availability can vary with time. This adds another layer of complexity to scheduling on grids.

The following sections present some of the existing techniques that are proposed for grids.

2.2.1 Static Task Scheduling

A scheduler that is said to be static is capable of computing the execution schedule before run time for a certain set of tasks [LIU]. To do this, resource information and performance parameters are to be known. Static task scheduling can be separated in to two categories depending on the divisibility of a task. These categories are divisible workload scheduling and fixed-sized independent task scheduling which are described in Section 2.2.1.1 and Section 2.2.1.2 respectively.

2.2.1.1 Divisible Workload Scheduling

The goal of divisible workload scheduling is to determine the number of divisions that the workload will be divided into such that the application's completion time is minimized [LOC-05]. For the divisible workload scheduling problem there are N worker processes that are controlled by one master process. Each process resides on a separate resource. The total workload, $W_{total,}$ can be divided into arbitrary chunks and are delivered

to appropriate workers by the master process. The master process is responsible for scheduling the chunks of the workload to the workers which is associated with network latency. The problem that is faced by this method is to determine the proportion in which the workload should be divided such that the overall execution time of the workload is minimized.

The authors of [LOC-05] present a solution to the divisible workload scheduling problem in grid environments with non-dedicated resources. The authors note that most solutions to the divisible workload scheduling problem assume that the computation resources are dedicated such that only grid applications are executed on the resource. Allowing resources to be solely dedicated for grid applications does not however, take into account the dynamic nature of grids. A different approach is taken by the authors whereby they assume that resources can contain both local applications as well as grid applications.

The authors propose to solve the problem by first predicting the performance of the workers and then scheduling the workload using the Uniform Multi-Round (UMR) algorithm. Each worker has a computational speed that is based on the available resource work capacity. This leads to a computational speed that is varied over time since it is a non-dedicated resource and must therefore be predicted using historical work capacity values. The performance of the workers is predicted by calculating the expected computation time of a chunk on a resource. Since local applications have a higher priority than grid applications the expected computation time must also take into account delays brought about by local application requests. A predictive factor is used in determining the

expected computation time in order to account for the varied work capacity that is allotted to grid applications.

The predictive factor is determined by first taking periodical measurements of the work capacity of the resource during each round and taking the average. Once several rounds have finished and several predictive factors have been determined, the predictive factor for the next round is predicted using the Mixed Tendency-Based strategy. The Mixed Tendency-Based strategy predicts expected computation times by observing whether the previous expected computation times have been increasing or decreasing. If the values are increasing the strategy predicts an increase for the adaptive factor in the next round and a decrease if the values are decreasing. Based on this estimation of the computation time of each worker the workload is divided among the processes and scheduled using UMR such that the execution time of the total workload is minimized.

The authors in [BER-96] develop an adaptive divisible workload scheduling system that adapts to changes in the resource pool. This is important since grids may have a varied number of resources over time. The system developed contains a *Scheduler* and a *Predictor* component. The scheduler component consists of two sub components, *Instance* and *Execution Manager.* The Instance component is responsible for dividing the workload and mapping the divisions to the resources. The workload division is done with the aid of the *Predictor* component to estimate the communication and computation time associated with the given resource set. The *Execution Manager* is responsible for executing the request schedule resolved by the *Instance* component. During the workload

execution the *Instance* listens for any event that has an effect on the workload such as resources joining, leaving, and performance variances.

The *Predictor* contains an *Estimator*, *Performance History Repository*, and a *Resource Monitor*. The *Estimator* dynamically estimates the computation times of resources as well as latency times for communications. The *Performance History Repository* keeps track of task and resource information. The *Monitor* monitors changes in grid resources and notifies the *Estimator* and *Scheduler* when changes occur.

The Adaptive UMR algorithm performs two main tasks: initial scheduling and adaptive rescheduling. The initial scheduling phase is the same as the UMR scheduling. The adaptive rescheduling phase occurs when there is a change in resources or a change in resource performance. The *Predictor* notifies both the *Estimator* and *Scheduler* of the changes. In the event that a resource joins the resource pool the *Scheduler* determines a new request schedule. If the new schedule is capable of completing all the requests sooner than the previous schedule the requests are rescheduled. Similarly if a resource leaves, or a performance change is detected, a new schedule is determined and the requests are rescheduled.

2.2.1.2 Matchmaking of Independent Tasks

The research presented in [LIU, CAS-00] is described by the authors as independent task scheduling. However, to keep the definition of scheduling consistent throughout this thesis this work is presented as independent task matchmaking. Scheduling and matchmaking are sometimes used interchangeably in the literature.

Independent task matchmaking focuses on minimizing the response time of a set of tasks by finding an effective mapping of tasks to resources [LIU]. The total execution time is estimated using *turnaround time* of each task that includes time for locating a resource as well as the expected time to compute. Because of the complexity of the grid, search heuristics are used to find a near-optimal solution. The general procedure for an algorithm that maps independent tasks involves calculating a metric for a task on all available resources and then choosing the resource that gives rise to the *best* value for the metric. This is done for all tasks of the set.

In [CAS-00] the authors use various heuristics along with a prediction model in order to ensure efficient deployment of parameter sweep applications of the grid. Parameter sweep applications are generally structured as a set of independent experiments each with their own set of parameters. This allows them to be modeled as independent sequential tasks. The proposed system iteratively assigns tasks to processors and computes the expected completion time of each task. This is merely an estimate. Based on the completion times of the tasks a metric is computed for each task. For example in the case of a min-min heuristic algorithm the metric is computed by determining on which resource a task completes the fastest and recording this completion time. The *best* metric is determined using a heuristic.

The general procedure for each task is: (i) compute the completion time on all resources (ii) using the completion time compute the metric for each task (iii) choose the resource which gives rise to the *best* metric for the task by performing the heuristic (iv) schedule the task on the resource. The metric for each task and *best* metric is defined by

the heuristic used such as in the *min-min* [FRE-98], *min-max* [FRE-98], and *suffrage* [CAS-00, MAH-99] heuristics. In the case of the min-min heuristic the metric is the minimum completion time of the task on all the resources. Among these metrics the metric with the smallest value is chosen. Therefore the min-min algorithm determines which mapping of task to resource results in the shortest execution time for the task. In the *min-max* algorithm, the metric for each task is the minimum completion time among all the resources. The *best* is defined as the metric which has the maximum completion time. The metric for the *suffrage* algorithm is the difference between the lowest and second lowest completion time of the task. The *best* metric is defined as the metric with the largest value. The idea behind *suffrage* is that a task should be mapped to a resource when that task would incur the greatest loss if it is not mapped to that resource because the other mapping options would lead to the request having a higher response time.

2.2.2 Application Level Scheduling

Application level scheduling (AppLeS) is based on the principles that (i) application and system specific information are needed for good schedules (ii) dynamic information is necessary to determine system state (iii) good schedules involve some prediction of application and system performance (iv) all resources can be evaluated based on the performance they deliver to the application [BER-96]. The approach used by this method is to use parameterizable application and system specific models to predict application performance using a given set of resources.

The authors of [MAN-05] use the principles of the AppLeS method scheduling with applications that use workflows to schedule them onto the grid. A workflow is

represented by a directed acyclic graph in which every vertex is a request and each edge represents a data dependency between requests [MAN-05]. The authors have developed techniques for scheduling and executing workflow applications on grid resources using the *Grid Application Development Software* (GrADS). The GrADS project has been developing tools to make the construction of applications for the grid simpler, while still maintaining a high level of performance. The GrADS infrastructure uses resource selection performance models which are extended to be used to schedule workflows. The authors have adopted a two stage approach. In the first stage, for each component the resources that the component can be assigned to are ranked and assigned specific rank values. These rank values represent the expected performance of a particular component on a certain resource. Next, the rank values for each component are used to build a performance matrix which can be used to map components to resources. The mapping problem is an NP-Complete problem that is solved by using heuristics which doesn't guarantee an optimal solution but return a solution in polynomial time. The heuristics used were that of the *min-min*, *max-min*, and *suffrage* which were explained in Section 2.2.1.2. After running the heuristics the mapping that produces the minimum completion time is chosen.

2.2.3 Resource Characteristic Prediction

As seen from Section 2.2.1.1 and Section 2.2.2 schedulers often require some form of a predictive resource performance model that is used for scheduling. This is due to grids being a dynamic resource-sharing computing environment with many different applications running simultaneously. Therefore run time resources such as processing

resources and available memory are variable [LIU]. However, sometimes a performance model is not available therefore empirical data analysis can be used to predict performance. In this type of a strategy previous execution time records for similar applications are used to compute an average execution time. In order to be effective these calculated average execution times must be as error free as possible. Hence the computed average execution time is held within a tolerable error. These computed averages are then used as the execution time for the application.

In [SUN-06] a performance model is developed for long term application performance prediction and task matchmaking. The resources within the system are capable of handling both local and grid tasks. The arrival rate and service time of local tasks are such that the system can be represented by an M/G/1 queuing system. A grid task is comprised of one single parallel phase and a final synchronization phase. The parallel phase consists of executing the task by splitting it into subtasks. Each subtask is assigned to a resource. The completion time for a subtask is computed as $T_k = \frac{w_k}{\tau_k} + Y_{k1} + Y_{k2} + \ldots + Y_{ks_k}$, where w_k is the workload of the subtask, τ_k is the speed of the resource k, and Y_k is the time for a local task to execute. The distribution along with the mean and variance for the subtask completion times are then determined. Since all subtasks are executing in parallel, the completion time of the whole task is the maximum of the completion times of all the executing subtasks. By evaluating the mean and coefficient of variation of the task completion times on a group of available machines, the task can be assigned to the most appropriate resources [SUN-06]. The selection criteria when using the mean and coefficient of variation is determined by the application

requirements. For example, in Grid Harvest Service (GHS) the machine with the smallest $E(T)(1 + Coe.(T))$ [SUN-06] (where Coe.(T) is the coefficient of variation of task completion time T) is chosen.

2.3 Advance Reservation Requests in a Grid

As mentioned in Section 1.3.2 one of the benefits of using an Advance Reservation request is a guarantee on quality of service that can be provided. This section presents some relevant research performed in the area of using AR requests to provide better end-to-end QoS to clients.

The authors of [MAR-07] explore the effects of Advance Reservation requests within a grid environment that also consists of On Demand requests. The performance metrics studied were resource utilization, queue wait times, and expansion factors which are considered paramount. Utilization is the proportion of the entire cluster of resources being used by the requests, queue wait time is the difference between the time a task starts running and the task arrival time, whereas the expansion factor is the ratio of the sum of requested wall-clock time and queue wait time divided by the requested wall-clock time. A simulator was created by the authors to investigate the impact of Advance Reservation requests on system performance. The simulator was injected with real task log data that was available from DataStar which is one of several resources offered by San Diego SuperComputer Center [SAN]. Three runs were performed in order to research the impact of Advance Reservation requests [MAR-07]: (1) a baseline run using only the real task workload, (2) a run containing both real task workload and Advance Reservation requests (3) a run containing real task workload and Advance Reservation

requests with policies enabled to control the number of reservations. The third run was performed since it was expected that a significant increase in Advance Reservation requests would impact the performance metrics negatively by increasing queue wait times and decreasing utilization.

The results showed that there was a drastic increase in queue wait times when Advance Reservation requests were introduced into the system. This was expected as tasks had to be scheduled around the Advance Reservation requests which led to delays and therefore higher queue times. However, the addition of the policies helped in reducing the large increase in queue wait times since it limited the number of reservations in the system. In one simulation run using a different set of real world tasks, the combination of Advance Reservation requests and policies was able to increase performance and resulted in a lower queue wait time than the baseline simulation. This case shows that some combinations of workload and reservations may result in increased efficiency.

In [ELM-08] the authors present a grid resource broker that supports Advance Reservation requests and performs resource selection based on criteria set by the user. The main objective of [ELM-08] is to minimize the completion time of the requests which includes the time required for submission of input files and executables to the resource, waiting in the batch queue, execution, and transferring the output files to the requested location. To determine the best resource for the submitted request, the user is required to select one or more bench marks with performance characteristics similar to those of the request. This information is used to predict the execution time of the task

since both the bench mark(s) and the submitted request have roughly the same computational requirements. Network performance is predicted using the Network Weather Service (NWS) which combines periodic bandwidth measurements with statistical methods to make short term predictions about the available bandwidth [WOL-99]. Hence the combination of the predicted completion time using bench marks with the predicted network performance is used to determine the resource mapping which leads to the minimum execution time of the request. The broker presented in [ELM-08] also has the ability to reallocate a reservation to another resource if it is determined at a later time that another resource can reduce the completion time.

2.3.1 Workflows with Advance Reservation Requests

Advance Reservation policies for workflows are investigated in [ZHA-06]. Workflows are known as applications in the context of grid computing and consist of inter-dependent tasks. These types of applications require that tasks be executed in a specific order. Executing workflows in a grid environment can be difficult because the traditional model of running on homogenous parallel machines that is controlled by a single local scheduler is not valid. The authors note that since resources queue and schedule requests differently there is no guarantee that tasks will run in parallel. The goal of the work is to develop efficient heuristics that can automate the process of creating reservation slots for scheduling the individual tasks of a workflow.

Each workflow is associated with a deadline by which all tasks within the workflow must be completed. The procedure used by the authors is to obtain an allocation for each task using workflow scheduling algorithms that can support heterogeneous resources and

minimize completion time. The allocation is performed using estimated execution times for the tasks and communication. If the completion time is greater than the deadline the allocation is rejected and the user is notified that the workflow cannot be scheduled. If the completion time is less than the deadline then the allocation is used. The heuristic takes the remaining time, the difference between the deadline and completion time, and distributes this to the individual tasks. This is done to ensure that each task has sufficient spare time of its own for completion. The spare time can be allocated to the tasks in various ways such as dividing it evenly among all the tasks, or dividing the spare time among the tasks such that the spare time for each task is proportional to its estimated execution time.

The work presented in [GUO-07] also concerns workflows with a focus on the QoS requirements associated with the workflows. The authors present a QoS-aware workflow management system. The QoS requirements are divided into two groups, loose (soft) guarantees and strict (hard) guarantees. Soft guarantees allow users to select service instances that can potentially provide a best-effort QoS profile meeting the client's requirements. These instances are based on previous measurements of that service on similar machines. This leads to estimations of the performance measures attained by the service. Strict guarantees provide a certainty on the delivery of the prescribed level of service. A popular approach to providing a strict guarantee is to reserve resources (memory, bandwidth, etc) through the use of Advance Reservation mechanisms. The management system described in [GUO-07] contains a Workflow Editor, a Workflow Planner, Observer, and a Performance Repository.

The workflow Editor is used to produce Business Process Execution Language (BPEL) workflows and QoS documents to be used by the system. The QoS documents specify the QoS needed by the user. The QoS documents along with the BPEL workflow are used by the Workflow Planner to reserve resources in order to schedule the tasks. The resources are chosen with the help of the Performance Repository. The Performance Repository contains the performance history of the resources. Observer is used to monitor the progress of the executing workflow. The progress is monitored in case the workflow does not execute as initially scheduled. If the execution differs from the schedule Observer calls the Workflow Planner to re-compute the schedule to ensure the QoS requirements are met.

2.3.2 Overestimation of Advance Reservation requests

In [PEN-09] the authors propose a relaxed Advance Reservation scheme that allows reservations to overlap with one another under certain conditions. In the paper the authors focus on Advance Reservation requests without laxity. Requests that are scheduled are given a definite time slot. By overlapping requests the reserved time slot is no longer reserved for a particular request but can be temporarily assigned to multiple requests. This reservation violation is allowed because the authors expect that the required service time for the request that is specified by the user is often over estimated. Hence the authors attempt to predict the over estimation of the request's service time specified by the user to get the true value of the service time which is smaller than the overall request service time. This allows for more requests to be scheduled. This strategy has the ability to decrease the request rejection rate when the request deadlines have been

overestimated. However, if the prediction method is not tuned properly this strategy has a risk of performing reservation violation whereby reservations are not capable of meeting request deadlines.

A similar approach is taken by the authors of [BIR-09] who attempt to resolve the problem of overestimation using overbooking (overlapping of requests) while taking into account the probability of resources failing. Requests are overlapped based on the probability of success which is calculated using the probability of a machine failure and the probability that the incoming request finishes on time. For each time gap found between scheduled requests where the incoming request can be scheduled the probability of success is calculated. The probability of success is used in the overbooking algorithm in various ways. An incoming request can be placed in the gap with the highest probability of success. Or a first fit approach can be used where the request is placed in the first gap with an acceptable probability of success.

2.3.3 Frameworks for Management of Advance Reservation requests

In [QU-07] the authors propose a generic framework that allows grid schedulers to manage AR requests across heterogeneous local Resource Management Systems (RMSs). At present only a few available RMSs such as Platform's LSF are capable of supporting AR requests [PLA]. In addition, current grid schedulers are only capable of handling AR requests within a homogeneous environment, where all local RMSs are of the same type and are AR capable. The framework enables the use of both AR-capable and non AR-capable RMSs within the environment. The core component of the framework is the AR manager which can perform the AR functionality for the local RMSs and can interact

with local systems [ELM-08]. An AR algorithm along with a list of AR requests for each local RMS is used to externalize the AR functionality from the local RMSs. Hence through the AR algorithm the AR manager is capable of scheduling requests to local RMSs for local scheduling or in the case of non-capable RMSs the AR manager schedules the AR request directly to a resource. In order to use the framework an API must be implemented at the local RMSs which allows for communication between the AR manager and the RMSs.

A Grid Advance Reservation-based System (GridARS) framework used for Advance Reservations-based co-allocation within grids is presented in [TAK-07]. GridARS has the ability to co-allocate distributed resources managed by various local schedulers by using a global resource scheduler in conjunction with the local resource schedulers. Several problems are addressed by the GridARS framework: (1) Co-allocation of various types of resources (2) Coordinating the global scheduler and the local schedulers to perform resource management (3) An Advance Reservation capability for local and global schedulers to co-allocate resources (4) An interface for submitting requests with secure communication (5) A two-phase commit protocol used to request and then commit reservations. These problems are solved by combining a global scheduler with resource managers for computers, network, and other resources. In each resource manager, a resource scheduler maintains a reservation table of its resources for Advance Reservation. When a user sends requirements on resources and reservation time to the global scheduler, the global scheduler uses the reservation table of the resource manager to coordinate resource reservation and perform request scheduling.

The work done in [YOS-05] takes a different approach in comparison to the method described in the previous paragraph. The goal is to have users set the reservations and avoid the traditional approach where a centralized scheduler controls the local schedulers at each of the resource sites. This is possible in systems where the local schedulers give users the ability to set reservations manually. This is known as user settable reservations. However, instead of having the user select resources manually and set reservations manually, the entire process is automated using the Generic Universal Remote (GUR) metascheduler system.

GUR is a metascheduler that creates coordinated reservations on local schedulers. This is done by having users submit requests along with a set of requirements to GUR. These requirements include the number of resources needed, request duration, earliest start time, a deadline, and the type of request. The type refers to whether the request runs at a single site or multiple sites.

In order to find appropriate resources, GUR probes each site to make a rough guess of each system's queue load [YOS-05]. The queue load is the number of reservations waiting to execute at that system. This is done by setting test reservations on each system and checking the delay for each test reservation. GUR then chooses the site with the least amount of load that can perform the request within the required time frame.

2.4 Cloud Computing

As reflected in the literature survey presented in this chapter, management of resources acquired by applications on demand has been studied extensively in the context of grids. More recently resource management on clouds has started receiving attention from

researchers. A cloud is a collection of virtualized computers that are dynamically provisioned for providing computational resources based on Service Level Agreements (SLA) to clients [BUY-09]. The currently available clouds include both storage and computing resources for clients that submit requests that are to be executed on resources to the system. Cloud service providers such as Amazon have already started providing cloud environments such as the "Elastic Computing Cloud" (Amazon EC2) [AMA] and "Simple Storage Service" (Amazon S3) [AMA2] for buying computing cycles and storage on demand. Cloud computing offers a number of advantages including elastic computing that enables a user to grow and shrink its resource pool based on demand, elimination of the cost of acquiring resources by users and the ability to pay per use [ARM-09]. A large part of the existing techniques for resource management on grids can be extended to clouds. Recent research on clouds has been adding to the exiting knowledge in the domain of resource management. A representative set of papers focusing on resource management on clouds is presented next.

The authors of [CAR-09] note that clouds have an improved scalability over grids yet grids offer a larger computational power. In order to gain the benefits of both systems, the authors have devised a solution that uses the Diet-Solve [CAR-06] middleware to manage cloud resources. Eucalyptus [NUR-08] a cloud management system is used within the Diet-Solve to act as an allocator and scheduler for resource management. A game-theoretic method for allocating tasks with QoS requirements to resources in a cloud is presented in [WEI-09]. The approach uses game theory to determine an effective allocation such that the multiple QoS constraints are met.

OVIS, a resource monitoring and management tool for cloud computing environments is presented in [BRA-09]. The goal of OVIS is to provide the ability to dynamically provide and respond to information about the cloud environment and application state of services that would enable more appropriate, efficient, and flexible use of the resources. Using feedback control theory, the authors of [LI-09] model an adaptive controller that dynamically adjusts the utilization of multiple virtualized resources to achieve application Service Level Agreements in cloud computing. The authors also present a Virtual Machine (VM) based architecture for adaptive management of virtualized resources.

Memory interference can play an important role in the management of resources. Clouds often comprise virtual machines, where a single resource shared by multiple users is visualized as a set of virtual machines. The authors of [RAJ-09] propose a solution to eliminate cache interference between two virtual machines that share the same cache. This is done using cache hierarchy aware core assignment where each core is allocated to one service.

Although resource management has been studied in the context of grids and clouds, little existing work has concerned matchmaking without using detailed *a priori* knowledge of the resource scheduling policies that this thesis focuses on. The matchmaking strategy investigated is based on an Any Schedulability criterion that is presented next.

2.5 Any Schedulability Criterion

The Any Schedulability (AS) criterion introduced in [MAJ-09] refers to the schedulability of the AR request set under any work conserving scheduling policy. This criterion allows one to determine whether a set of AR requests will meet their deadlines without use of any knowledge of the local scheduling policy used at the resource. Under a work-conserving scheduling policy the CPU is never kept idle when a request is available for execution. A request set is said to be schedulable if each request in the set can be completed before its deadline. For the Any Schedulability criterion to be satisfied the conditions out lined in the following theorem need to be fulfilled [MAJ-09]:

Any Schedulability Theorem:

A set of requests (i= 1..N) each of which is characterized by an earliest start time and deadline is any schedulable if the following conditions are satisfied for ***each*** *request i:*

$L_i \geq \sum_{j \neq i} X_1 . \ min\{Ej, (D_j - S_i)\} + \ X_2 . Ej$

Where,

$X_1 = 1$ *if* $(S_j < S_i, D_j < D_i, S_j < D_i, S_i < D_j)$,

OR if $(S_j \geq S_i, D_j < D_i, S_j < D_i, S_i < D_j)$,

OR if $(S_j \geq S_i, D_j \geq D_i, S_j < D_i, S_i < D_j)$;

$X_1 = 0$ *otherwise;*

$X_2 = 1$ *if* $(S_j < S_i, D_j \geq D_i, S_j < D_i, S_i < D_j)$;

$X_2 = 0$ *otherwise.*

The basic idea underlying the theorem is that, to be schedulable, the laxity of a given request must be greater than or equal to the sum of the worst case execution times for other requests present in the system that can overlap with the given request [MAJ-09].

A discussion of Any Schedulability-based matchmaking is presented in Chapter 3.

Chapter 3: Any Schedulability-based Matchmaking

This chapter focuses on Any Schedulability criterion based matchmaking. It describes the system architecture and the toolkit used in developing a simulator for the matchmaking strategy. It also details the components used within the simulation, defines the various allocation algorithms and scheduling policies. Hybrid matchmaking strategies that combine different brokers and resources are introduced in order to increase system performance. The chapter includes a description of the performance measures, simulation parameters, and test conditions.

3.1 Any Schedulability-Based Matchmaking

The purpose of the matchmaking strategy investigated in this research is to select resources for executing primarily Advance Reservation requests. Handling of On Demand requests is discussed in Section 3.4. This is done by determining which of the resources is capable of scheduling the request while using a certain matchmaking criterion. The Any Schedulability-based broker is a broker devised using the Any Schedulability criterion and it performs matchmaking without using detailed *a priori* knowledge of scheduling policies used at the resources. The benefits of using an AS-based broker in this type of system include the following:

- The ability to determine whether or not a request is schedulable on a resource without knowing the underlying scheduling policy which can happen very frequently in a real world environment (see discussion in Section 1.5)

- Simulating the scheduling policies of an operating system is time consuming. This can be avoided by implementing an AS-based broker that performs fewer computations and is less time consuming.

The AS-based broker performs its duties by requesting the characteristic of the resource that contains all the current requests allocated to the resource, and then using the AS criterion to determine whether the resource should be allocated the request. When multiple resources are capable of meeting the request deadline, a First Fit or Next Fit algorithm is used to allocate the request to a resource. These algorithms are described in the following subsections. The broker that uses knowledge of resource scheduling policies simulates the scheduling policy to determine if a request is schedulable at a resource. If the request is schedulable on multiple resources, the First Fit or the Next Fit algorithm can be used to select the resource on which the request is allocated.

3.1.1 First Fit Algorithm

With this algorithm, the broker chooses the first available resource from the resource list on which the Any Schedulability criterion is satisfied.

3.1.2 Next Fit Algorithm

The broker keeps track of which of the resources from the resource list was last assigned a request. Starting with the resource that appears directly after the resource in the resource list that was last assigned a request, the broker chooses the first available resource that meets the Any Schedulability criterion.

3.2 Matchmaking Based on Knowledge of Scheduling Policies

In order to evaluate the Any Schedulability-based broker two additional brokers based on several existing scheduling policies are simulated. These brokers use the knowledge of the scheduling policy used at a given resource to verify if the newly arriving request is capable of meeting its deadline when allocated on the resource. The resource then determines the scheduling of the request based on this scheduling policy. All resource schedulers ensure that the work conserving principle is maintained. Once again, when a request is schedulable on multiple resources the algorithms used by the broker can be used to allocate the request to a resource. If a request is rejected by all the resources it is deemed unschedulable and rejected.

The following sections explain the implementation of the scheduling policies that the broker uses to determine if a request is schedulable on a given resource. The scheduling policies described are non pre-emptive scheduling policies.

3.2.1 Earliest Deadline First Scheduling

Earliest Deadline First (EDF) scheduling uses the deadline of a request to determine its order of execution. The earlier the deadline, higher is its priority of execution. To keep

the scheduling policy work conserving the EDF scheduling policy is modified in this research. Upon arrival of a new request it is determined if the request is schedulable based on its deadline. If the request is not schedulable then the request is rejected. If it is schedulable then any earlier idle time slot for the resource that falls within the request's execution window is filled even if it places the request out of order. If no such idle time slot is available then the request is scheduled based on its deadline. When the order of a new request is being determined by using its deadline only those requests that are still ordered by their deadlines are used in determining the order of this newly arrived request. Requests that are out of order because of filling earlier resource idle time slots are not considered.

3.2.2 Earliest Start Time First Scheduling

Earliest Start time First (ESF) scheduling orders requests based on the earliest start time for the requests. The earlier the earliest start time, the sooner the request is executed. In this policy the work conserving principle is always maintained. When an incoming request *i* arrives at the resource it is scheduled with respect to the earliest start times of the requests in the reservation list. In order for the scheduling of request *i* to be accepted it must be able to meet its deadline and must not cause other requests to miss their deadlines. If these two requirements are met then request *i* is scheduled.

3.3 Hybrid Matchmaking Strategies

From a resource management perspective a distributed environment may include two types of resources: *transparent* and *opaque* (see Figure 3-1). The transparent resources

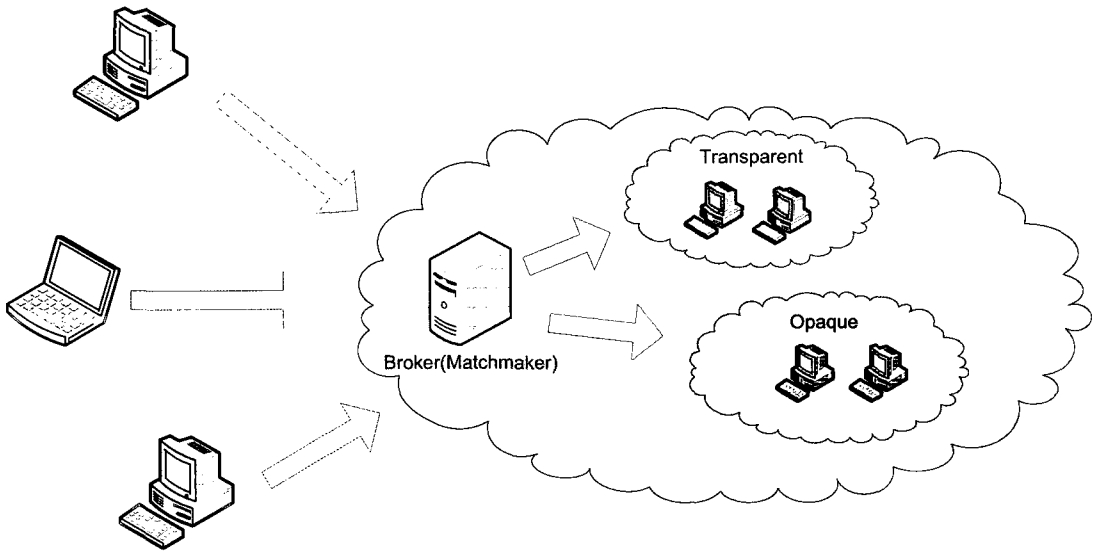

Figure 3-1: Distributed System with transparent and opaque resources

use local schedulers whose behaviours are known to the matchmaking broker whereas the local scheduling policy deployed at an opaque resource is unknown. Section 1.5 discusses the reasons why resources with unknown scheduling policies may be present in the system. In order to use both the *transparent* and *opaque* resources within the same grid or cloud system, the two types of resources are separated into pools. Each pool of resources is managed by a broker that is based on its resource type. Transparent resources are managed by a transparent broker and opaque resources by an opaque broker.

Two hybrid matchmaking strategies are investigated for handling such a mixed environment and are discussed in the following subsections.

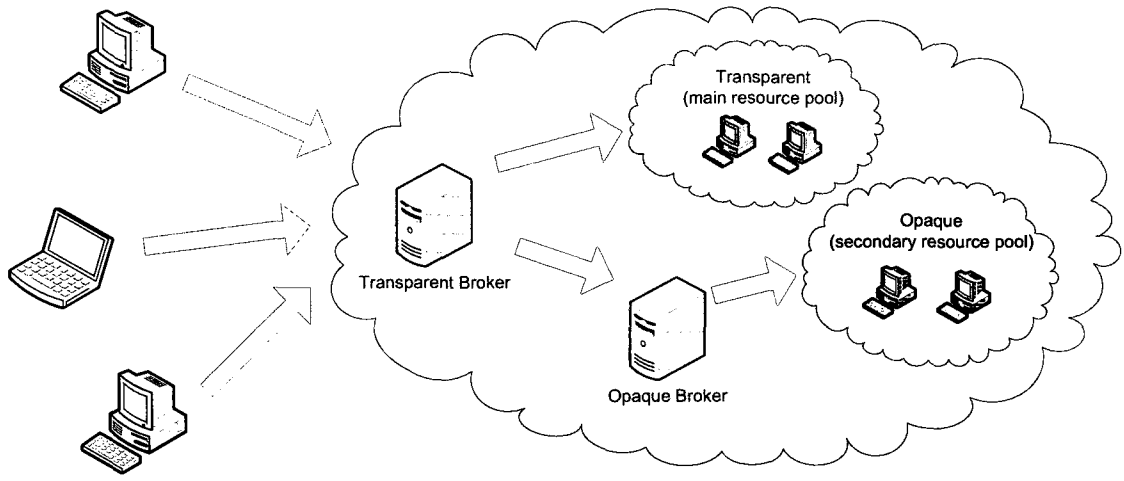

Figure 3-2: The Combined Strategy for handling incoming requests.

3.3.1 Combined Strategy

In the combined strategy the two resource pools are termed the main resource pool and the secondary resource pool (see Figure 3-2). The main resource pool consists of transparent resources and is managed by the transparent broker while the secondary pool consists of opaque resources and is managed by the opaque broker (AS-based broker). In this strategy a request, upon arrival is sent to the main resource pool where the transparent broker determines if it is schedulable on any of its resources. If schedulable, the transparent broker assigns the request to a resource. If the transparent broker determines that the request cannot be scheduled on any of its resources then the request is sent to the secondary resource pool where the opaque broker determines if it is schedulable on any of its resources. If neither broker can find a resource for the request then the request is rejected. It is expected that using such a strategy will help improving

performance in comparison to a system that does not include a secondary resource pool with a broker that uses the AS criterion and thus only comprises the main resource pool. The improvement accrues from the additional computing power that is offered by incorporating more (opaque) resources into the system.

3.3.2 Independent Strategy

In the Independent strategy each resource pool is given a certain fraction of the total workload (see Figure 3-3). A fraction, x, of the request arrivals per unit time are sent to the transparent broker while, a fraction $(1-x)$ of the request arrivals are sent to the opaque broker. For example, if there are requests arriving at the system at λ arrivals/sec then the transparent broker receives requests at a rate of $(x)(\lambda)$ arrivals/sec while the AS-based broker receives requests at a rate of $(1-x)(\lambda)$ arrivals/sec, where x is a fraction of the total workload. Note that the value of x depends on the request mix submitted to the system. It is the proportion of the total number of requests that are to be submitted to resources managed by the transparent broker.

3.4 Mixed Workload

The focus of the research performed in this thesis is a system that is subjected to Advance Reservation requests. Certain cloud and grid systems are subjected to both On Demand and Advance Reservations. Various experiments are presented that include On Demand requests in the workload. Generally an On Demand request is scheduled on a best effort basis and can be modelled as an Advance Reservation request with an arbitrarily large deadline. The On Demand requests are deemed to always be schedulable

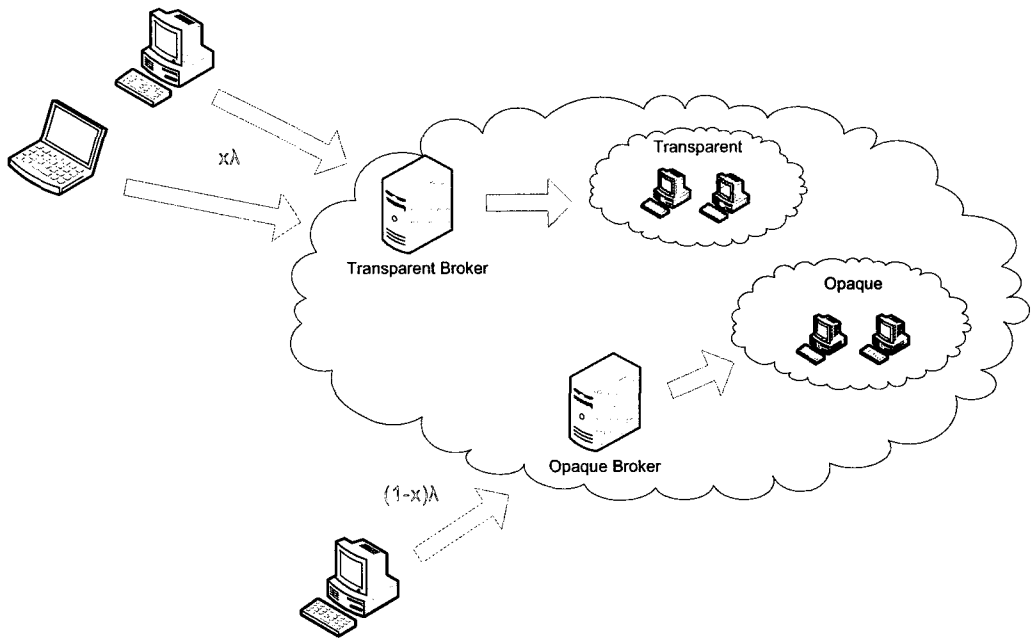

Figure 3-3: The Independent Strategy for handling incoming requests.

due to a lack of deadline; hence they are not sent to the broker and are sent directly to the resource for scheduling. The scheduler schedules the OD request based on the large deadline. This ensures that the OD requests remains schedulable irrespective of the scheduling policy used. When the Any Schedulability-based broker attempts to determine the schedulability of requests, where a request with a very large deadline is present, the newly arrived is request is often not schedulable. This happens because the request with a very large execution window overlaps with the newly arrived request violating the Any Schedulability criterion. A preliminary analysis showed that this in turn leads to a significant number of rejected requests.

Thus, the On Demand request's deadline is altered after scheduling and is set to the sum of its service time and a laxity. The laxity used in the simulation experiments is

discussed in Section 3.9.3.The earliest start time of the request is set to its current start time.

3.5 System Architecture

As mentioned in Section 1.1 a grid system is a collection of distributed resources with a traffic source submitting requests to the various resources via a broker. The broker allows the source to forego the need to directly choose among the resources. The broker itself determines which resource is best suited to perform the request based upon a QoS requirement. Resource discovery for the broker is performed by using a Grid Information Service (GIS) which is an entity that tracks which resources have registered with it. Once a resource is initiated it must find a GIS with which to register so that it may be discovered. Figure 3-4 shows the general architecture of the system used in the research.

3.6 GridSim Toolkit

Grid and cloud systems often contain a large number of resources. The GridSim Toolkit is a tool that allows users to build simulation models of such distributed systems with various components and simulate the environment. As described in [BUY-02] the toolkit provides a comprehensive facility for simulation of different classes of heterogeneous resources, users, applications, resource brokers, and schedulers.

The resources and brokers can be tuned to include certain types of operating systems, a certain number of processors, and the cost of using the resource. The toolkit comes with some standard predefined scheduling policies but allows users to define their own scheduling policies.

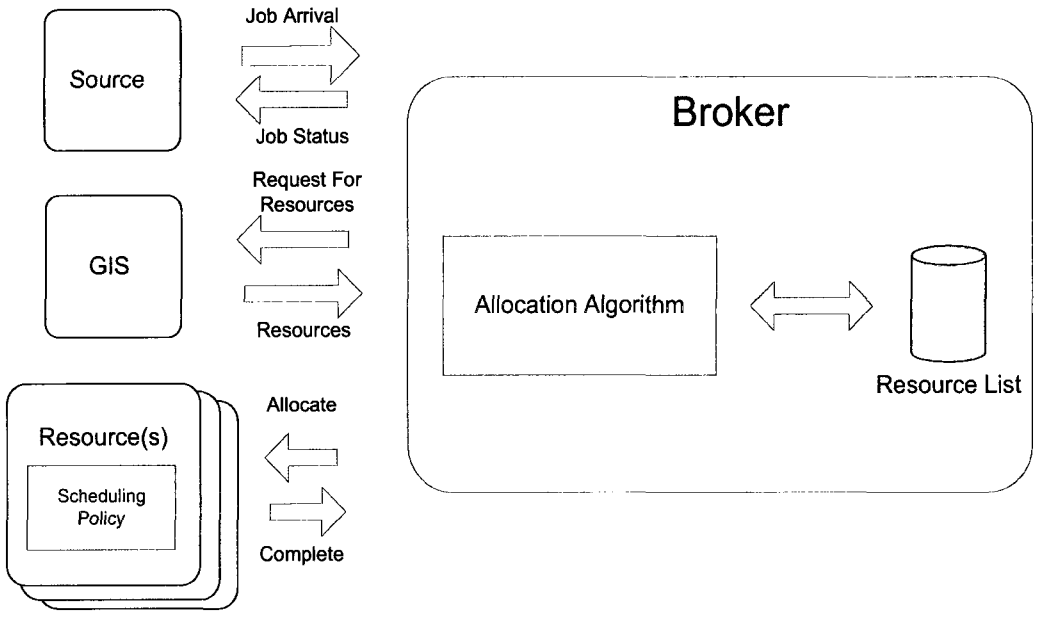

Figure 3-4: General Architecture of the System

3.6.1 GridSim Architecture

GridSim is built on top of existing software technologies (See figure 3-5) resulting in a multi-layered architecture [BUY-02]. The first layer comprises a scalable Java interface and runtime mechanism called the Java Virtual Machine (JVM). The second layer called SimJava focuses on a basic discrete-event infrastructure that is built using the interfaces supplied by the first layer. The third layer focuses on simulating core grid entities such as resources and information services. The fourth layer is concerned with the simulation of GridSim entities such as resource brokers as well as the resources themselves. The fifth and final layer is concerned with applications to be used in evaluation of resource management strategies, scheduling policies, and other types of algorithms.

3.7 Simulation Entities

This section describes how the various entities described in Section 3.5 and Section 1.4 are used and implemented using the GridSim Architecture described in 3.6.1.

3.7.1 Source

The source is the component that is used to generate user requests sent to the resource broker and collect statistical data. It is the starting point and end point of the simulation. The simulation ends once the source has finished submitting a certain number of requests.

3.7.2 Broker

The broker component is responsible for receiving requests from the source, obtaining a list of resources from the GIS as well as performing resource management. Once the broker receives a request it decides which available resource is the most suitable to

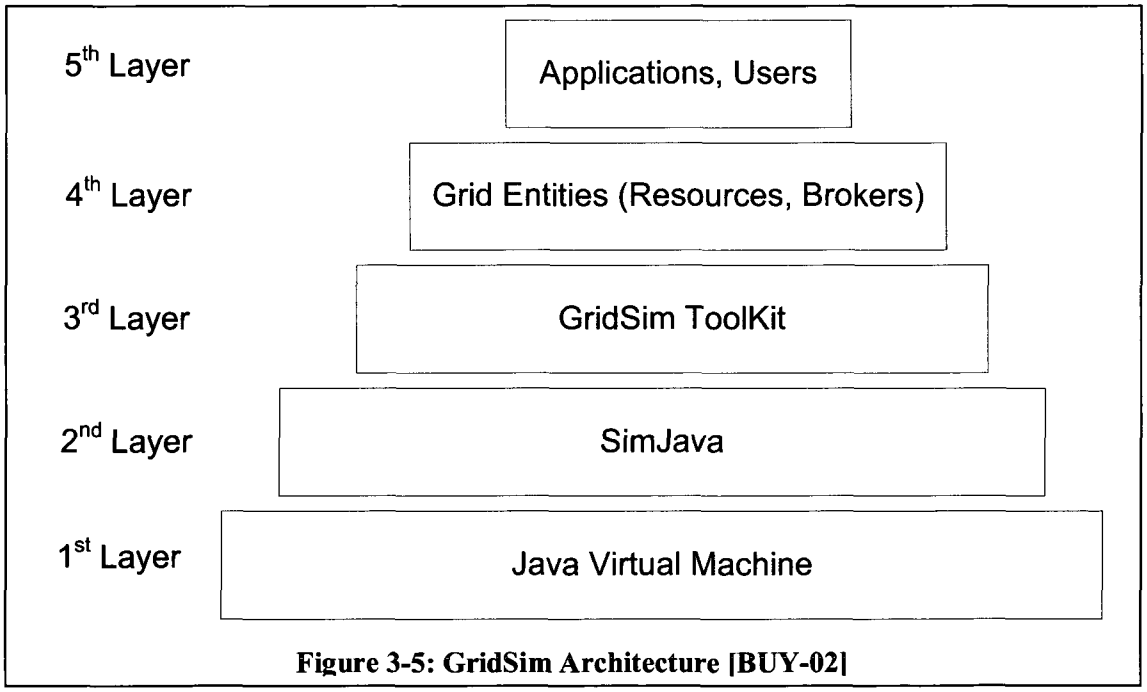

Figure 3-5: GridSim Architecture [BUY-02]

undertake the request by using the allocation algorithm.

3.7.3 Grid Information Service

The Grid Information Service is responsible for keeping track of resources in the system. This is done by having the resources register with the Grid Information Service when they become available for use within the system. The broker interacts with this component to determine the resources which are available for use. Multiple Grid Information Services can be distributed geographically in order to spread the resource management across several entities. This research focuses on schedulability of Advance Reservation requests and not on the geographic distribution of resources. Thus using a single GIS is appropriate for the research described in this thesis.

3.7.4 Resource

Resources are computing resources which simulate the computational requirements of requests submitted by users via the broker. Within the GridSim toolkit it is possible to create two types of resources, a general resource or an Advance Reservation resource. The general resource is able to handle On Demand requests only while the AR resource can handle both On Demand and Advance Reservation requests. Several sub entities are used to define the resource.

3.7.4.1 Resource Characteristics

Every resource contains a resource characteristic entity which is used by the broker to determine various features about the resource such as the operating system on the resource or the number of processors available for use at the resource. Within the system

simulated in this thesis the resource characteristic entity is also used to keep track of the requests scheduled on the resource. This is then used by the broker to determine if the resource is a suitable candidate for a newly arrived AR request. For the purposes of this research each resource is assumed to comprise a single CPU with other properties being set to default values provided by GridSim.

3.7.4.2 Scheduling Policy

Each resource is associated with a scheduling policy that is used to schedule incoming requests. In this research two existing Advance Reservation scheduling policies are implemented. These include Earliest Deadline First and Earliest Start Time First. They are described in Section 3.4.1 and Section

3.7.5 Requests

Requests are known as gridlets in the Gridsim toolkit but are referred to as requests throughout this thesis. A request entity contains information regarding the request such as its earliest start time, start time, end time and the other request parameters described in Section 2.1.

3.8 Performance Measures

The performance of the Any Schedulability-based broker is compared to that of a transparent broker by determining the Blocking Ratio, Rate of Accepted Requests and Revenue Rate of the system for various workload and system parameters. These performance metrics are described in the next subsections.

3.8.1 Blocking Ratio

The Blocking Ratio (B) of the system is the ratio of the number of requests that are rejected to the total number of requests that arrive on the system. Since each broker is responsible for determining schedulability this performance measure is computed by the broker. For the combined strategy the blocking ratio is the blocking ratio of the opaque broker (AS-based) broker, since it is the last broker in the chain that can potentially schedule the request. In the independent strategy the blocking ratio is weighted relative to the amount of workflow received at each broker. The blocking ratio in this case is:

$$B = (x)(B_{TransparentBroker}) + (1-x)(B_{OpaqueBroker})$$

In the equation above x is a fraction of the total workload.

3.8.2 Rate of Accepted Requests

The Rate of Accepted Requests (A) is the rate at which the system can accept requests into the system. This is determined as the product of the arrival rate of requests and the acceptance ratio. The acceptance ratio is the ratio of requests that are accepted to the total number of requests that are sent into the system and is given by (1-B). Thus,

$$A = (\lambda)(1-B)$$

In order to achieve a large value of A both the rate of accepted requests and λ must be high.

3.8.3 Revenue Rate

Revenue Rate is a metric for business performance. It is the rate at which revenue is earned by a service provider that owns the resources. R is measured in currency units (*cunits*) per second. For example, if a resource costs 1 cent per second then *cunit* is 1 cent

and the total revenue earned per second for the resources is \$0.01R when it is fully utilized. This is important in the context of clouds and grids that charge their client for resource usage. This metric is used to determine the amount of revenue that is earned by the system.

Revenue Rate is determined by adding the execution time of all the requests that were accepted on the resources and dividing the sum by the simulation time.

3.9 Parameters and Resource Management Strategies

This section describes the system and workload parameters of interest. The system parameters are described first followed by the workload parameters.

3.9.1 System Parameters

3.9.1.1 Number of Resources

This parameter refers to the number of resources in the pool of resources.

3.9.1.2 Resource and Broker Type

Resource type determines whether a resource is transparent or opaque. Broker type indicates whether the broker handles transparent or opaque resources.

3.9.1.3 Allocation Algorithm

As discussed in Section 3.1 this is the algorithm that the broker uses for allocating requests to resources. The allocation algorithms include First Fit and Next Fit.

3.9.1.4 Hybrid Matchmaking Strategy

As discussed in Section 3.3 two types of hybrid matchmaking strategies are used in a multi-broker based system: the combined strategy and the independent strategy.

3.9.1.5 Workload Type

This parameter specifies whether the system workload contains solely Advance Reservation requests or a mix of both Advance Reservation and On Demand requests. For the simulation results reported in this thesis a mixed workload consists of 20% OD requests and 80% AR requests.

3.9.2 Workload Parameters

3.9.2.1 Arrival Rate (λ)

The arrival rate is the rate at which requests arrive on the system. This is measured as the number of arrivals/sec and is referred to by the symbol λ. A Poisson arrival process is often used in studies of distributed systems (see [FAR-05] for example). In this thesis a Poisson arrival process is used to model the arrival of requests.

3.9.2.2 Service Time

The service time for a request is the amount of time the request requires at the resource to complete its task. The default service time used for this research is uniformly distributed and has a lower bound of 10 minutes and an upper bound of 90 minutes which gives a mean service time of 50 minutes. A similar distribution is used in [MAJ-09] and [SUL-04].

3.9.2.3 Earliest Start Time

The earliest start time is set as the sum of the arrival time, or creation time of the request, and a uniformly distributed random variable with a lower and upper bound of 0 and 12 hours respectively. A similar distribution is used in [SMI-00].

3.9.2.4 Laxity

The thesis defines laxity of an AR request as the percentage laxity multiplied by the mean service time of the requests. Laxity is calculated as, laxity = (mean service time)(percentage laxity). The percentage laxity is an input parameter and can assume any percentage value greater than zero. A percentage laxity of 200% gives rise to a laxity that is twice the mean service time.

3.9.2.5 Coefficient of Variation of Service Times

In addition to a uniform distribution, the service time is modelled using a hyper exponential distribution. This enabled the generation of larger variabilities in request execution times. With this distribution the coefficient of variation is varied to determine its effect on performance.

3.9.3 Workload Parameters for On Demand Requests

The service time is determined in the same way as discussed in Section 3.9.2.2. The start time of an On Demand request is set as the time at which the request enters the system. The laxity (which determines the end time) is set as the sum between 12 hours and (mean service time)*(100) in order to generate a very large window of execution for the request.

As discussed in Section 3.7 once an On Demand request is scheduled it is assigned a new earliest start time and given a laxity of 200% of the mean service time.

3.10 Verification of Simulation Experiments

Verification of the simulation is performed in several ways. The following details the methods used for verification.

3.10.1 Manual Verification

In terms of the allocation algorithms and scheduling policies developed and used within this research some verification is performed manually. This is done with the aid of the programming environment and its debugging functionality. During the course of a simulation, the simulation is paused at strategic points in order to determine the existing requests on a resource and the workload parameters of a newly arriving request. The ending result of the scenario is determined manually using the information attained from the simulation. The simulation is then allowed to continue and the manual result is compared to the simulation result. The ability of debugging also allows for step by step execution of the various policies and algorithms, where the interim results are recorded and again verified manually. Once both results agree with one another the policy or algorithm is deemed to perform as expected.

3.10.2 Little's Law

Little's law states that the number of requests within a system is equal to the throughput of the system multiplied by the mean response time of the system [LIT-61]. The mean response time of the system is determined by taking an average of the response time of

the requests that are accepted. The throughput of the system is the ratio of the number of requests that the system can handle and the period of simulation. Since the acceptance ratio is 1-B, the system throughput is $\lambda^*(1-B)$. The average number of requests in the system is calculated by dividing the total response time of all accepted requests in the system by the total time the system is under observation. The observation time in this case is the total simulation time.

This procedure is performed for multiple simulations. The measured value of throughput is found to be in close agreement with the calculated value by using Little's Law [LIT-61].

3.10.3 Accuracy

The accuracy of the simulations is determined by repeating the simulation runs a number of times and calculating the confidence interval. The Simulation was repeated sufficient number of times such that a confidence interval under ±5% of the mean value with a confidence level of 95% was achieved for the Blocking Ratio.

Chapter 4: Performance Results

This chapter describes the results of the various simulations conducted for determining system performance. In order to study the effect of a given parameter on performance, a factor at a time approach is used in the simulation experiments. This approach holds all parameters at their default values (see Table 1), while the parameter of interest is varied. In Table 1 U(10,90) refers to a uniform distribution used with a lower bound of 10 and an upper bound of 90.

Two types of systems are discussed in this chapter. The first type of system comprises only a single resource whereas there are multiple resources in the second type of system. Thus, in the first type of system the functionality of the broker reduces to performing admission control as it examines a request and then accepts it if it is schedulable on the resource and rejects it otherwise. The broker in the second type of system not only performs schedulability analysis but also selects an appropriate resource according to an allocation algorithm when the request is schedulable on multiple resources.

The following sections describe the effects of changing a given parameter on performance.

Table 1: Default Value of System and Workload Parameters

Parameter	Value
Number of Resources, Nr	1
Resource Scheduling Policy	Earliest Deadline First
Allocation Algorithm	First Fit
Service Time, S	U(10,90) minutes
Earliest Start Time, E	U(0,12) hrs
Laxity, L	200%
Workload Type, W	Advance Reservations only

4.1 Objectives

The objectives of this chapter are to outline the effects of using an AS-based broker and compare its performance to a transparent broker. A transparent broker is a broker that has *a priori* knowledge of the resource scheduling policy. Various system and workload parameters are varied in the simulation experiments to determine the effects they have on the performance measures B and A of the system. As indicated in Section 3.8.1 and Section 3.8.2, B is the proportion of requests rejected and A is the rate at which requests are accepted by the system. Several specific scenarios are investigated for the performance measure R.

4.2 Performance of the Any Schedulability-based Broker

In the first experiment the Any Schedulability-based broker is examined in order to determine its performance. It is compared with a transparent broker that acts as a point of reference. Each broker has 1 resource during the simulation. Hence in these experiments the broker performs admission control and accepts a request only if it is schedulable. The service time, start time, and laxity are set to their default values as given in Table 1. The default underlying scheduling policy that is used at the resources is the Earliest Deadline First (EDF) scheduling policy. In this scenario the source of this system generates requests and sends them to the AS-based/transparent broker. The broker is responsible for determining whether the request can be scheduled on the available resource. The AS-based broker uses the Any Schedulability criterion and the transparent broker uses a schedulability algorithm based on the EDF policy used at the resource in order to determine schedulability. If the resource is capable of scheduling the request as determined by the broker, the request is sent to the resource where it is scheduled using the EDF scheduling policy. If the request cannot be scheduled the broker records this and discards the request. The resource executes requests in accordance with the EDF scheduling policy. Once finished, the requests are removed from the resource and the broker is notified of their completion. The request arrival rate is varied until a very high utilization and B value are reached. Figure 4-1 shows the performance of the AS-based broker with respect to the transparent broker labelled by EDF. The results show that as the arrival rate is increased the blocking ratio is also increased. It is interesting to note that although the AS-based broker did not use any *a prior* knowledge of the scheduling

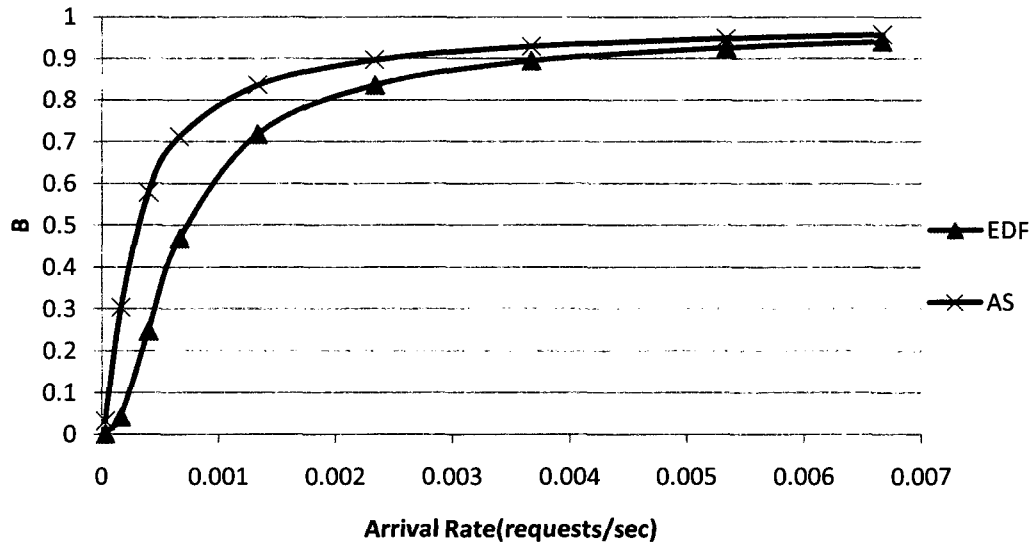

Figure 4-1: B values of the AS-based broker compared with the EDF-based broker for varying arrival rates

policy used at the resource, it performs comparably with the transparent broker at low and high arrival rates. The performance of the two brokers are comparable at low arrival rates because, irrespective of the strategy used by the broker, the resource contains a low number of requests in its scheduling queue allowing for more requests to be accepted. At high arrival rates the resource becomes saturated irrespective of whether an AS-based or a transparent broker is used thereby causing comparable results. Since the transparent broker can accept more requests at medium arrival rates the transparent broker performs better than the AS-based broker. Figure 4-2 shows the corresponding A values for the experiment. The figure shows that at higher arrival rates, with a decreased blocking ratio, the EDF-based broker is capable of servicing more requests per unit time than the AS-based broker. The lower values of A achieved with the AS-based broker occur because it

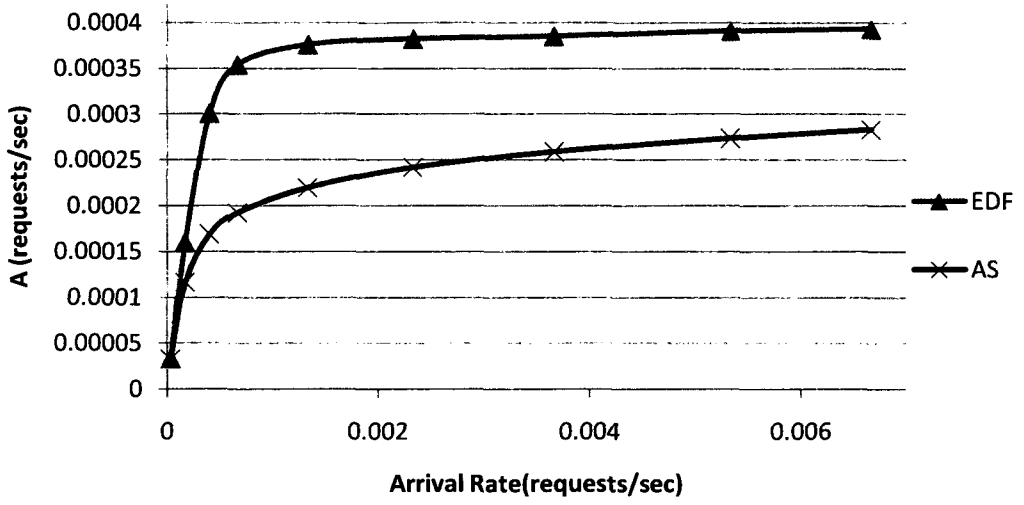

Figure 4-2: A values of the AS-based broker compared with the EDF-based broker for varying arrival rates

considers the worst case scenario when determining schedulability thereby limiting the number of requests that are accepted.

Figure 4-3 compares the B values of the AS-based broker with the EDF-based broker when the system contains 50 resources. This experiment is performed in order to study a scenario that is more common in a grid environment that is characterized by multiple resources. Although for both brokers the B values in this scenario are lower compared to that achieved with a single resource, the difference between the B values for a given λ between brokers has also decreased. The increase in resources allows the AS-based broker more options for allocating requests; this diminishes the limitations of the AS-based broker. A similar outcome is seen for the A values when using 50 resources seen in Figure 4-4.

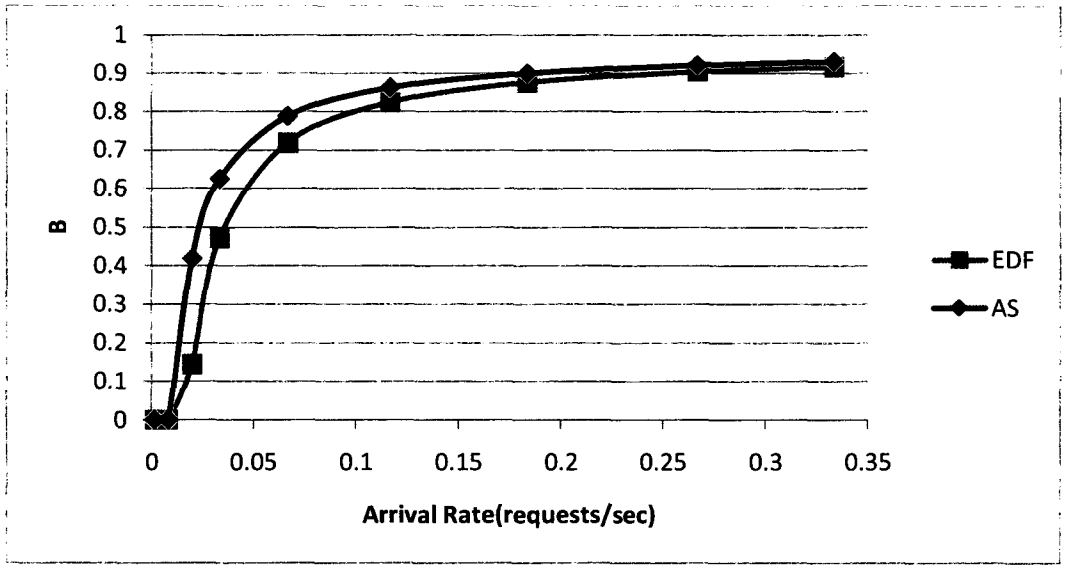

Figure 4-3: B values for the EDF and AS-based brokers, no. of resources = 50

4.3 Effect of the Number of Resources

The effect of the number of resources in the resource pool on performance is examined next. The number of resources, N_r, in the pool is set to 1, 5, and 10. The Any Schedulability-based broker is used in the simulations along with the default values for the workload parameters presented in Table 1. In this scenario the AS-based broker picks a resource based upon its allocation algorithm, in this case FF. A discussion of FF is presented in Section 3.1.1. If the request can be scheduled on the chosen resource the request is sent to the resource for execution. If the broker concludes that the resource cannot meet the request's deadline the broker attempts to allocate it on the next resource. This is continued until a suitable resource is found or all resource options have been considered. The purpose of having a larger resource pool is that it allows the broker additional options in the case that a request cannot be scheduled on the initial or

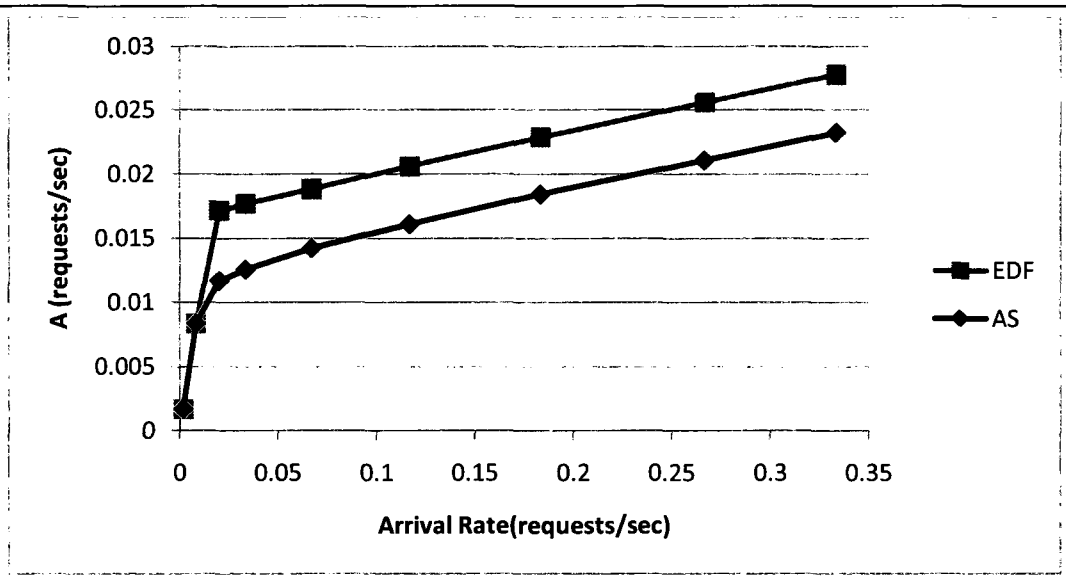

Figure 4-4: A values for the EDF and AS-based brokers, no. of resources = 50

subsequent choice(s). If a request cannot be scheduled on any of the resources, the request is rejected.

Figure 4-5 shows the B values achieved with different number of resources. The results demonstrate that for a given λ, a lower B value is achieved with a high number of resources. Using more resources seems to be most beneficial in the medium values of λ. This is because at very low λ there is very low contention for resources and the requests can be handled by 1 resource. At higher values of λ there is more contention for the resource causing a larger number of blocked requests resulting in larger B values for all scenarios. The corresponding A values, displayed in Figure 4-6, shows that a larger resource pool is capable of servicing more requests.

4.4 Effect of the Underlying Scheduling Policy

The scheduling policy used by the resources may affect system performance. Thus experiments are conducted to study the impact of this scheduling policy on performance.

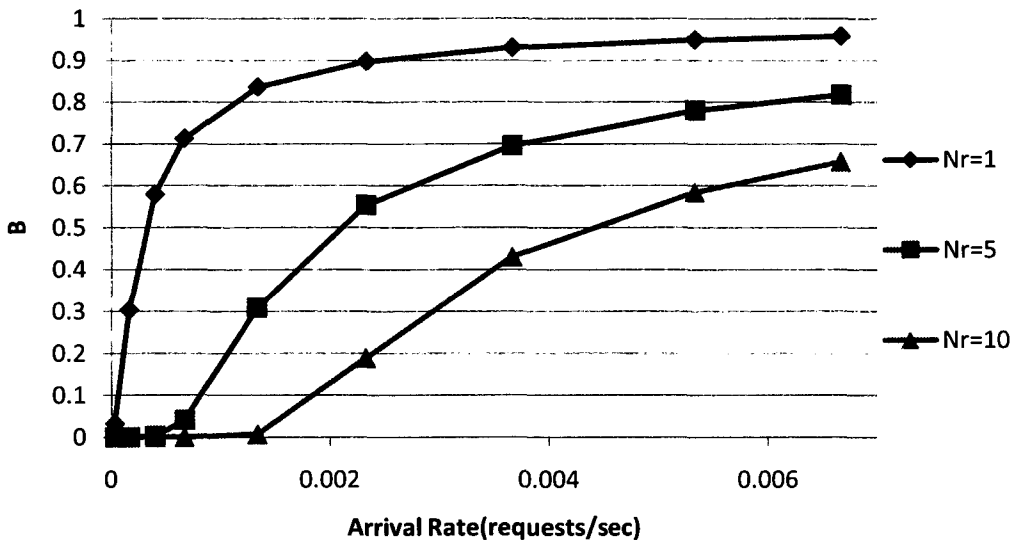

Figure 4-5: B values for the AS-based broker for various number of resources

Two scheduling policies are simulated: Earliest Deadline First which schedules requests based on deadlines (Section 3.5.1) and Earliest Start Time First which schedules requests based on earliest start times (Section 3.5.2). The varying parameter in this experiment is the resource's scheduling policy and the accompanying transparent broker's schedulability algorithm to match that of the resource. A resource pool of 1 resource is used with the values of the other parameters held at their default values presented in Table 1.

The results shown in Figure 4-7 indicate that the different scheduling policies give rise to nearly the same B values for a given λ. The A values seen in figure 4-8 also shows similar results which is expected due to the very close B values. The EDF curve is not visible because the values for both the EDF and ESF curve are nearly the same. Because of the close performance of the EDF-based broker and ESF-based broker, only the EDF

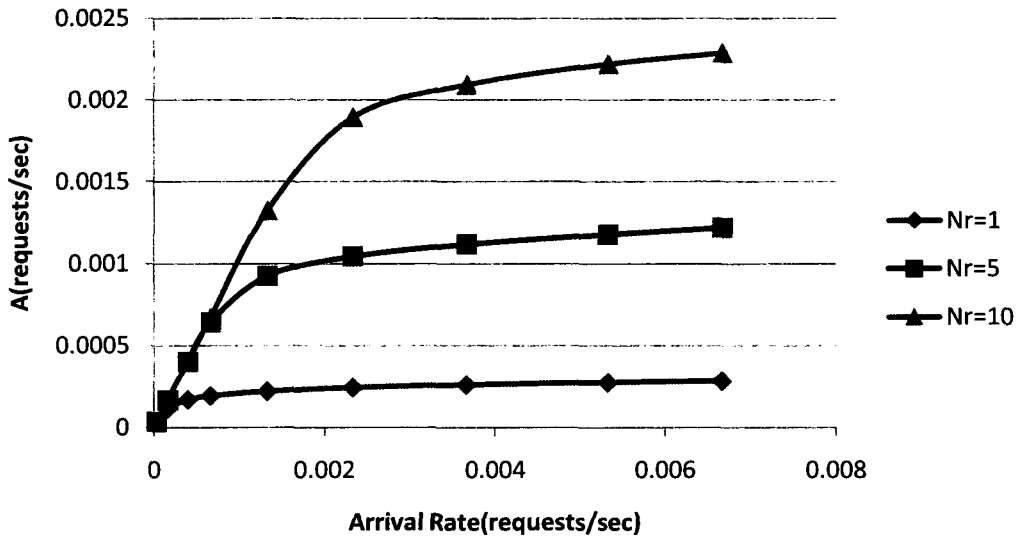

Figure 4-6: A values for the AS-based broker for various number of resources

based broker has been used in the other sections that compare its performance with an AS-based broker.

4.5 Effect of the Allocation Algorithm used by the AS-based Broker

The performance of the different allocation algorithms is investigated. In this experiment the AS-based broker is used. The resource pool contains 5 resources. Figure 4-9 and Figure 4-10 show the B and A values respectively. The results indicate that the allocation algorithms do not have any significant impact on the performance of the AS-based broker. The curve for First Fit is not visible in the figures because the values for both the First Fit and Next Fit curve are nearly the same.

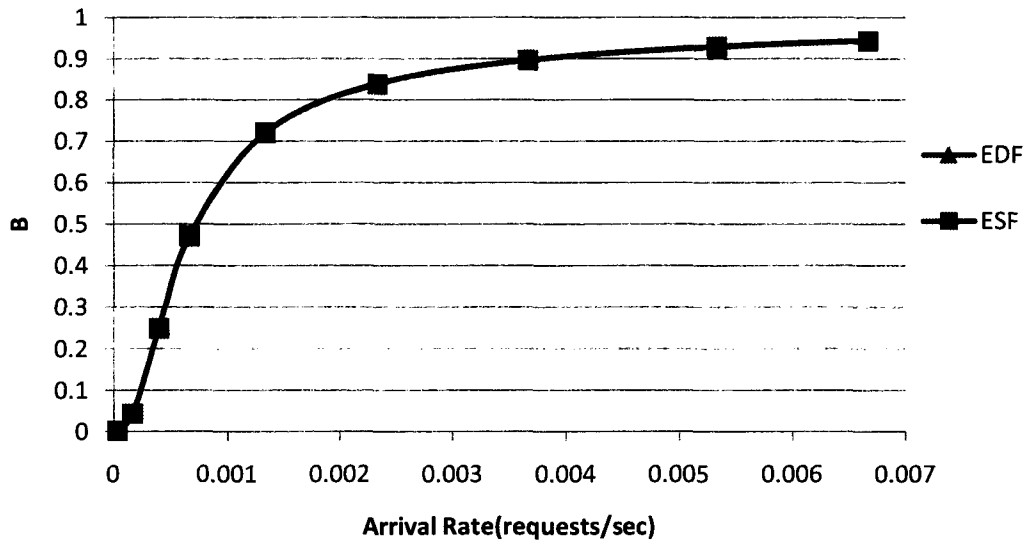

Figure 4-7: B values for brokers using *a priori* knowledge of the resources' scheduling policy.

4.6 Effect of Service Time

The effects of service time on performance are investigated next. Two experiments are considered when investigating service time. One focus is the impact of the variability factor between the upper and lower bounds of the distribution the other investigates the impact of the mean service time on performance. The variability factor characterizes the distance between the upper and lower bounds of the service times: lower bound = (1-variability factor)*mean; upper bound = (1+variability factor)*mean .The default value of the variability factor is 0.8. With a mean service time of 50 minutes this generates requests with service times between 10 ((1-0.8)*50) minutes and 90 ((1+0.8)*50) minutes. For both experiments the AS-based broker is used with a resource pool of 1 resource. The λ values used are the same as those used in other experiments with one resource. All other parameters are kept at their default values, see Table 1.

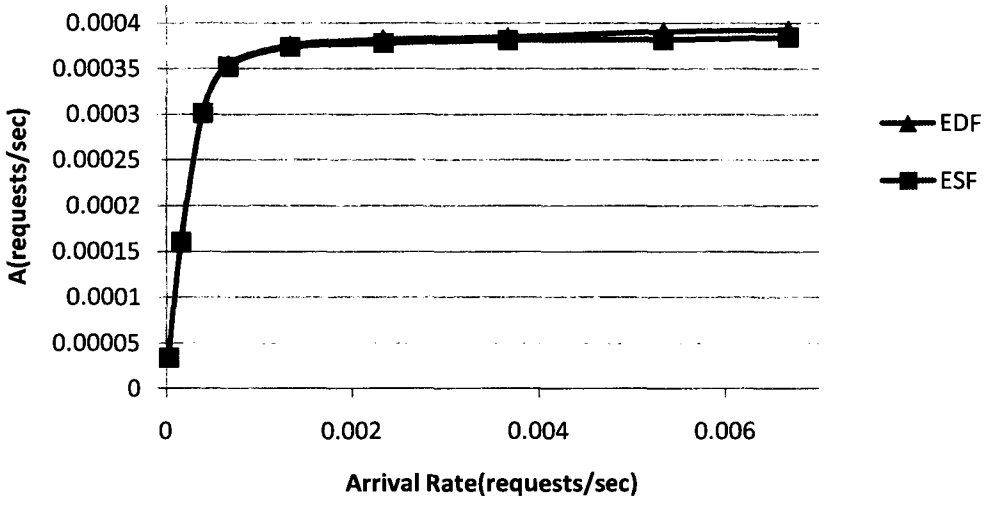

Figure 4-8: A values for brokers using *a priori* knowledge of the resources' scheduling policy.

4.6.1 Effect of the Service Time Variability Factor

In the first set of experiments the variability factor, that gives the distance between the upper and lower bound of the service time uniform distribution, is changed. A constant mean service time of 100 minutes is used while the variability factor is changed to three different ranges giving three different levels of variability. The low level of variability uniform distribution creates request service times between 98 and 102 minutes, the medium level of variability uniform distribution creates request service times between 50 and 150 minutes, and the high level of variability creates request service times between 2 and 198 minutes. These levels give rise to variability factors of .02, 0.5, and 0.98 respectively.

The B values achieved are shown in Figure 4-11. This figure shows that the lower values of B are achieved for workloads with a high level of variability followed by

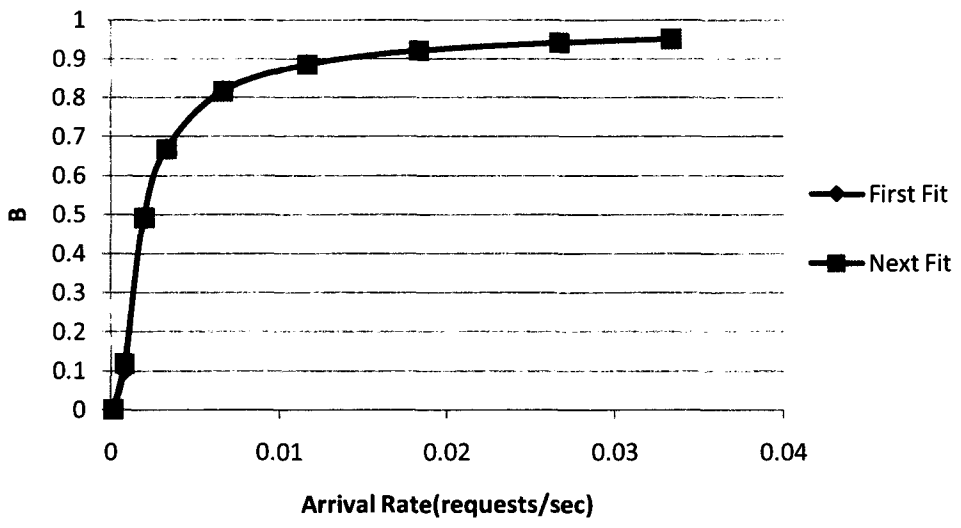

Figure 4-9: B values for the AS-based broker using different allocation algorithms

workloads with medium and low variability. The medium level cannot be seen in the figure because its values are close to that of the low level. The AS-based broker performs best when request service times have a large variability because there are more requests with smaller service times produced. This combined with the fact that the laxity is related to the mean service time gives rise to small service time requests with a relatively larger window of execution. This allows for more requests to be accepted and thereby reducing the blocking ratio.

In the case of a workload with a level of low variability all requests have service times that are close in value to each other and also to the mean. Because a laxity value of 200% of the mean service time is used in the experiment the laxity value is approximately double the service time of requests in this workload. This limits the schedulability of the requests compared to the case where a workload with high level

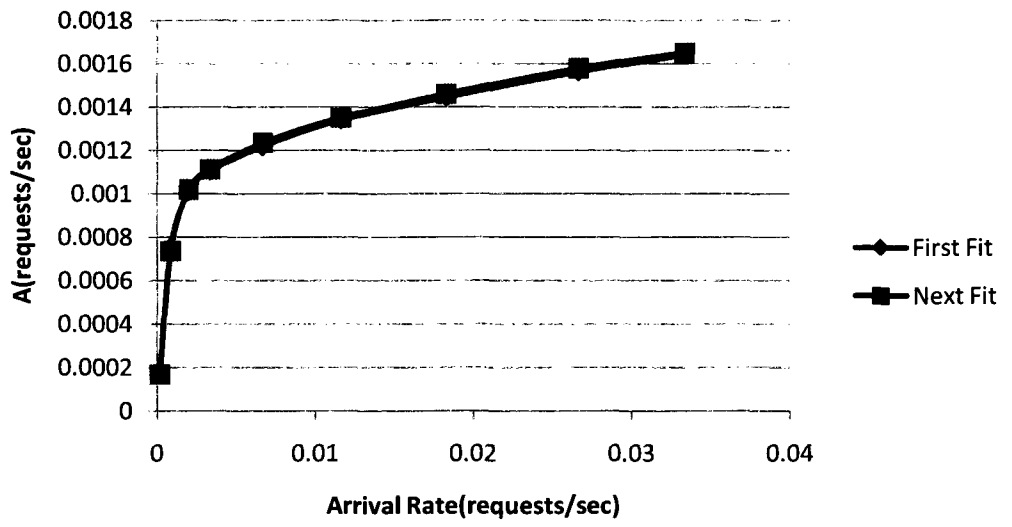

Figure 4-10: A values for the AS-based broker using different allocation algorithms

variability is used and causes the blocking ratio to be higher. A similar outcome is observed for the medium level of variability case, where the requests are larger in comparison to the low level of variability and gives rise to limited schedulability. The only exception to this is at very low λ values where there is low contention for arriving requests and comparable B values are obtained for the three levels of variability.

Figure 4-12 shows the A values obtained in this experiment. Although the workload with a high level of variability gives rise to a slightly better B values, this improvement is magnified in the A values. The high variability workload scenario outperforms the two others by substantial margins The A values for both the low and medium level of variability exhibit comparable A values for a given λ.

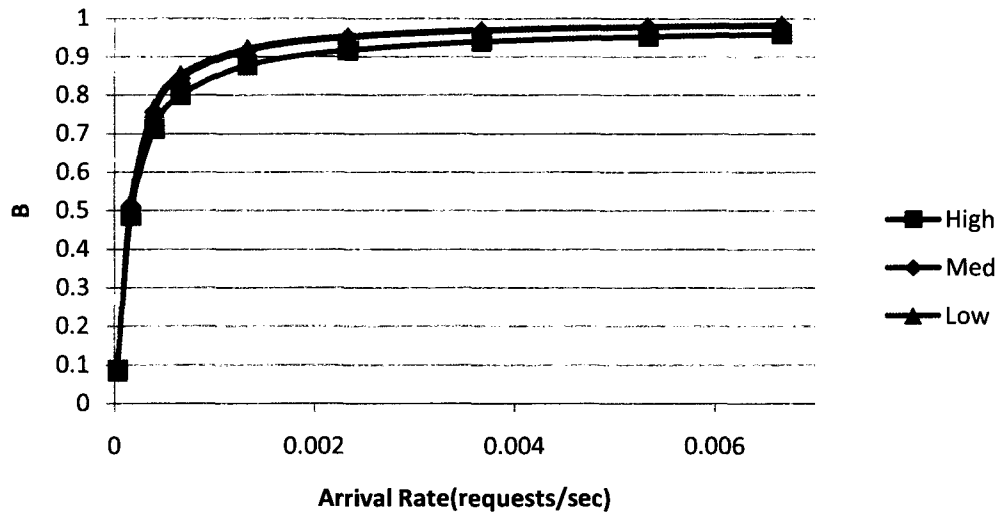

Figure 4-11: B values for the AS-based broker for different variability factors

4.6.2 Effect of the Mean Service Time

This set of experiments concerns the impact of changing the mean service time, while holding the variability factor at 0.8 has on system performance. In these experiments the AS-based broker is used with a system comprising 1 resource. Three levels of mean service times are examined. The low level mean service time is set to 10 minutes and has an upper and lower bound of 18 and 2 minutes respectively. The medium level mean service times is set to the default mean value of 50 minutes with an upper and lower bound of 90 and 10 minutes respectively. The high level of mean service time is set to 200 minutes with an upper and lower bound of 360 and 40 minutes respectively. Figure 4-13 shows the B values observed. This figure indicates that for a given λ, a lower mean service time leads to lower B values. This outcome is intuitive because requests with

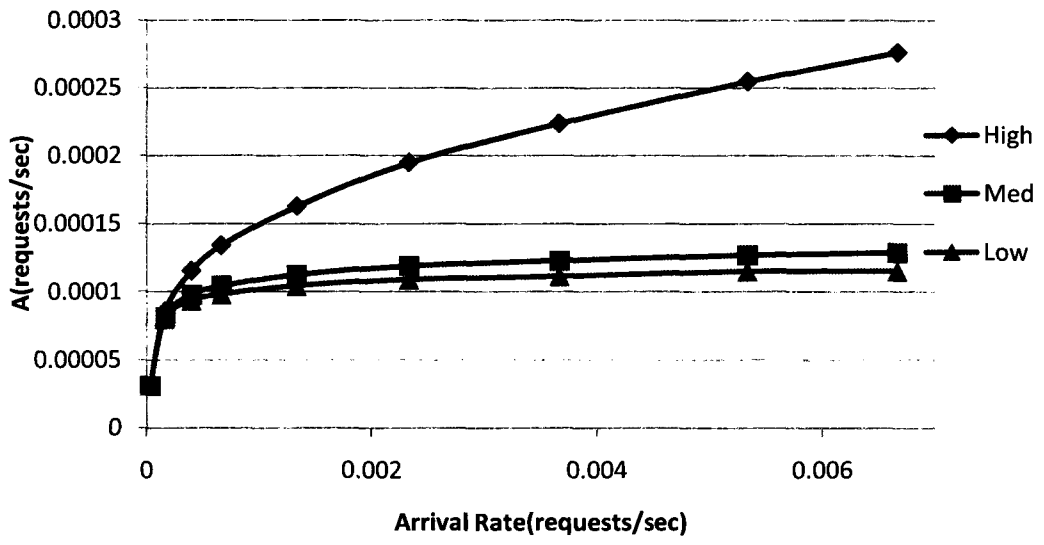

Figure 4-12: A values for the AS-based broker for different variability factors

smaller service times spend less time on the resource and have less of a chance of interfering with other requests

Figure 4-14 displays the A values for this scenario. These A values follow the pattern exhibited by the B values. For the highest value of λ experiments with the highest value of A are obtained with low level mean values.

4.7 Effect of the Request's Earliest Start Time

This section focuses on determining the effect of the request's earliest start time on performance. By default the earliest start time is modelled by a uniform distribution with an upper and lower bound of 12 and 0 hours respectively. The experiments performed in this set involved changing the upper bound of the start time to various other values. These values are chosen as 1, 5, and 20 hours. The AS-based broker is used in all experiments with a resource pool of 1 while all other parameters are set to their default values (see

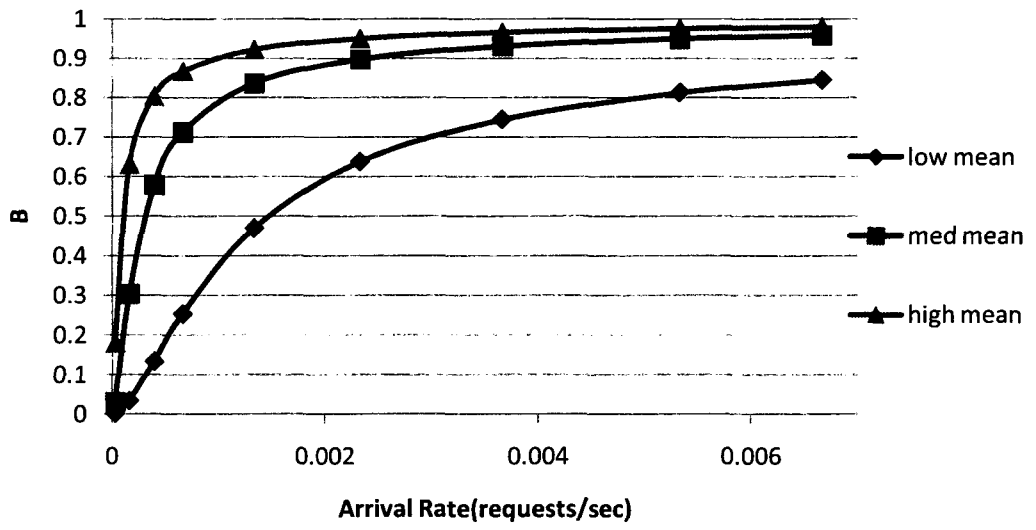

Figure 4-13: B values for the AS-based broker with requests for various mean service times

Table 1).Figure 4-15 shows the resulting curves for the B values for this experiment. The values for an upper bound of 20 hours are not seen in the figure because it has similar values to the case where the upper bound is 12 hours. The figure shows that for a given λ value, a lower upper bound on the earliest start time gives rise to lower B values when compared to requests with a higher upper bound on the start time. This is due to requests being scheduled earlier and hence being executed sooner which in turn leads to requests remaining on the resource for a relatively short amount of time. As requests are kept on the resource longer, the chances for interference and overlapping with other requests increases thereby leading to higher B values. A decrease in B is expected to be accompanied by an increase in A. This is confirmed in Figure 4-16 which shows the A values for this experiment. Since the requests are leaving the resource at a faster rate, the resource is able to effectively schedule more requests per unit time.

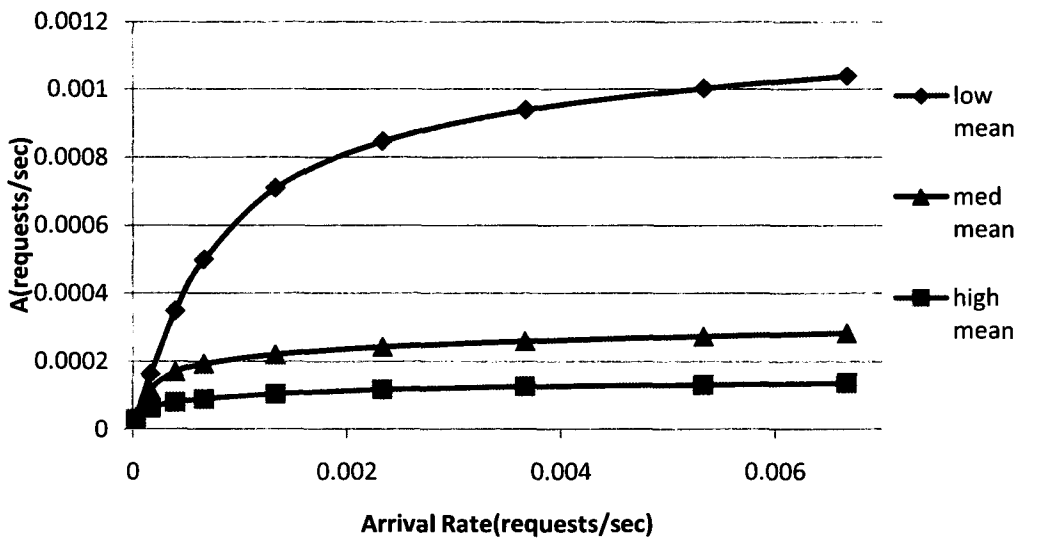

Figure 4-14: A values for the AS-based broker with requests for various mean service times

4.8 Effect of Laxity

This section focuses on analyzing the effect of laxity of the requests on system performance. By default requests are given a laxity of 200% of their mean service time (Section 3.8.2.4). For this set of experiments the laxity is varied from 100% to 200% to 500% to 1000%. All other parameters are set to their default values (see Table 1). Figure 4-17 shows the B values achieved. The values for the case where laxity is set to 100% are not visible because its values are close to 200% laxity case. Changing the laxity values inherently modifies the deadline of the requests allowing the resource less or more time to complete the request depending on whether the laxity is decreased or increased. With an increase in laxity the resource is granted more time to complete the request and the opposite is true for a decrease in laxity. Thus, as can be seen in Figure 4-17, for medium and high values of λ, an increase in laxity causes a decrease in B. The decrease in B is

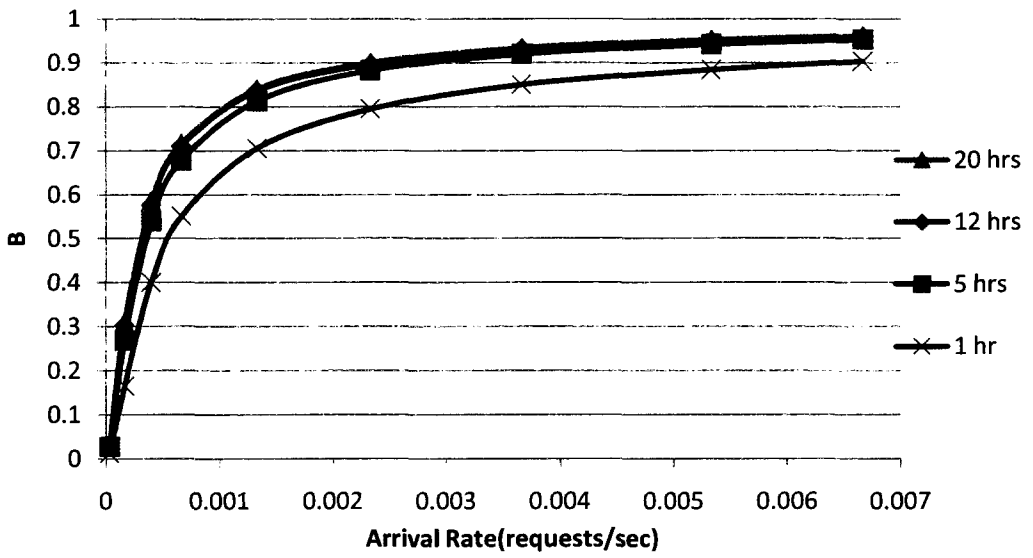

Figure 4-15: B values for the AS-based broker with requests for various earliest start times

most notable at the medium ranges of the curve. At very low λ and at very high λ the improvement in B is smaller. At low λ the requests do not arrive fast enough to cause significant overlapping of requests to cause high blocking ratio values. At high λ the requests arrive such that the system tends to become saturated in each case. The increase in laxity aids in alleviating the strain put on the system but requires a large increase in laxity for a fairly small improvement in performance. Figure 4-18 shows the A values for this experiment. The observation made from this figure is that a decrease in blocking ratio that accompanies a larger laxity causes an increase in system throughput.

4.9 Effect of Requests with a Higher Variability in Service Times

All the other experiments described earlier had requests with times generated by using a uniform distribution resulting in values of coefficient of variation lower than 1. This

section focuses on looking at the effect of using a hyper-exponential distribution to create requests with a higher variability in service times. More specifically, the effects of larger coefficient of variation of request service times on system performance are explored. The coefficient of variation is the ratio of the standard deviation to the mean and is the measure of dispersion of values from the mean. For the experiments described in this section, the coefficient of variation is set to 1, 2, and 3. In this set of experiments the AS-based broker has a resource pool of 1 resource and default values for all other parameters (Table 1). In principle the experiments performed here are similar to the experiments performed in section 4.6.1. Hence a similar result is expected. Figure 4-19 gives the B values for the experiments performed. The results show that for a given λ, requests with a larger coefficient of variation produce lower B values. This is due to the fact that with a greater coefficient variation there are a larger number of requests with smaller service

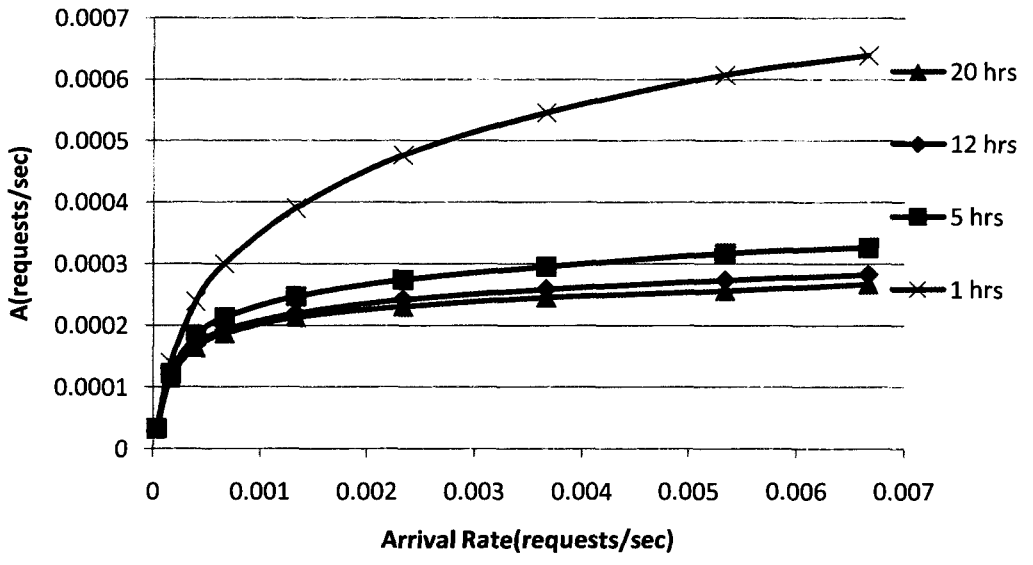

Figure 4-16: A values for the AS-based broker with requests for various earliest start times

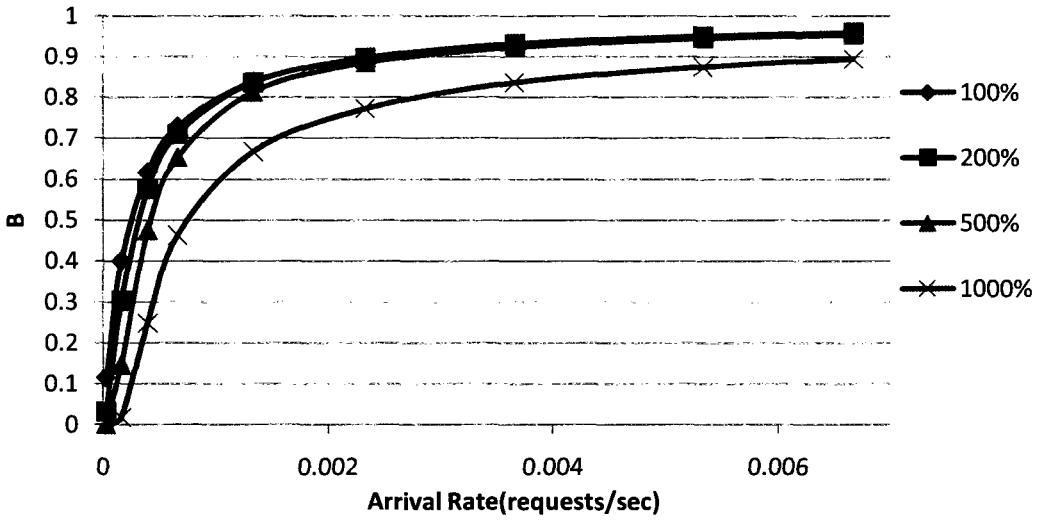

Figure 4-17: B values for the AS-based broker for various laxities of requests

times. These requests with a smaller service time are likely to be accepted and hence reduce the blocking ratio as seen in Section 4.6.1. B values for the different values of the coefficient of variation are close to each other for low and high λ. Large differences in B values are observed for intermediate λ. At low λ the system performance is comparable across all values used for the coefficient of variation. At high λ the system is saturated and the benefits of using higher coefficient of variations are reduced. Figure 4-20 shows the results of the corresponding A values observed. The results show that for a given λ an increase in the coefficient of variation results in an increase of the A values.

4.10 Effect of Hybrid Matchmaking Strategies

This section focuses on systems using both opaque and transparent resources as mentioned in Section 3.6.

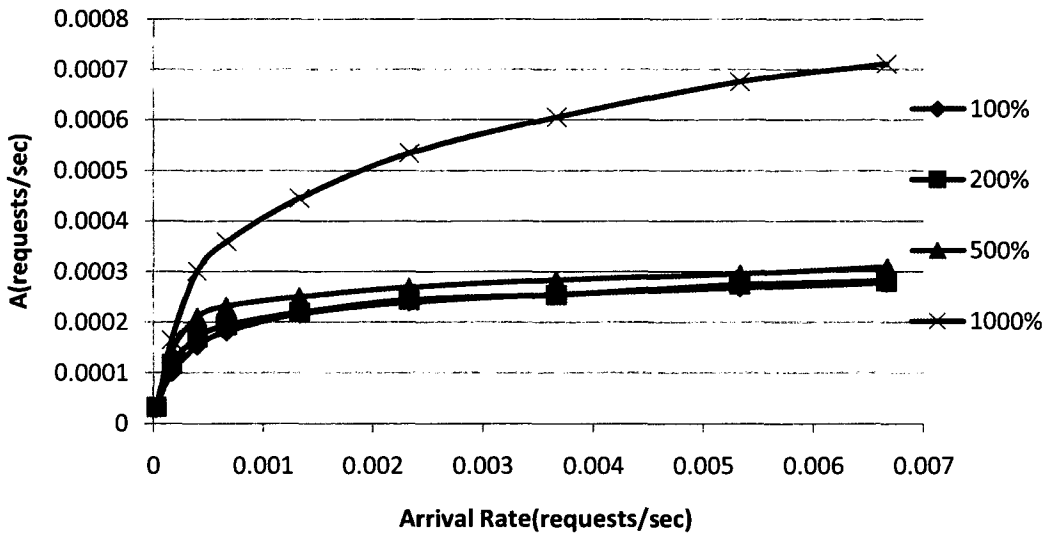

Figure 4-18: A values for the AS-based broker for various laxities of requests

4.10.1 Effect of Workload Distribution in the Independent Strategy

Two brokers, one managing transparent resources and another managing opaque resources are used. With the independent strategy the workload is distributed among the two resource pools (and brokers) such that each resource pool is subjected to an independent stream of request arrivals. Each resource pool is given a certain proportion of the workload. Hence if the requests that arrive at the system have an arrival rate of λ, then the transparent resource pool receives requests at a rate of $(x)(\lambda)$ and the opaque resource pool receives requests at a rate of $(1-x)(\lambda)$ (where $0 < x < 1$). For the experiments discussed in this section the value of x is varied from 0.9 to 0.5 to 0.2 while the total number of resources is kept at a constant value of 10. Two sets of experiments are conducted such that the number of resources allocated to each resource pool is different. In the first set of experiments there are 9 transparent resources and 1 opaque resource. In

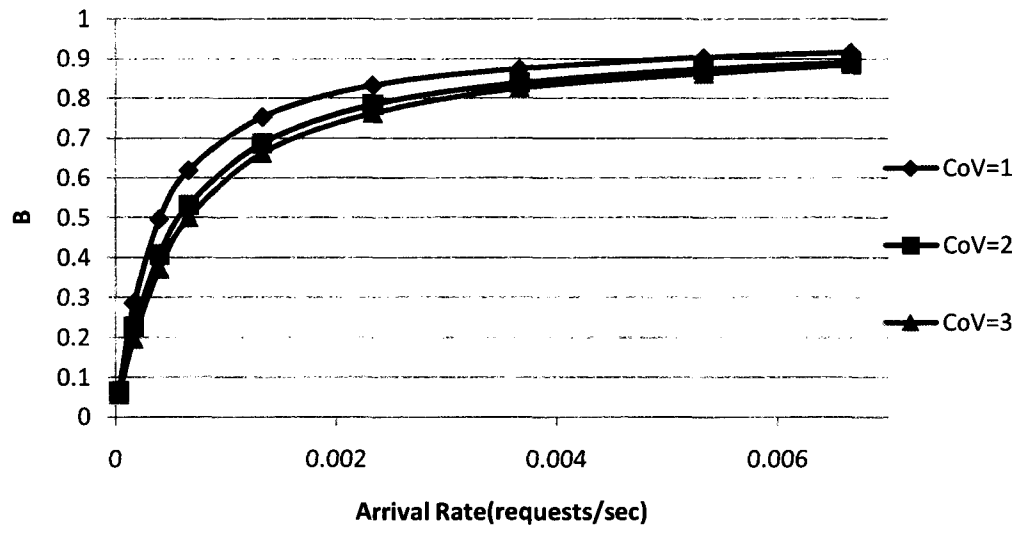

Figure 4-19: B values for the AS-based broker for various coefficient of variations of request service times

the second set of experiments there are 5 transparent resources and 5 opaque resources. All other parameters are set to their default values (see Table 1).

Figure 4-21 shows the B values for the experiments performed using 9 Transparent and 1 Opaque resources. The results obtained show that at low and medium values of λ distributing the workload such that the transparent resource pool is assigned most of the requests works best (90%) followed by splitting the workload equally. Providing 80% of the workload to the opaque resource pool gives rise to the highest B for a given λ. The transparent resource pool contains 9 resources and therefore by forwarding most of the workload to it leads to a lower blocking ratio.

The most significant areas of difference occur at low λ values. At medium to high λ values the all resources begin to saturate, thereby causing all B values to have relatively similar values. Figure 4-22 shows the B values for the experiments for the independent strategy with 5 transparent and 5 opaque resources. The results show that at low values of

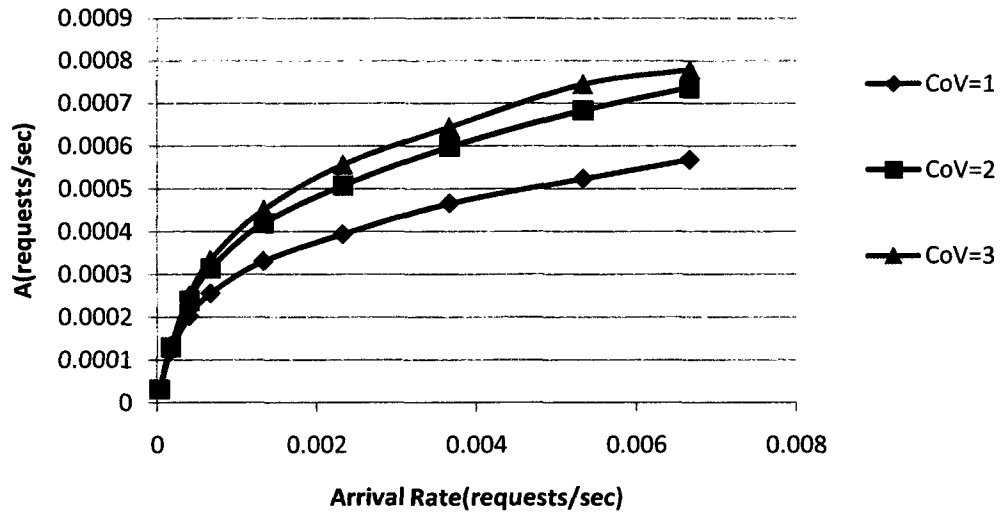

Figure 4-20: A values for the AS-based broker for various coefficient of variations of request service times

λ all cases perform similar to each other, as λ increases x=0.5 performs the best while at high λ all scenarios exhibit nearly the same B values for a given λ.

At intermediate values of λ, using a workload distribution with x=0.9 performs better than using x=0.2. This is because the transparent resources are capable of handling the request of arrivals better than the opaque resources. Hence, when the transparent resources receive 90% of the workload they are still capable of achieving a greater system performance in comparison to the situation in which the opaque resources receive 80% of the workload (x=0.2).

At high values of λ distributing the workload evenly among the resource pools performs the best since the resource distribution (5 resources for each broker) matches the workload distribution. Eventually once the arrival rates reach very high values the

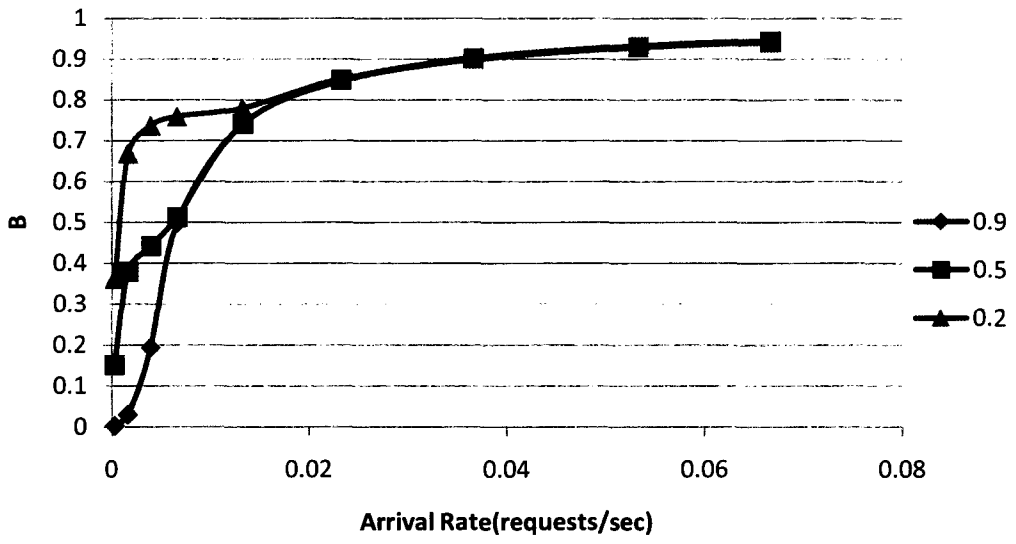

Figure 4-21: B values for independent strategy for various workload distributions using 9 transparent and 1 opaque resources

resource distribution of the system has little effect because in all cases the resources are saturated. Figures 4-23 and 4-24 show the A values for the experiments.

Overall, the results indicate that the x value that results in the best performance is one that best matches the resource distribution. For example, in the resource distribution that results in 9 transparent resources and 1 opaque resource the workload distribution that gave the best performance is the distribution where the transparent resources received 90% of the workload and the opaque resources received 10% of the workload. However, once saturation begins to take place all the different values of x tend to produce similar values of B.

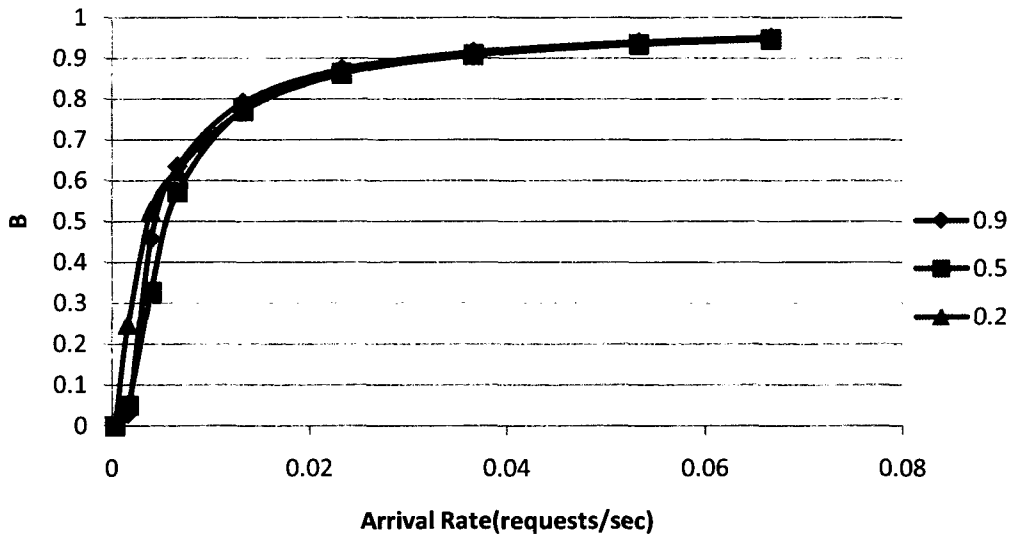

Figure 4-22: B values for independent strategy for various workload distributions using 5 transparent and 5 opaque resources

4.10.2 The Combined Strategy

In the combined strategy the system comprises two brokers: 1 transparent and 1 opaque each with their own resource pool. However, unlike the independent strategy this strategy has a main resource pool comprising transparent resources and a secondary resource pool comprising opaque resources. The workload is not distributed and goes directly to the main resource pool via the transparent broker. If the broker is capable of finding a suitable resource for scheduling a request it accepts the request. If the broker is unable to accept the request, it sends the request to the secondary broker which then attempts to find an available resource for the request. Two experiments are conducted for this strategy. In the first set of experiments the transparent broker manages a resource pool of 9 resources while the opaque broker handles a resource pool of 1 resource. In the second set of experiments both resource pools contain 5 resources. All other workload

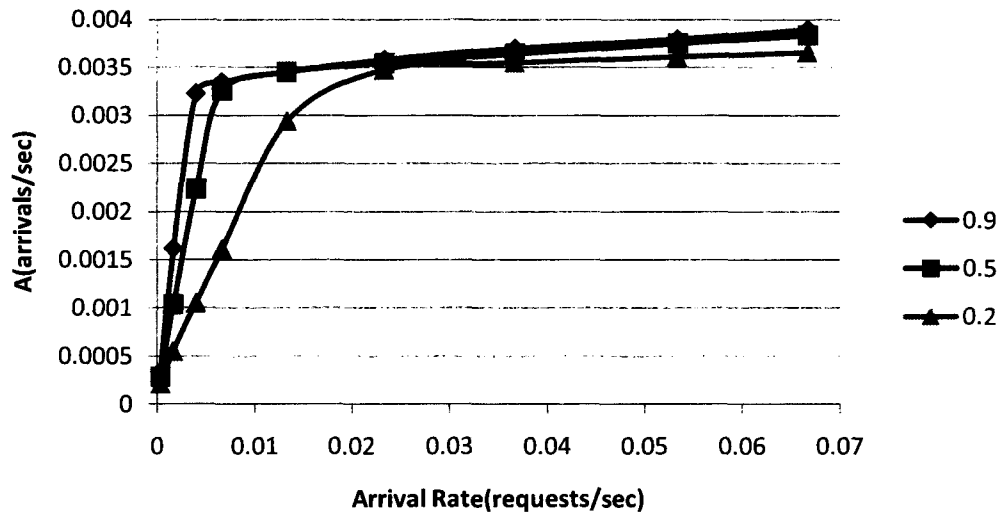

Figure 4-23: A values for the independent strategy for various workload distributions using 9 transparent and 1 opaque resources

parameters are kept at their default values (see Table 1). Figure 4-25 shows the B values for the two sets of experiments and Figure 4-26 shows the corresponding A values.

The figures show that the strategy where there are 9 transparent resources and 1 opaque resource (9T, 1O in the figure) outperforms the strategy where both resource pools contain 5 resources (5T, 5O in the figure). The former strategy performs better since transparent resources are capable of handling more requests than opaque resources. As this strategy contains more transparent resources than the alternative we expect a better performance. The superiority of the 9 transparent resources and 1 opaque resource is best witnessed at medium λ values whereas at very low λ values and high λ values the difference between the curves is not as significant. The rationale behind such behaviour is the same as that explained in Section 4.2.

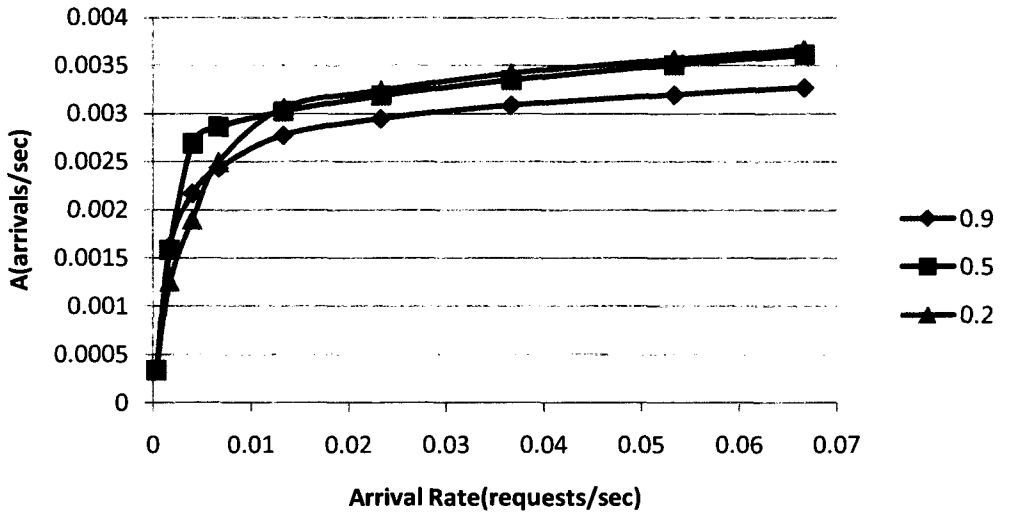

Figure 4-24: A values for the independent strategy for various workload distributions using 5 transparent and 5 opaque resources

4.10.3 Comparison of Hybrid Matchmaking Strategies

Sections 4.10.1 and 4.10.2 present two possible matchmaking strategies that combined both opaque and transparent resources. This section compares these two matchmaking strategies with each other as well as to a system with only transparent resources. The baseline system consists of a transparent (EDF-based) broker with 5 resources. This system is used as the system for which comparisons are made to. All other workload parameters to set their default values (Table 1). The independent and combined strategies consist of 5 transparent (EDF-based) and 5 opaque resources. For the independent strategy x is set to 0.5 since under the given resource distribution this workload distribution performs the best. The baseline system is given 5 resources instead of 10 in order to showcase the benefits of a system where opaque resources are incorporated

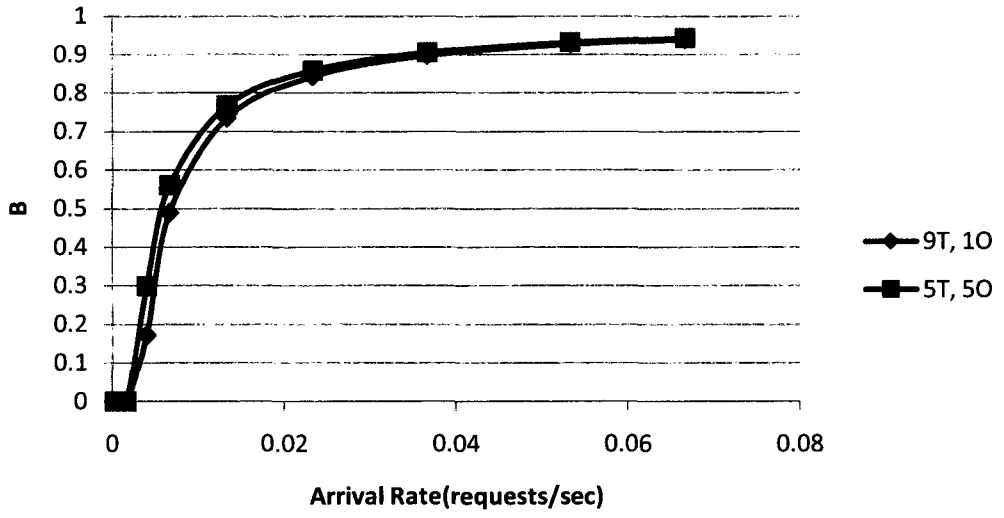

Figure 4-25: B values for the combined strategy with 9 transparent and 1 opaque (9T, 1O) and 5 transparent and 5 opaque (5T, 5O) resources

rather than neglected due to not having *a priori* knowledge of their scheduling policy. The values for both B and A are shown in Figure 4-27 and 4-28 respectively. In the figures Transparent Only refers to the baseline system.

The results show that both the combined and the independent strategies perform better than the baseline system. Overall, the combined strategy performs the best as can be seen in Figure 4-27. At very low arrival rates all strategies give rise to similar B values (see Figure 4-27) since the resource contention is low. Once the arrival rate begins to increase the benefit of having the extra resources becomes apparent as the independent and combined strategies both outperform the baseline system. Eventually all strategies tend to be become saturated at various arrival rates but the combined strategy still outperforms the independent strategy. The combined strategy outperforms the independent strategy because it first checks all possible transparent resources before sending it to the opaque resource. Whereas the independent strategy reserves some of the

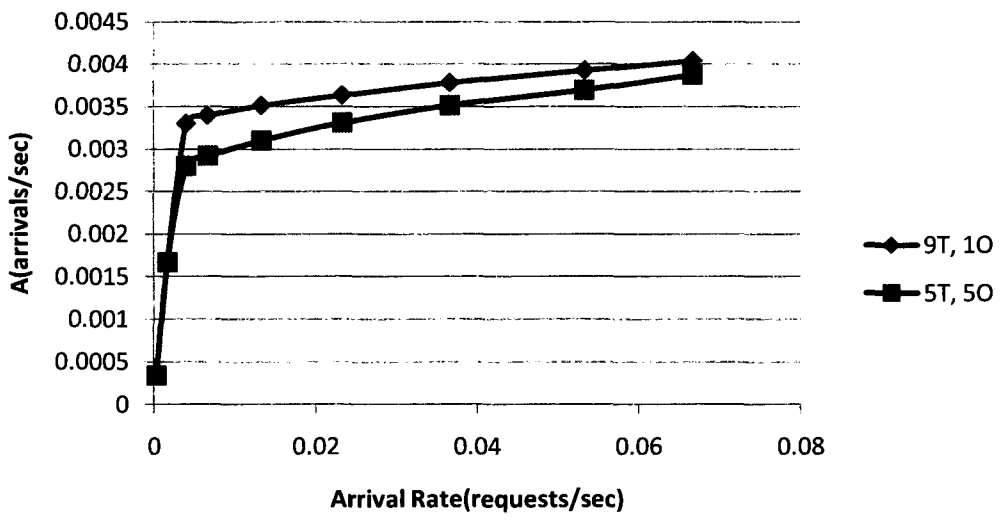

Figure 4-26: A values for the combined strategy with 9 transparent and 1 opaque (9T, 1O) and 5 transparent and 5 opaque (5T, 5O) resources

workload for the opaque resource which may contain requests that can be scheduled on a transparent resource and not on an opaque resource.

4.11 Effect of On Demand Requests

This section presents the effect of including On Demand requests in the workload on system performance. For all Advance Reservation requests the default workload parameters are used while the On Demand requests use the parameters mentioned in Section 3.9.3. On Demand requests comprise 20% of the requests. Figure 4-29 compares the B values of the AS and EDF-based broker when only 1 resource is used. Included in the figure are the results where the same system is used while using only Advance Reservation requests. The workload types W are denoted by AR for Advance Reservation only and by MI for a mixed workload. Using the On Demand requests provides little

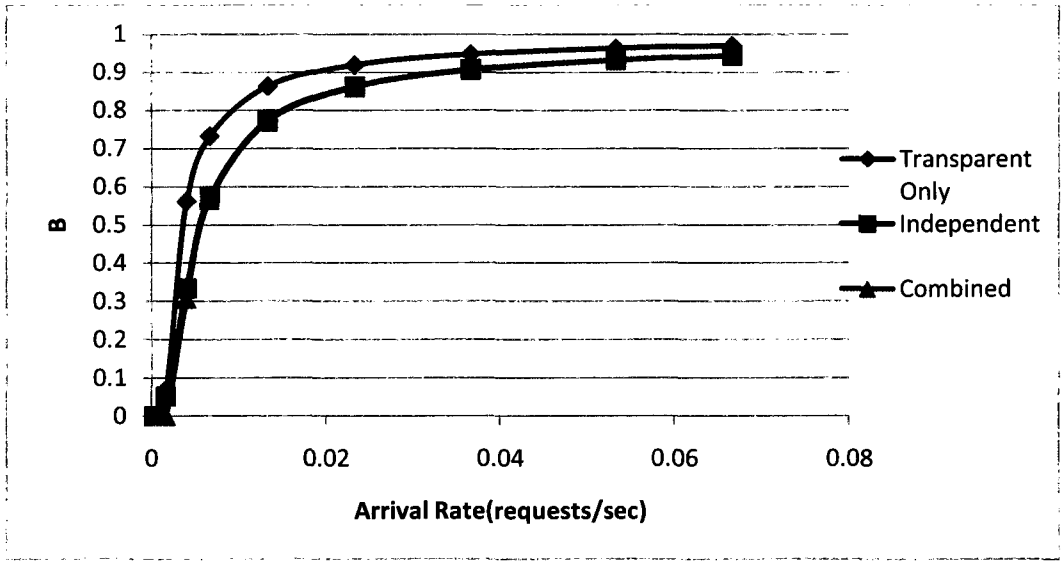

Figure 4-27: B values for the various matchmaking strategies

difference in B values achieved by the EDF-based broker (EDF, W=MI) in comparison to the B values achieved with Advance Reservation only requests (EDF, W=AR). However for the AS-based broker the On Demand requests lead to a superior performance (comparing AS, W=AR with AS, W=MI).

Improvements of 27% can be achieved with the inclusion of On Demand requests. This performance increase is the result of the On Demand requests always being schedulable. Out of the approximate 5000 requests for each λ value no On Demand request is rejected. For a given λ, the addition of the On Demand requests also translates to the arrival of less Advance Reservation requests, per unit time allowing the AS-based broker to accept more requests. Another important outcome of including On Demand requests is that the B values achieved with the AS-based broker and the EDF-based broker become more comparable especially at higher λ values. This is seen in Figure 4-29

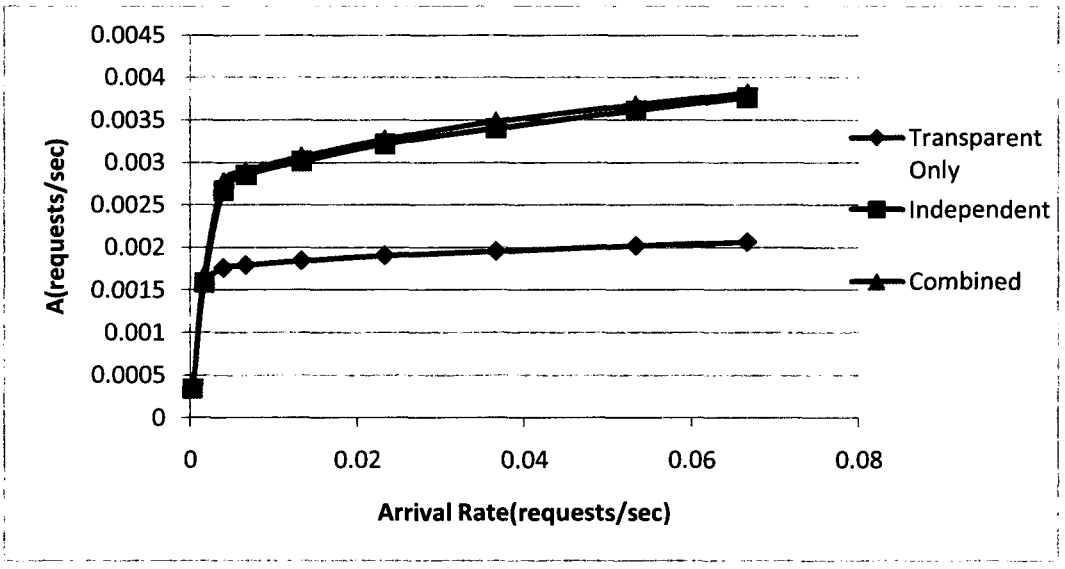

Figure 4-28: A values for the various matchmaking strategies

when comparing AS, W=MI with EDF, W=AR. When comparing AS, W=AR with EDF, W=AR the difference in performance is about 54% the difference after including On Demand requests the difference drops to approximately 28%. This is observed by comparing AS, W=MI with EDF, W=MI.

Figure 4-30 shows the A values for the experiment described earlier. This figure reinforces the results determined from the B values. The most notable difference occurs for the AS-based broker (AS, W=MI vs. AS, W=AR) where improvements of 34% are observed at medium to high λ values. This result is expected since none of the On Demand requests are rejected meaning that more requests are accepted into the system. Figure 4-31 and 4-32 show the B values and A values respectively for the system where each broker contains 10 resources. EDF, W=MI is not visible in Figure-31 but has similar B values as EDF, W=AR for low λ values and similar B values as AS, W=MI for high λ

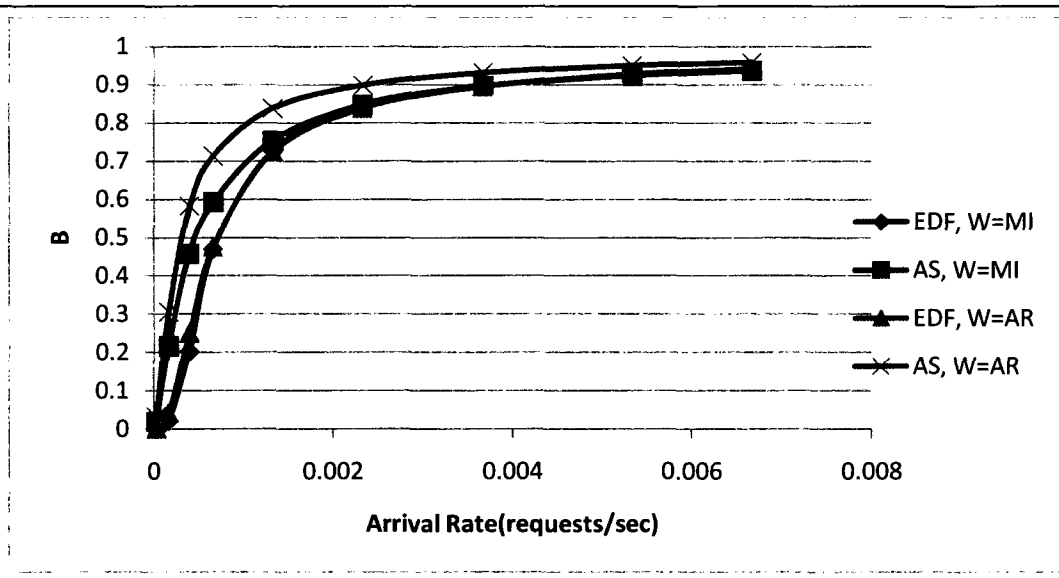

Figure 4-29: B values for the EDF and AS- based brokers under mixed and AR only workloads

values. The results show a similar outcome as Figures 4-29 and 4-30 for low λ to medium λ values, that On Demand requests significantly increase the performance of the AS-based broker. However at high λ values both AS and EDF based brokers with a mixed workload are capable of outperforming the EDF-based broker with an Advance Reservation only workload. This is most notable looking at the A values (Figure 4-32). Thus, including On Demand requests into the workload, especially when using multiple resources, allows the AS-based broker to improve its performance.

4.12 Revenue Rate Achieved on the System

As mentioned in Section 3.8.3 Revenue Rate is the rate at which a resource provider receives revenue per unit time. This section focuses on Revenue Rate (R) for important matchmaking strategies and not the effects the workload parameters have on R. The amount of Revenue that can be earned from the resources is limited by the number of

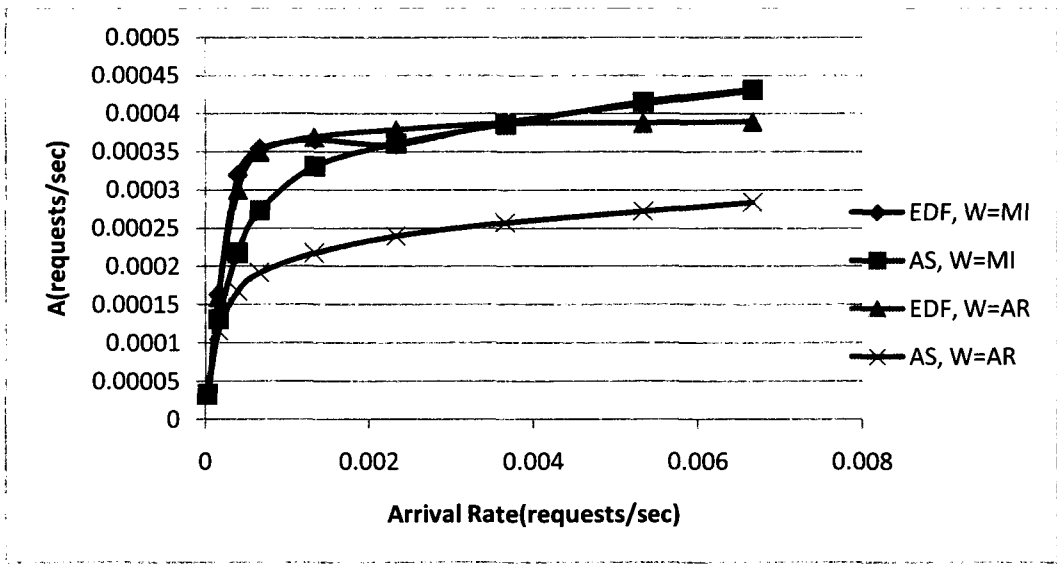

Figure 4-30: A values for the EDF and AS-based brokers under mixed and AR only workloads

resources. For example, a system comprising 1 resource at full capacity is able to return at most 1 cunits/sec. 2 resources at full capacity can return at most 2 cunits/sec.

4.12.1 Revenue Rate Achieved with a Single Broker

This section focuses on the revenue rate achieved when a single broker is used with both a single resource as well as multiple resources. For this set of experiments all workload and system parameters are set to their default values (see Table 1).

Figure 4-33 shows the R values for the AS-based and EDF-based brokers where each broker has one resource. The result derived from this figure shows that the AS-based broker achieves about 58% of the R achieved by the EDF-based broker. The EDF-based broker, once saturated, is capable of reaching R values close to 1. Clearly the constraints of the AS-based broker cause a significantly lower Revenue Rate at higher λ values.

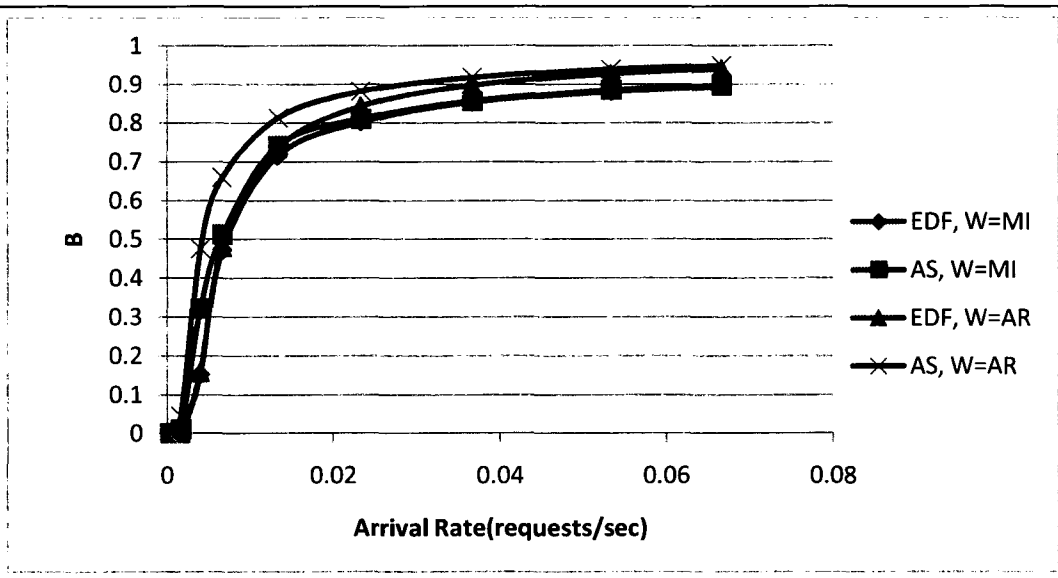

Figure 4-31: B values for the EDF and AS-based brokers with 10 resources subjected to mixed and AR only workloads

Figure 4-34 shows the R values for the AS-based and EDF-based broker when both brokers have 10 resources. In this scenario the AS-based broker displays a better R, but it still does not perform nearly as good as the EDF-based broker. It performs approximately 67% as effectively as the EDF-based broker.

4.12.2 Revenue Rate Achieved by Hybrid Strategies

One of the benefits of including the AS-based broker and opaque resources is that it has the capability of increasing the number of resources available for executing requests.

This section focuses on the Revenue Rate achieved by the different matchmaking strategies. The strategies that are compared are an EDF-based broker with 5 transparent resources (Transparent Only in Figure 4-35), the independent strategy with 5 transparent and 5 opaque resources (x=0.5), and the combined strategy with 5 transparent and 5 opaque resources. All other parameters are set to their default values (see Table 1). The

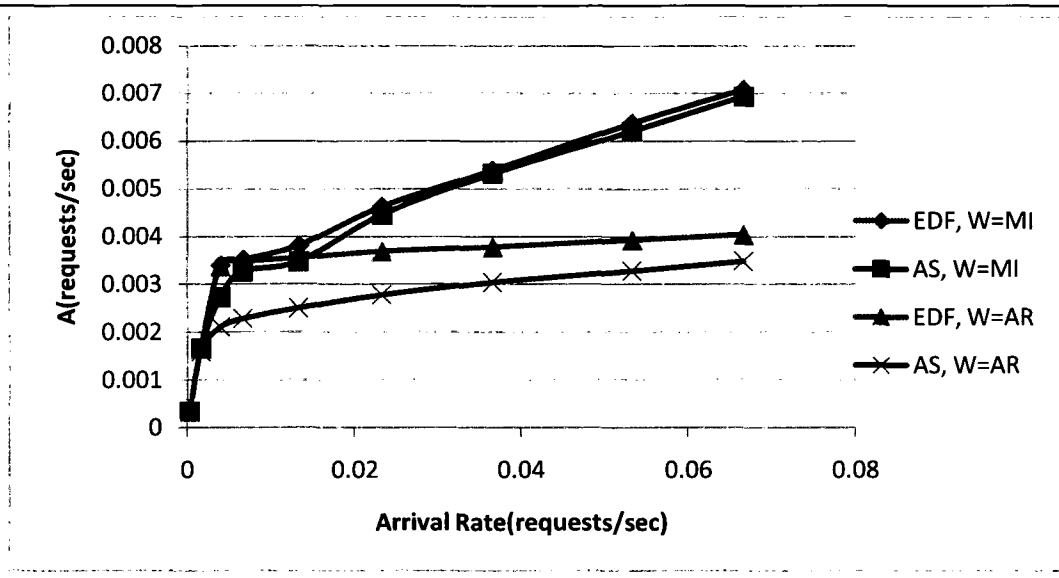

Figure 4-32: A values of EDF and AS-based brokers with 10 resources subjected to mixed and AR only workloads

results are shown in Figure 4-35. It is evident that including the opaque resources allows the system to have a greater Revenue Rate. Between the independent and combined strategies the combined strategy is capable of giving a slightly better Revenue Rate. Thus, the revenue that is earned can be increased by including the AS-based broker and opaque resources.

4.12.3 Revenue Rate on a System with a Mixed Workload

As seen in Section 4.11 the performance of a system can increase when On Demand requests are introduced into the system. This section focuses on the effect that On Demand requests have on the Revenue Rate that is achieved by the system. Figure 4-36 shows the R values for both the AS-based and EDF-based broker with only 1 resource. Included in the figure are the R values for the AS-based and EDF-based broker with a workload that consists only of Advance Reservations. The workload type is denoted by

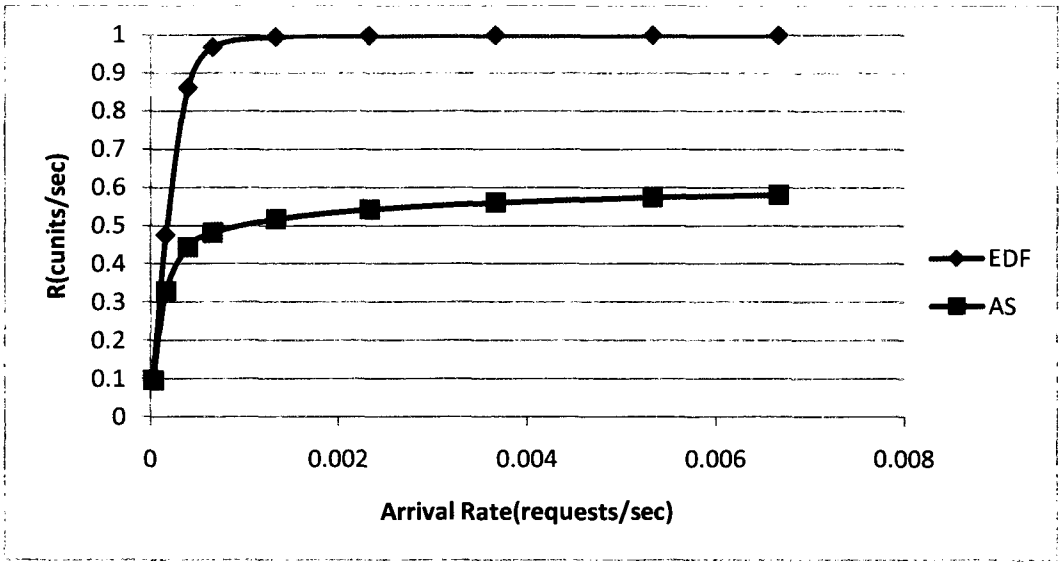

Figure 4-33: R values for EDF and AS-based brokers with 1 resource

W where MI refers to a mixed workload and AR refers to an Advance Reservation only workload. The results show that the Revenue Rate of the AS-based broker can be significantly increased through the inclusion of On Demand requests. At high arrival rates the AS-based broker is capable of matching the EDF-based broker that is subjected to either an AR only or mixed workload. This accounts for an approximate 40% increase in revenue that is earned by the AS-based broker at high arrival rates.

Figure 4-37 shows the R values for the AS-based and EDF-based broker when 10 resources are available. Again there is a significant increase in revenue for the AS-based broker when On Demand requests are present in the system. The AS-based broker is also able to match the revenue earned by the EDF-based broker on all arrival rates except for two medium arrival rates. Therefore the Revenue Rate which is attained by the AS-based broker can be significantly increased by increasing the number of resources available and including On Demand requests.

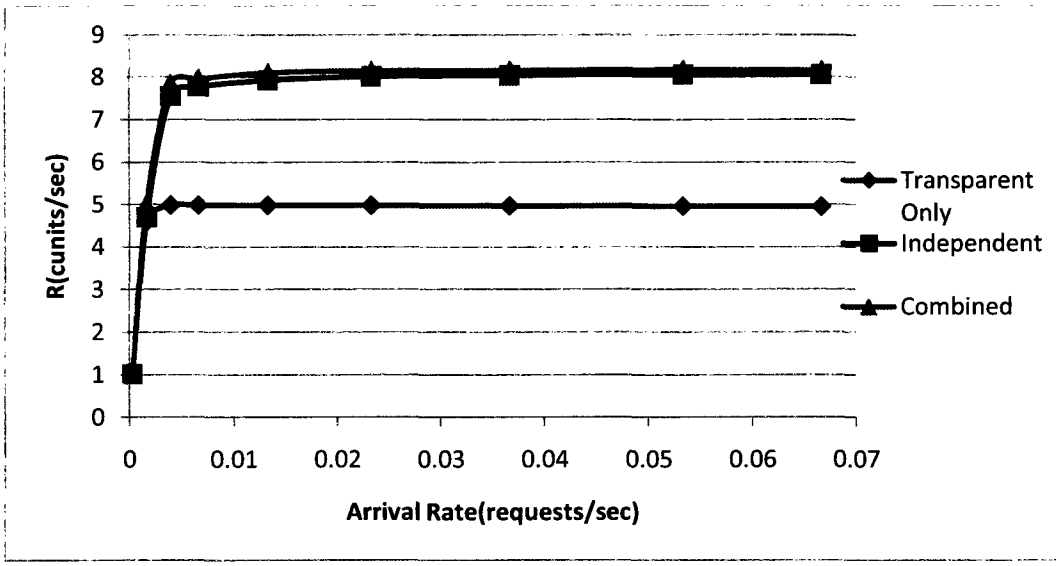

Figure 4-35: R values for different matchmaking strategies

The next chapter summarizes the work performed in this thesis as well as the conclusions drawn from this chapter.

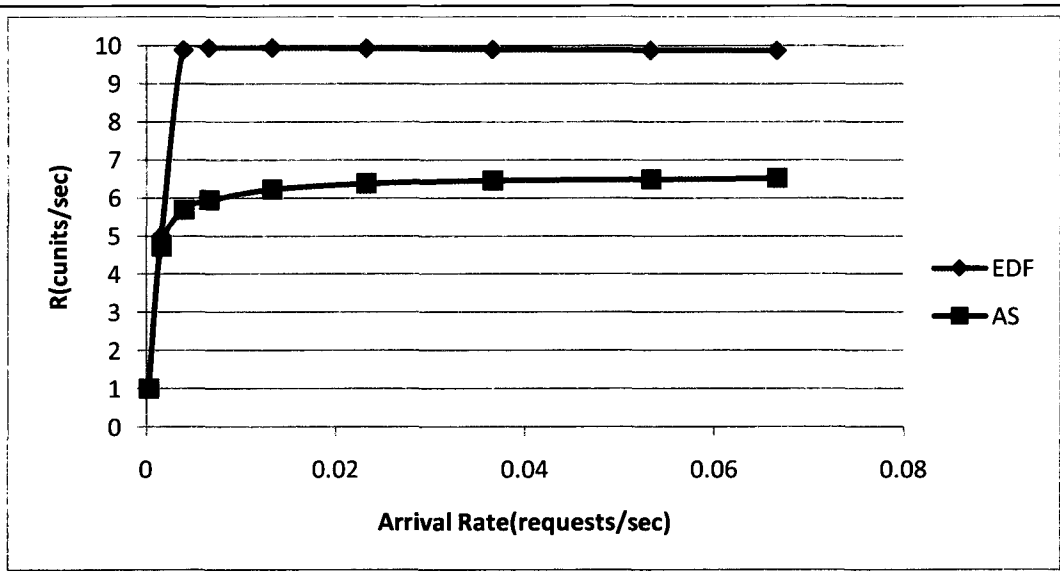

Figure 4-34: R values for EDF and AS-based brokers with 10 resources

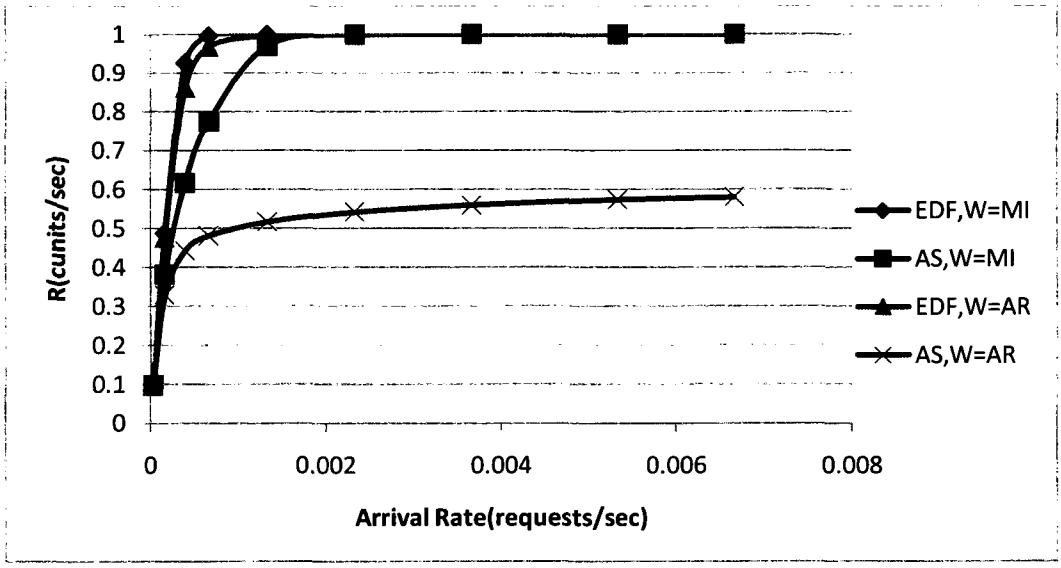

Figure 4-37: R values for EDF and AS-based brokers with 1 resource subjected to mixed and AR only workloads

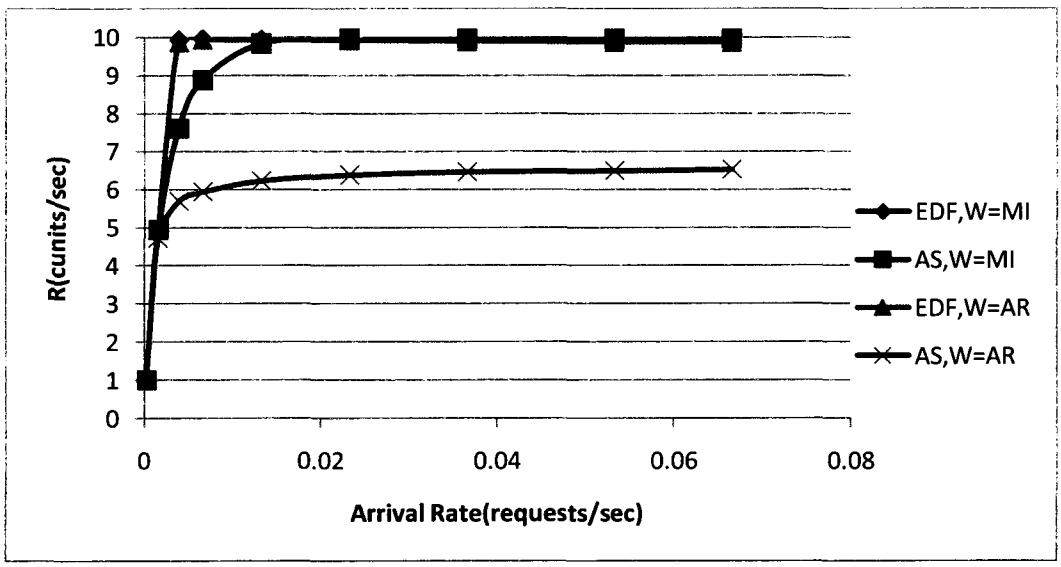

Figure 4-36: R values for EDF and AS-based brokers with 10 resources subjected to mixed and AR only workloads

Chapter 5: Conclusions

This chapter provides a summary of the research performed in this thesis as well as directions for future research.

5.1 Summary

Distributed systems such as grids and clouds consist of heterogeneous resources that may contain different hardware and software components. Heterogeneity is an important aspect of these systems as it allows various types of resources to be part of the grid or cloud environment. This can lead to difficulties for brokers that must perform resource management.

With regards to scheduling and matchmaking resources can be separated into two groups, transparent and opaque. Transparent resources are resources where the broker has *a priori* knowledge of the scheduling policy used at the resource while opaque resources are those resources where the broker does not have *a priori* knowledge of the scheduling policy used at the resource. The broker may be unaware of which scheduling policy is used at a resource for various reasons such as the uncertainty of the type of resources to be used during system design. A more detailed discussion is presented in Section 1.5. A system that is capable of incorporating both transparent and opaque resource is expected to perform better than a system that is only capable of using transparent resources.

Grids and clouds are subjected to two types of requests: On Demand requests that are to be executed on a best effort basis and Advance Reservation requests that have deadlines that must be met. The deadlines in Advance Reservation requests can be used to provide Quality of Service guarantees. The goal of the broker is to determine whether an Advance Reservation can be scheduled on a resource in the system such that it meets its deadline. If a system contains opaque resources it is difficult for the broker to determine if the request is schedulable on those resources.

The Any Schedulability criterion [MAJ-09] is used to incorporate opaque resources into the system. The Any Schedulability criterion comprises a set of inequalities that can determine whether a set of Advance Reservation requests are schedulable. The strength of the Any Schedulability criterion is that any scheduling policy can be implemented at the resource (as long as it is work conserving) and the schedulability of the requests can be determined. Incorporating the Any Schedulability criterion into the broker allows the deployment of opaque resources that would remain unutilized otherwise.

A broker that is based on the Any Schedulability criterion is devised and compared with a traditional broker that performs matchmaking using *a priori* knowledge of the scheduling policy used at the resource. When a request is schedulable on multiple resources, the broker uses an allocation algorithm to select the resource on which the request is to be executed. Two allocation algorithms that allocate the request using different criteria are presented.

The allocation algorithms investigated are First Fit and Next Fit. Two scheduling policies are experimented with: Earliest Deadline First and Earliest Start time First. Both

policies attempt to order the requests while maintaining the work conserving principle. These scheduling policies are also used at the brokers that use *a priori* knowledge of the scheduling policy used at the resources.

Two hybrid matchmaking strategies that incorporate both transparent and opaque resources are investigated. The strategies devised use a transparent broker to manage a transparent resource pool and an AS-based broker to manage an opaque resource pool. The transparent broker has *a priori* knowledge of the scheduling policy used at the resources. The two strategies are called the independent and the combined strategies. In the independent strategy both resource pools receive a proportion of the total workload arriving at the system. In the combined strategy the transparent resource pool is assigned to be the primary resource pool while the opaque resource pool is assigned to be the secondary resource pool. All newly arriving requests are sent to the primary resource pool where the transparent broker determines if the requests are schedulable on its resources. If the transparent broker determines that the request is not schedulable the request is sent to the secondary resource pool where the AS-based broker determines if it can be scheduled on its resources.

A simulator is developed to analyze the performance of the matchmaking algorithms. A grid simulation environment, GridSim [BUY-02] is used in order to implement the various grid components (brokers, resources, traffic source, and requests) in the simulator and to measure the various performance measures. The simulation system is set with various different system and workload parameters that are altered independently of one another so as to see their effects on the performance measures. The main performance

measures of interest are the Blocking Ratio (B), the Rate of Accepted Requests (A), and the Revenue Rate (R).

5.2 Insights into System Behaviour and Performance

A Summary of key observations made from the simulation results is presented.

Impact of using AS-Based matchmaking:

- The experimental results demonstrate the effectiveness of using AS-based matchmaking. The ability to perform matchmaking without having exact *a priori* knowledge of resource scheduling policies makes it possible to incorporate opaque resources that would otherwise remain unutilized within the system.
- As expected, for the same number of resources comparing the B values achieved by the AS-based broker to those achieved by the EDF-based broker shows that the AS-based broker does not perform as well as the EDF-based broker. This result is observed when both brokers have access to a single resource. However, the increase in the number of resources reduces the performance difference between the two brokers.
- In fact, the difference in B achieved by the EDF-based and the AS-based strategies for a given number of resources are observed to approach each other for very small and high values of λ.
- By incorporating additional (opaque) resources the hybrid matchmaking strategies demonstrate a higher system performance than a system using only the available

transparent resources. The benefit of incorporating the opaque resources in the resource pool translates directly to an improvement in A and R.

Impact of system and workload parameters on performance:

- The number of resources that is available to a broker has a significant effect on the performance of the system for both the AS-based and EDF-based broker. As more resources are available to the broker the system sees a decrease in B and an increase in system performance.
- The allocation algorithms used by the AS-based and the EDF-based broker did not seem to have a significant effect on system performance.
- Increasing the variability factor of the request service times leads to an improvement in system performance. This is a result of the increased number of requests with smaller service times that can be scheduled more effectively than requests with large service times.
- An increase in laxity results in an increase in system performance. An increase in laxity means that each request has a larger window for completion which allows the accommodation of more incoming requests on the system.

Effect of using a mixed workload:

- A mixed workload comprising both AR and OD requests is investigated. Comparing the performance of the AS-based broker on a system subjected to Advance Reservation requests with that of the AS-based broker handling a mixed workload shows that the addition of On Demand requests can improve performance (decrease B) significantly.

- Revenue Rate (R) achieved by an AS-based broker can be increased further by using a mixed workload consisting of both Advance Reservation and On Demand requests.

5.3 Future Research

A number of issues that warrant further investigation are presented in this section.

- Investigation of the robustness of the AS-based matchmaking strategy when there is an error associated with the user estimated job execution times forms an important direction for future research.
- Investigating the performance of AS-based matchmaking using various different distributions of request execution times needs investigation.
- Workflows are another important type of applications executed on grids and clouds. Workflows constitute multiple tasks and all the tasks are to be completed by a given deadline. Workflows are also an example of requests that can exhibit inter-dependencies. The research performed in this thesis focused on a workload comprising independent requests. Investigating the use of the AS criterion in matchmaking on systems that need to handle workflows is worthy of research.
- The performance results seen in this research are attained using a grid simulation environment. Investigating the performance of AS-based matchmaking using real workloads and real network delays is worthy of research.

References

[AMA] Amazon Elastic Compute Cloud, "Amazon EC2", http://aws.amazon.com/ec2/ , Accessed July 2010.

[AMA2] Amazon Simple Storage Service, "Amazon S3", http://aws.amazon.com/s3/ , Accessed: July 2010.

[ARM-09] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing", Technical Report No. UCB/EECS-2009-28, Dept. of Electrical and Computer Eng., University of California, Berkeley, CA, USA, February 2009.

[BER-96] F. Berman, R. Wolski, S. Figueira, J. Schopf, G. Shao, "Application-Level Scheduling on Distributed Heterogeneous Networks" in the *Proceedings of the 1996 ACM/IEEE conference on Supercomputing (CD-ROM)*, p. 39-es (article 39), DOI: 10.1145/369028.369109, Pittsburgh, Pennsylvania, USA, 1996, http://doi.acm.org/10.1145/369028.369109, Accessed: July 2009.

[BER-09] V. Berstis, "Fundamentals of Grid Computing", IBM, 2002, http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf, Accessed June 2009.

[BIR-09] G. Birkenheuer, A. Brinkman, H. Karl, "The Gain of Overbooking", in *Proceedings of the 14th International Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 80-100, Rome, Italy, May 29, 2009.

[BRA-09] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, M. Wong, "Resource monitoring and management with OVIS to enable HPC in cloud computing environments", in the *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing*, pp.1-8, Rome, Italy, May 2009.

[BUY-02] R. Buyya, M. Murshed, "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing", in *Concurrency and Computation: Practice and Experience,* 14(13-15): 1175-1220, December, 2002.

[BUY-09] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility", in *Future Generation Computer Systems,* 25(6): 599-616, June, 2009.

[CAR-06] E. Caron, F. Desprez, "DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid" in the *International Journal of High Performance Computing Applications*, 20(3):335–352, March 2006.

[CAR-09] E. Caron, F. Desprez, D. Loureiro, "Cloud Computing Resource Management through a Grid Middleware: A Case Study with DIET and Eucalyptus", in the *Proceedings of the 2009 IEEE International Conference on Cloud Computing*, pp. 151-154, Bangalore, India, September, 2009.

[CER] CERN Press Release, "LHC Computing Grid Goes Online", CERN Press Release, PR13.03, 2003,

http://public.web.cern.ch/press/pressreleases/Releases2003/PR13.03ELCG-1.html

Accessed June 2009.

[ELM-08] E. Elmroth, J. Tordsson, "A Grid Resource Broker Supporting Advance Reservations and Benchmark-Based Resource Selection", in the *Journal of Future Generation Computer Systems,* 24(6):585-593, June 2008.

[FAR-05] U. Farooq, S. Majumdar, E.W. Parsons, "Dynamic Scheduling of Lightpaths in Lambda Grids", in the *Proceedings of the 2nd International Conference on Broadband Networks*, pp. 1463-1472, Boston, Massachusetts, October, 2005,

[FOS-99] I. Foster, C. Kesselman, *The grid: blueprint for a new computing infrastructure,* San Francisco, CA: Morgan Kaufmann Publishers, 1999.

[FOS2-99] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", in the *Proceedings of the International Workshop on Quality of Service,* pp. 27-36, London, UK, May, 1999.

[FOS-01] I. Foster, "The Globus Toolkit for Grid Computing" (Tutorial), in the *Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid*, p. 2, Brisbane, Australia, May 2001.

[FOS2-01] I. Foster, C. Kesselman, S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", in the *International Journal of Supercomputer Applications,* 15(3):200-222, Fall 2001.

[FOS-02] I. Foster, "What is the Grid? – a three point checklist", in *GRIDtoday,* 1(6) 2002, http://www.gridtoday.com/02/0722/100136.html, Accessed July 2009.

[FRE-98] R.F. Freund M. Gherrity, S. Ambrosius, M. Campbell, M. Halderman, D. Hensgen, E. Keith, T. Kidd, M. Kussow, J.D. Lima, F. Mirabile, L. Moore, B. Rust, H.J. Siegel, "Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet", in the *Proceedings of the Seventh Heterogeneous Computing Workshop*, pp. 184 - 199, Orlando, Florida, USA, March, 1998.

[GOO] Google, "Google App Engine", http://code.google.com/appengine/ , Accessed July 2010.

[GUO-07] L. Guo, A. S. McGough, A. Akram, D. Colling, J. Martyniak, M. Krznaric, "Enabling QoS for Service-Oriented Workflow on Grid", in *Proceedings of the 7th IEEE international Conference on Computer and information Technology*, pp. 1077-1082, Washington, DC, USA, October, 2007.

[ITT] IT-Tude, "Types of Grid", IT-Tude, http://www.it-tude.com/types-of-grids.html , Accessed June 2009

[JAC-03] B. Jacob, "Grid computing: What are the key components? Taking advantage of grid computing for application enablement", IBM, June 2003, www.cngrid.org/download/gridtechnology/gridbook/Gridcomputing.doc , Accessed June 2009.

[KIM-09] H. Kim, S. Chaudhari, M. Parashar, C. Marty, "Online Risk Analytics on the Cloud", in the *Proceedings of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, pp. 484-489, Shanghai, China, May, 2009.

[LI-09] Q. Li, Q. Hao, L. Xiao, Z. Li, "Adaptive Management of Virtualized Resources in Cloud Computing Using Feedback Control", in the *Proceedings of the First IEEE*

International Conference on Information Science and Engineering, pp. 99-102, Nanjing, China, December, 2009.

[LIT-61] J. D. C. Little, "A Proof for the Queuing Formula: L= λ W", *Operations Research*, 9(3): 383-387, May, 1961.

[LIU] Y. Liu, "Grid Scheduling", *Technical report,* Department of Computer Science University of Iowa, www.cs.uiowa.edu/~yanliu/QE/QEreview.pdf, Accessed July 2009.

[LOC-05] N. T. Loc, S. Elnaffar, T. Katayama, H.T. Bao, "A Scheduling Method for Divisible Workload Problem in Grid Environments", *Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, pp. 513-517, Dalian, China, December 2005.

[MAH-99] M. Maheswarar, S. Ali, H. J. Siegel, D. Hensgen, R. F. Freund, "Dynamic Matching and Scheduling of a Class of Independent Tasks onto Heterogeneous Computing Systems", in *Proceedings of the Eight Heterogeneous Computing Workshop,* pp. 30-44, San Juan, Puerto Rico, April, 1999.

[MAJ-09] S. Majumdar, "The Any Schedulability criterion for Providing QoS Guarantees through Advance Reservation Requests", *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (International Workshop on Cloud Computing)*, pp. 490-495, Shanghai, China, May, 2009.

[MAN-05] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, L. Johnsson, "Scheduling Strategies for Mapping Application Workflows onto the Grid", in the *Proceedings of the IEEE International Symposium on High Performance*

Distributed Computing, pp. 125-134, Research Triangle Park, North Carolina, USA, July 2005.

[MAR-07] M. Margo, K. Yoshimoto, P. Kovatch, P. Andrews, "Impact Reservations on Production Job Scheduling", in the *Proceedings of the 13th International Workshop on Job Scheduling Strategies for parallel Processing*, pp. 116-131, WA, USA, June 2007.

[MEL-10] J. O. Melendez, S. Majumdar, "Matchmaking with Limited Knowledge of Resources on Clouds and Grids", in the *Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunication Systems,* pp. 102-110, Ottawa, Canada, July 2010.

[NUR-08] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Userful Systems", in the *Proceedings of Cloud Computing and its Applications,* Chicago, Illinois, October, 2008, http://www.cca08.org/papers/Paper32-Daniel-Nurmi.pdf, Accessed August 2010.

[PEN-09] X. Peng, H. Zhi-gang, "Relaxed resource Advance Reservation policy in grid computing", in *The Journal of China Universities of Posts and Telecommunications,* 16(2):108-113, April 2009.

[PLA] Platform Computing, "Platform LSF", http://www.platform.com/Products , Accessed July 2010.

[QU-07] C. Qu, "A Grid Advance Reservation Framework for Co-Allocation and Co-reservation Across heterogeneous Local Resource Management Systems", in the

Proceedings of the 7th International Conference on Parallel Processing and Applied Mathematics, pp. 770-779, Gdansk, Poland, September 2007.

[RAJ-09] H. Raj, R. Nathuji, A. Singh, P. England, "Resource management for isolation enhanced cloud services", in the *Proceedings of the 2009 ACM workshop on Cloud computing security*, pp.77-84, Chicago, Illinois, USA, November 2009.

[SAN] San Diego Supercomputer Center, "SDSC", http://www.sdsc.edu/, Accessed March 2010.

[SMI-00] W. Smith, I. Foster, V. Taylor, "Scheduling with Advance Reservations", in the *Proceedings of the IEEE/ACM 14thInternational Parallel and Distributed Processing Symposium,* pp.127-132, Cancun, Mexico, May 2000.

[SUL-04] A. Sulistio, R. Buyya, "A Grid Simulation Infrastructure Supporting Advance Reservation", in the *Proceedings of the 16thInternational Conference on Parallel and Distributed Computing and System*, pp. 1-7 Boston, MA, USA, November 2004.

[SUN-06] X. H. Sun, M. Wu, "Grid harvest service: a performance system of grid computing", in the *Journal of Parallel and Distributed Computing,* 66(10): 1322-1337, October 2006.

[TAK-07] A. Takefusa, H. Nakada, T. Kudoh, Y. Tanaka, "GridARS: An Advance Reservation-based Grid Co-allocation Framework for Distributed Computing and Network Resources", in the *Proceedings of the 13th International Workshop on Job Scheduling Strategies for Parallel Processing*, pp. 152-168, Seattle, WA, USA , June , 2007.

[TER] TeraGrid, "TeraGrid", https://www.teragrid.org/web/about/ Accessed February 2010.

[WEI-09] G. Wei, A. V. Vasilakos, Y. Zheng, N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services", in the *Journal of Supercomputing,* DOI: 10.1007/s11227-009-0318-1, Springer Netherlands, July, 2009, http://dx.doi.org/10.1007/s11227-009-0318-1, Accessed: August 2010.

[WOL-99] R. Wolski, N. Spring, J. Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing", in the *Journal of Future Generation Computing Systems*, 15(5-6):757-768, October, 1999.

[YOS-05] K. Yoshimoto, P. Kovatch, P. Andrews, "Co-Scheduling with User-Settable Reservations", *in Proceedings of the 1tth International Workshop on Job Scheduling Strategies for Parallel Processing*, 11th International Workshop, pp. 146-156, Cambridge, MA, USA, June, 2005.

[ZHA-06] H. Zhao, R. Sakellariou, "Advance Reservation Policies for Workflows", in the *Proceedings of* the 12th International Workshop on *Job Scheduling Strategies for Parallel Processing*, pp. 47-67, Saint-Malo, France, June, 2006.