Performance of a Cluster that Supports Resource Reservation and On-demand Access

by

Gerald Leung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2009

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Next generation data centres are expected to support both advance resource reservation and on-demand access, but the system performance for such a computing environment has not been well-investigated. A reservation request is characterized by a start time, duration, and resource requirement. Discrete event simulation is used to study the performance characteristics of reservation systems. The basic strategy is to accept a request if resources are available and reject the request otherwise. The performance metrics considered are resource utilization and blocking probability. Results showing the impact of input parameters on these performance metrics are presented. It is found that the resource utilization is quite low. Two strategies that can be used to improve the performance for advance reservation are evaluated. The first strategy allows the start time to be delayed up to some maximum value, while the second allows the possibility of non-uniform resource allocation over the duration of the reservation. Simulation results showing the performance improvements of these two strategies are presented.

Resources not used by advance reservation are used to support on-demand access. The performance metrics of interest is the mean response time. Simulation results showing the impact of resource availability and its variation over time on the mean response time are presented. These results provide valuable insights into the performance of systems with time-varying processing capacity. They can also be used to develop guidelines for the non-uniform resource allocation strategy for advance reservation in case the reserved resources are used for interactive access.

Acknowledgements

I would like to thank my supervisor, Professor Johnny W. Wong, and my thesis reviewers, Professor Peter A. Forsyth and Professor Ian McKillop.

Dedication

This is dedicated to my parents.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A data center typically consists of heterogeneous computing resources including individual servers and server clusters. It hosts a diverse set of applications; these applications may have different resource and performance requirements. Resource management in large data centers is an important consideration, especially for next generation data centres. Traditionally resources are allocated to applications on demand. In some applications, however, it may be attractive to make advance reservations for resources to be used in the future. As an example, consider an e-commerce website where a major sale event will occur at some future date/time and this event will draw a large number of users to the website. During this event, the amount of computing resources required is expected to increase significantly and it would be desirable to reserve resources in advance to ensure that the user response time is acceptable. Another example is the Virtual Computing Lab (VCL) at North Carolina State University. VCL allows users to access resource on-demand or by reservation [5]. With reservation, the user is provided with pre-configured hardware and software systems to be used at the requested time.

This thesis investigates systems that support both resource reservation and on-demand access. In general, a reservation is characterized by a start time, the

duration and the amount of resources required. On the other hand, for on-demand access, jobs are processed as soon as resources are available. Our focus is on resource management, namely, the allocation of resources to two types of services: resource reservation and on-demand access. As an example, one may wish to impose a limit on the amount of resources that could be reserved. Another example is to allow the possibility of an alternative start time when a reservation request is processed. Of interest to our investigation are the system performance seen by reservation requests and the impact of resource reservations on the performance of on-demand requests.

Our approach is to use performance modeling and simulation. In our model, the computing resources are server nodes in a cluster; these nodes are connected by a high-speed network. Performance evaluation of resource reservation systems has been investigated by various authors in the context of communication networks [9, 10, 22] where the resource considered is communication bandwidth. Emphasis is placed on the scheduling of reservation requests in order to make optimal use of the available bandwidth. The results are applicable to applications such as video-conferencing and information delivery. In contrast, our work is concerned a more in-depth investigation of resource reservation with a view of maximizing resource utilization and minimizing blocking. This includes systems that allow alternative start time of reservation and non-uniform allocation of resources. Another important aspect of our investigation is that the amount of resources available to on-demand access may change over time, depending on how much resources have been committed to reservation requests. This would have an impact on the response time seen by on-demand access. The performance of a system with time-varying server capacity has been investigated in the context of a wireless channel [18]. Our work is concerned with a more complex scenario where the resources under consideration

consist of multiple processor nodes instead of a single channel[1].

In our model, there are two classes of requests: class 1 and class 2, corresponding to resource reservation and on-demand access, respectively. These two classes share the same pool of resources. Since the demand for resources by class 1 is not uniform over time, the amount of resources available to class 2 is time-varying. The impact of such variations on the response time performance of class 2 is investigated. In addition, strategies in resource management, e.g, a limit is imposed on resources that could be reserved and the possibility of alternative start time and non-uniform resource allocation, will have an impact on the acceptance rate of class 1 and the resources availability to class 2. The performance of such strategies is also investigated.

The results from this thesis are significant because next generation data centres are expected to support both reservation and on-demand access, but the system performance for such a computing environment has not been well-investigated. The results in this thesis provide a valuable insight into system performance and can be used by data centre administrators to develop strategies for resource management.

The organization of the remainder of this thesis is as follow. Chapter 2 describes the background work in advance reservation and time-varying server capacity. Chapter 3 presents the performance model used to carry out the simulation experiments. Chapter 4 is concerned with results for the system performance seen by reservation requests. In Chapter 5, performance results for allowing reservation start times to be delayed and non-uniform resource allocation are presented. In Chapter 6, the impact of resource availability on the response time of on-demand access is discussed. Finally, Chapter 7 contains a summary of our findings and a discussion of future work.

[1]In our discussions, we will use node and server node interchangeably.

Chapter 2

Literature Review

2.1 Resource Reservation

Investigation of advance resource reservation systems have been reported in [4, 8, 9, 11, 15, 21, 22]. In these studies, a single-link model was used in which users submit requests to reserve a number of resource units over a fixed period of time in the future; the start time of the reservation is also specified. Typically, a slotted time model is used where time is organized into units called slots and the reservation is for an integer number of slots. The resource under consideration is network bandwidth. Mathematical modeling and/or simulations are used to evaluate the performance of scheduling algorithms for reservation requests. These algorithms determine whether a request can be accepted or not. If not, the request is rejected.

The impact of acknowledgement delay was investigated in [9, 11]. This is the length of time from the submission of a request to the acknowledgement of this request (accept or reject) from the system. Delaying acknowledgements would allow the system to batch reservation requests and thereby make more informed decisions on how the resources may be used more efficiently. On the other hand, if the acknowledgement delay is zero (referred to as *immediate acknowledgement*), a

better service is provided in the sense that the requester is informed of the results of his request immediately. In [9, 11], the performance of scheduling algorithms with immediate acknowledgement was compared with those that use delayed acknowledgement.

The issue of how conflicting requests are handled has also been investigated [1, 2, 9]. A straight forward approach is to schedule requests in FCFS order and reject those requests where resources are not available. This is known as a *loss* system. Another approach is to allow the requester to specify acceptable start times. This approach is known as an *alternative-start-time* system. Alternative-start-time systems have not received much attention; a study can be found in [9]. A third approach is to accept requests that have conflicts, but allows the possibility of cancelling a reservation by the requester or by the system after it has been accepted. This approach, known as overbooking, has been studied in [1, 2].

In [16, 22], priority scheduling is used to schedule reservation requests. Priorities are assigned based on duration or the size of the resource required. The assigned priorities may be represented as cost functions that are dependent on the characteristics of the requests. In [10], advance reservation systems have been compared to on-demand systems and it was found that applications such as video distribution are good candidate for advance reservation and that the ability to make advance reservation allows more requests to be serviced.

2.2 Time-Varying Resource Availability

Performance of systems with time-varying server capacity has been investigated in the context of wireless networks [17, 18, 19]. In these studies, it was found that when such variations in server capacity are ignored, results using traditional performance modeling methods tend to overestimate system performance. Simulation studies

were carried out to investigate the effect of time-varying server capacity on system performance. A performance study of a measurement-based admission control over multiple time scales can be found in [7]. Measurement data were filtered into high- and low- frequency components to study variations in resource utilization.

Chapter 3

Description of the Simulation Model

In this chapter, we describe the simulation model used in our investigation. The base model contains a single service facility with S server nodes. There are two classes of requests: class 1 and class 2, corresponding to resource reservation and on-demand access, respectively. For class 1, time is organized in fixed-length units called slots. For example, a slot may correspond to 30 seconds, 5 minutes, or 15 minutes. We also assume a reservation is always for an integer number of server nodes. A class 1 request specifies the start time (relative to the arrival time of the request), a duration that is a multiple of slots, and the number of nodes required. These parameters are modeled by probability distributions. The interarrival time of class 1 requests is assumed to be exponentially distributed with mean $1/\gamma$.

The system keeps track of the amount of resources that have been reserved in the future. When a class 1 request is processed, the system checks to see if there are enough resources to accommodate the request. If the answer is yes, the request is granted; otherwise the request is rejected. Class 1 requests are processed in first-come first-serve (FCFS) order.

Figure 3.1: Examples where requests are accepted and rejected

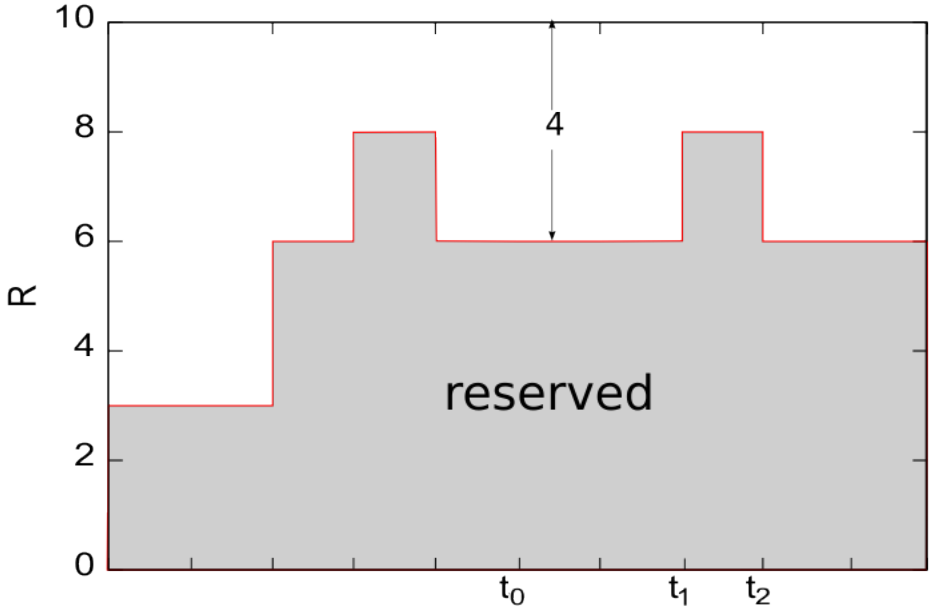


Figure 3.1 shows an example where some resources have already been committed to accepted reservation requests. In this example $S = 10$. A new reservation request for 2 nodes with a start time = t_0 and duration = t_1 - t_0 can be accepted because 4 server nodes are unreserved during this time interval. On the other hand, a request for 3 nodes with a start time = t_1 and duration = t_2 - t_1 is rejected. The reason is that only 2 nodes are available from t_1 to t_2.

As part of the resource management strategy, a limit is placed on the number of server nodes that can be reserved by class 1 at any time. We use R_{max} to denote this limit. We also require that $R_{max} < S$; this means that one or more nodes will be always be available to the on-demand class (or class 2).

At any time instants, server nodes that are not reserved are used to process class 2 jobs. We use R to denote the number of nodes reserved for class 1 ($R \leq R_{max}$). Let A be the number of nodes available to class 2. We have $A = S$ - R. The value of A varies with time because the number of reserved nodes is time-varying. The

interarrival time of class 2 jobs is assumed to be exponentially distributed with mean $1/\lambda$. Each class 2 job requires service from one server node and the service time is modeled by a probability distribution.

Service to class 2 jobs may start or end at any time instant; synchronization with time slots is not required. Class 2 jobs are serviced in FCFS order; each job requires service from one node. We assume that the system has infinite waiting room for class 2 jobs and consequently, class 2 jobs are never rejected. When a class 2 job is in execution, the node that is providing service may be required to meet a commitment to class 1. When this happens, execution of the class 2 job is suspended. Note that the number of jobs suspended may be larger than 1 depending on the number of nodes required to meet the commitment to reservation. As an example, consider the commitments shown in Figure 3.1. Suppose that at time t_0, $R = 6$ and the number of class 2 jobs in the system is much larger than 4. This means that $A = 4$ at t_0 and only 4 class 2 jobs are in execution; the other jobs are in queue. At time t_1, 8 nodes are required to meet the commitment to reservation. This means that 2 class 2 jobs must relinquish their server nodes; execution of these jobs are suspended.

Class 2 jobs that are suspended are placed at the head of the queue to preserve the FCFS order. For the suspended jobs, execution is resumed when fewer nodes are being reserved due to completion of a reservation request (e.g., at time t_2 in Figure 3.1) or a class 2 job has completed service. At resumption of job execution, the service required is given by the job's remaining service time.

In our model, we have assumed that the interarrival times of class 1 and class 2 requests are exponentially distributed. These assumptions are based on the infinite population model representing a potentially large number of users who may submit requests to the system at any time instant. We further assume that the usage of the reserved resources is not considered. This means that there is no need to define

how the class 1 requests would use the reserved resources. We also assume that for class 2 jobs, all nodes have the same processing capacity.

Chapter 4

Results for Reservation Class

In this chapter, we present simulation results for the performance seen by reservation requests (or class 1 requests). In our presentation, the following definition of terms in connection with input parameters will be used:

x = distribution of start time

d = distribution of duration

q = distribution of resource requirement

For x, d, and q, two probability distributions are considered in our simulation experiments:

uniform (y, z) - discrete uniform distribution between y and z; taking on integer values

normal (μ, σ) - normal distribution with mean μ and standard deviation σ; rounded to nearest integer and non-positive values are not used

For example x = uniform (0, 10) means that the start time is uniformly distributed between 0 and 10. For our experiments, the default values of the class 1

Table 4.1: Default class 1 input parameters

Input Parameter	Value
γ	0.5
x	uniform (0, 10)
d	normal (4, 3)
q	normal (2, 2)
S	30 nodes

input parameters are shown in 4.1. Another possibility to ensure positive values for a normal distribution is to use a log-normal distribution, which is not considered in our experiments.

We are not aware of any existing systems for which distributions can be found for the input parameters. We have chosen these distributions based on what appears intuitive. Start time is chosen to be uniform because we assume the start times for different requests have no common pattern. The duration and resource requirement are chosen to be normal because we assume there are few requests that are large and few requests that are small. The performance metrics of interest are:

E[R] - the mean number of nodes reserved

B - blocking probability (or probability that a reservation request is rejected)

In order to get reliable steady state results, we performed 10 experiments and determined a length of simulation run using the criteria that with 10 replications, the width of the 99% confidence interval of E[R] is within 2.5% of the sample mean and B is within 0.1% of the sample mean. Our results show that the above criteria is met with a length of run of 300,000 time units.

There are several ways to determine the arrivals of events that follow Poisson-like distributions in simulations. One way is to determine the number of Poisson

events is each small time step. Another way is to determine the time to the next event from the current event. We have chosen the latter method for the arrival of reservation requests.

4.1 Effect of S on E[R] and B

In our first set of experiments, we consider five different values of S (S = 10, 20, 30, 40, and 50) and investigate the effect of R_{max} on E[R] and B. The values of R_{max} are selected to be $0.5S$, $0.55S$, $0.6S$, ..., and $0.9S$. This range corresponds to scenarios where a maximum of 50% to 90% of the nodes can be reserved. As to the other parameters, the default values shown in Table 4.1 are used.

The results for E[R] and B are shown in Figures 4.1 and 4.2, respectively. It can be seen that increasing R_{max} leads to an increase in E[R] and a decrease in B. When S is increased from 10 to 50, E[R] is bounded by a value between 6.5 and 7. This effect is caused by the acceptance of nearly all class 1 requests and the value of E[R] is given by the mean resource requirement of class 1 requests over time. The above remark is confirmed by the results in Figure 4.2 where B approaches zero when S is large.

A larger arrival rate γ ($\gamma = 1.5$) is considered next and the corresponding results for E[R] and B are shown in Figures 4.3 and 4.4, respectively. We observe that with a larger γ, B approaches zero and E[R] approaches its maximum when S = 50. The value of this maximum is now approximately 22 instead of between 6.5 and 7. This increase is due to the higher arrival rate which results in higher resource demand over time.

A very large arrival rate γ ($\gamma = 2.5$) is considered next and the corresponding results for E[R] and B are shown in Figures 4.5 and 4.6, respectively. We observe

Figure 4.1: E[R] plotted against R_{max} for varying values of S for $\gamma = 0.5$

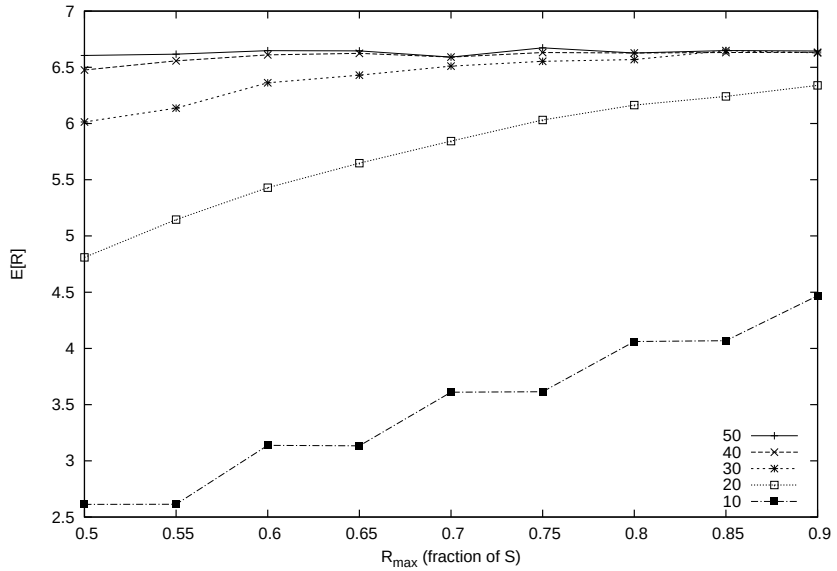


Figure 4.2: B plotted against R_{max} for varying values of S for γ=0.5

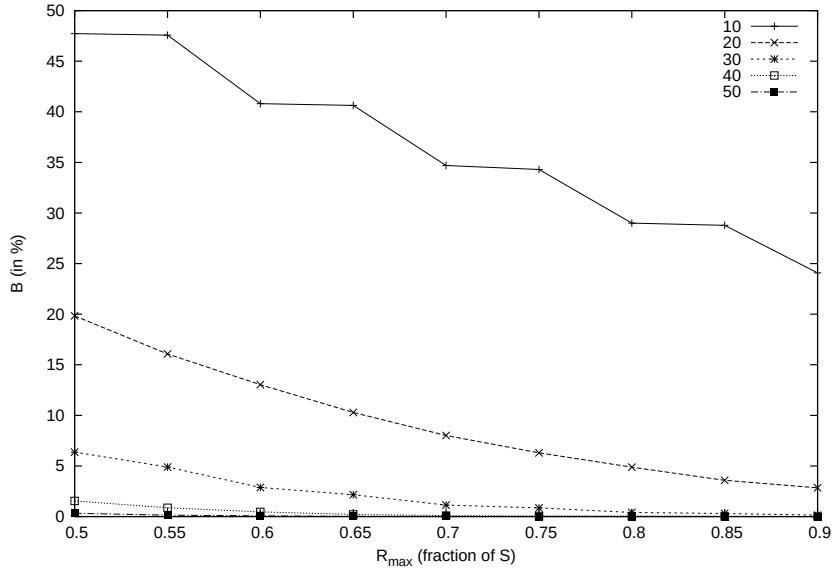


Figure 4.3: E[R] plotted against R_{max} for varying values of S for $\gamma = 1.5$

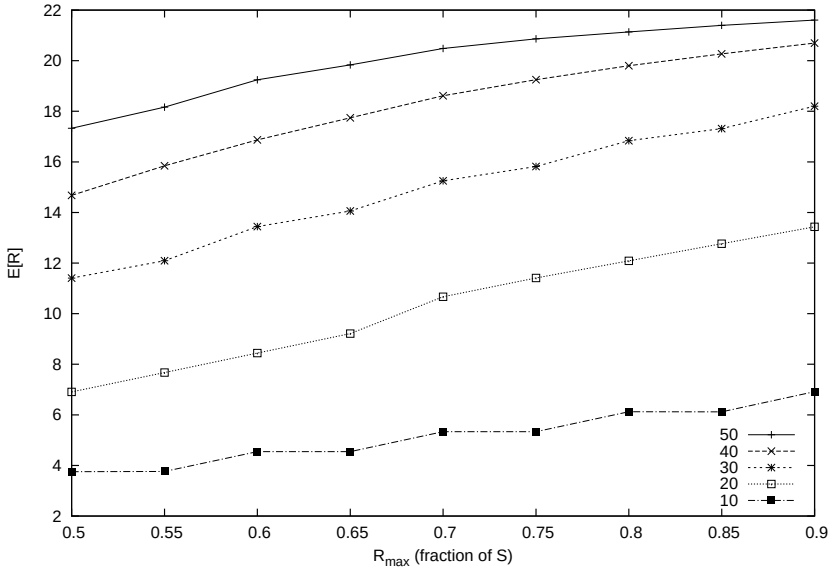


Figure 4.4: B plotted against R_{max} for varying values of S for $\gamma = 1.5$

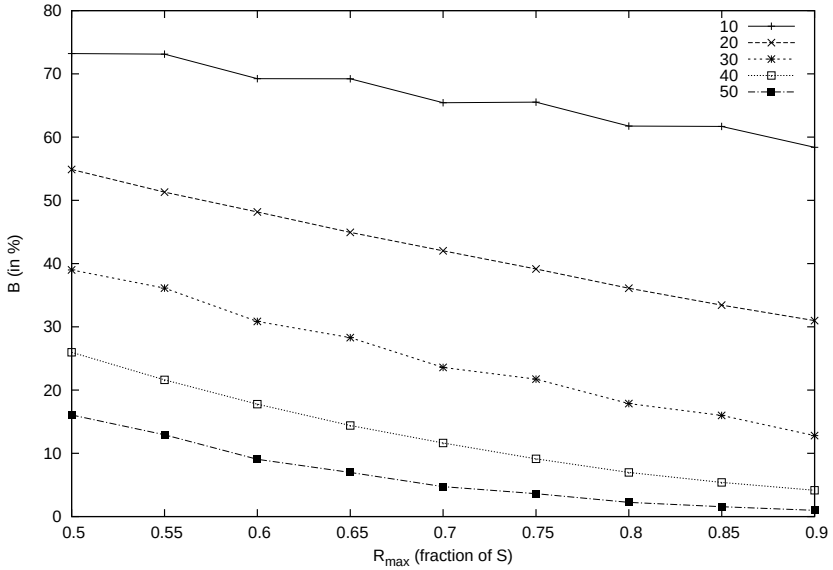
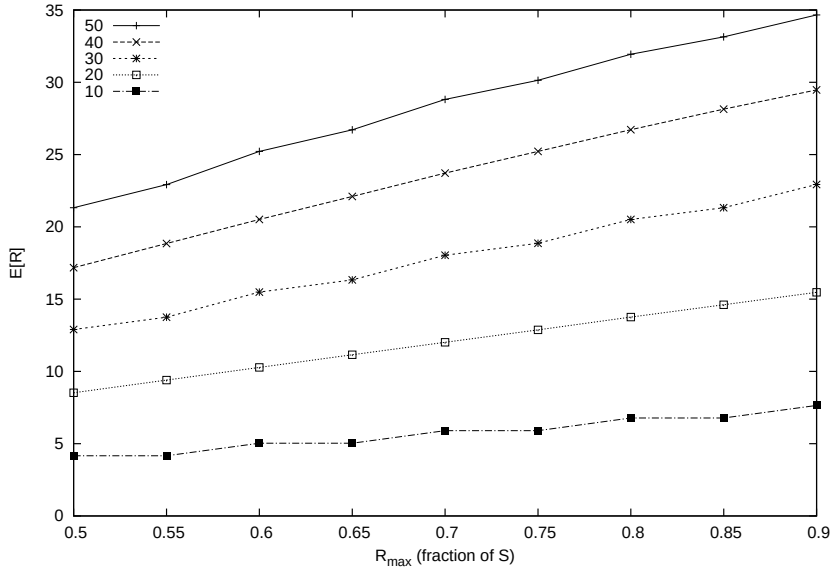


Figure 4.5: E[R] plotted against R_{max} for varying values of S for $\gamma = 2.5$



that as R_{max} increases, B does not become zero and E[R] does not reach its maximum value. In fact, over 10% of the requests are rejected even when at R_{max} = 0.9S. This large value of B indicates that the system does not have sufficient capacity to handle the load generated by reservation requests.

To provide further insight into the impact of input parameters on E[R] and B, a summary of the results in Figures 4.1 to 4.6 are shown in Figures 4.7 and 4.8 where E[R]/R_{max} and B are plotted against R_{max}. We observe that as R_{max} increases, the fraction of available resources that are reserved, given by E[R]/R_{max}, increases at first and then decreases when R_{max} is large. This latter behaviour is because of the system has sufficient capacity to handle the load. Also at large R_{max}, the value of E[R]/R_{max} is affected by γ, but not sensitive to S. As to the blocking probability B, we observe from the results in Figure 4.8 that B decreases with R_{max} and the value of B is again affected by γ and not S.

Figure 4.6: B plotted against R_{max} for varying values of S for $\gamma = 2.5$

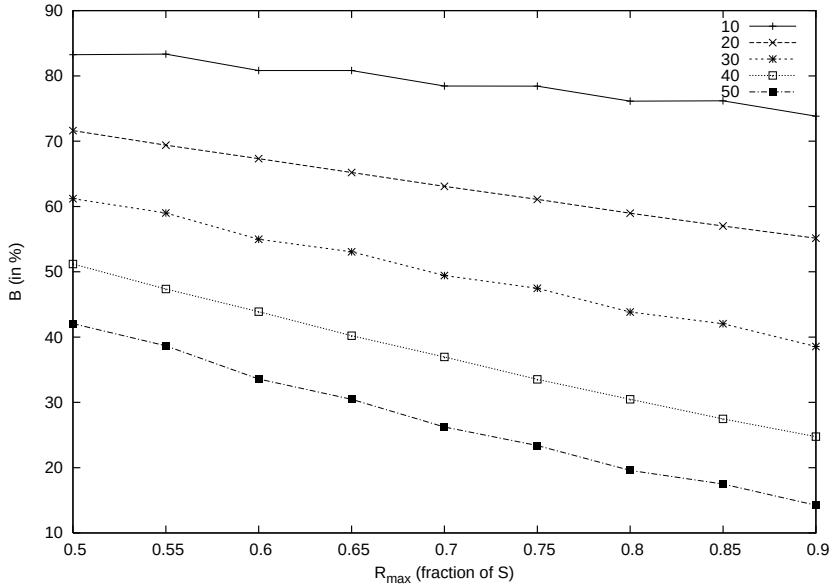


Figure 4.7: E$[R]/R_{max}$ plotted against R_{max} for varying values of S and γ

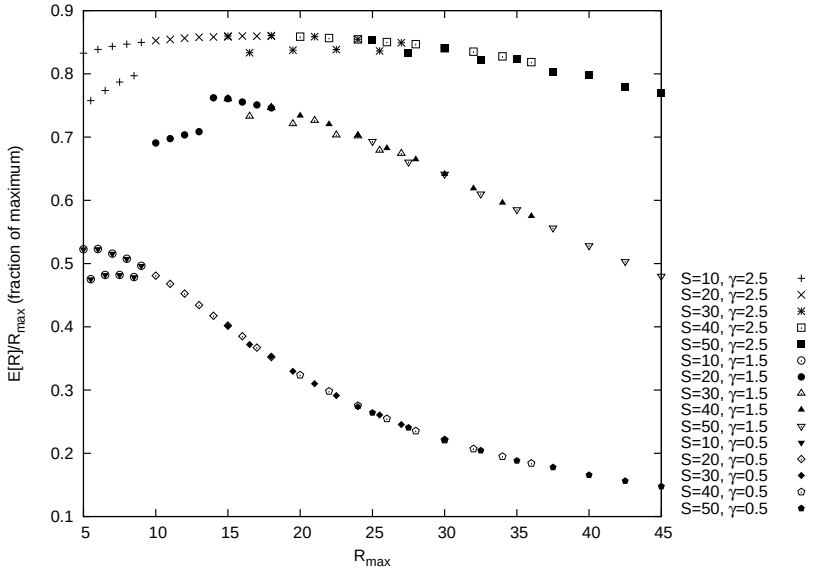
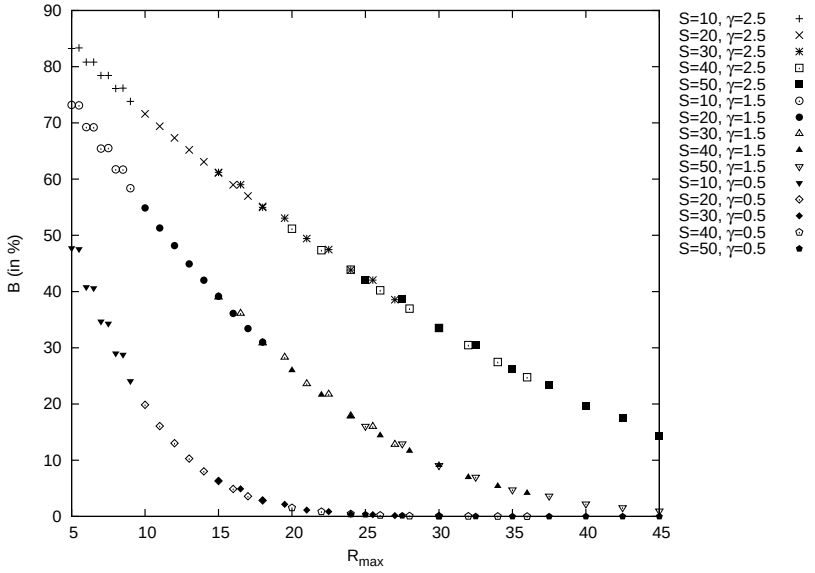


Figure 4.8: B plotted against R_{max} for varying values of S and γ



4.2 Effect of d, q, and x on E[R] and B

In our next set of experiments, we keep S at the default value of 30 and investigate the impact of d, q, and x on E[R] and B. Similar to the results in Figures 4.1 to 4.4, the values of R_{max} considered are 15, 16.5, ..., 27. The results for E[R] and B for different distributions for d are shown in Figures 4.9 and 4.10, respectively. We observe that, for a given value of R_{max}, increasing the mean duration results in an increase in both E[R] and B. A longer duration means higher resource requirement so it is not surprising to see an increase in E[R]. A longer duration also means more resources will be reserved over time which would lead to higher blocking probability for new requests. Similar increase is observed when the standard deviation of the duration is larger. A larger standard deviation means more requests with large service requirements. This explains the increase in E[R]. The increase in E[R] also causes more reservation requests to be rejected.

Figure 4.9: E[R] plotted against R_{max} for varying d

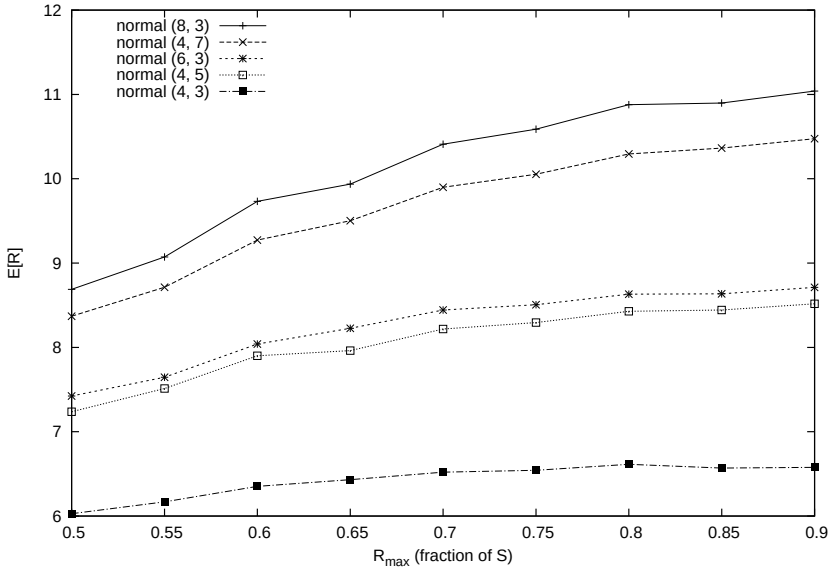


Figure 4.10: B plotted against R_{max} for varying d

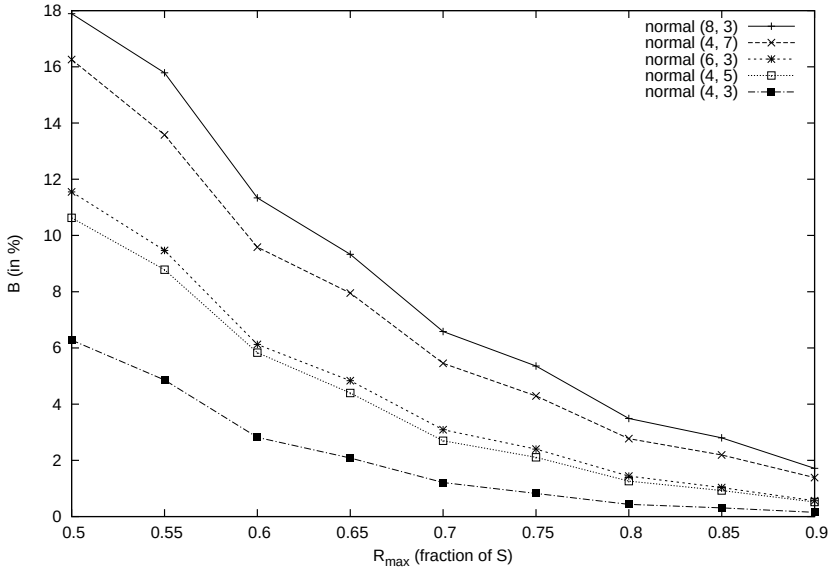
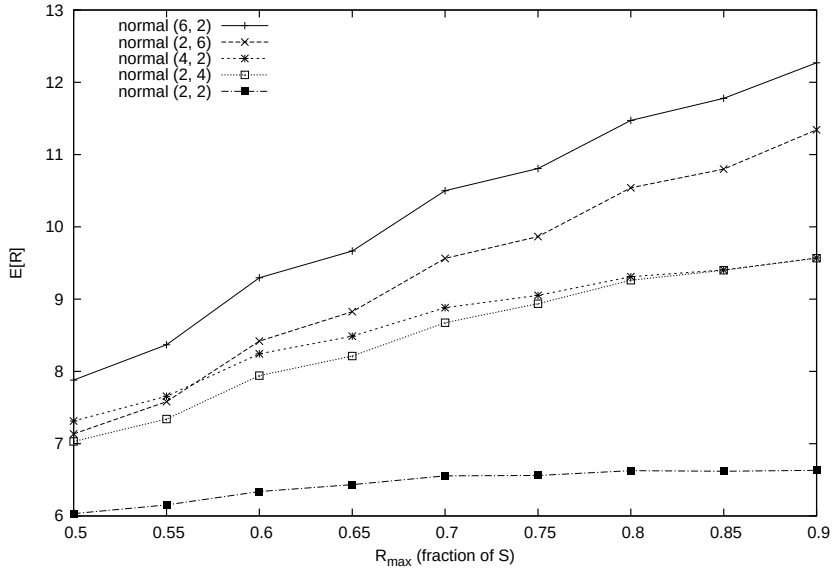


Figure 4.11: E[R] plotted against R_{max} for varying q



The results for E[R] and B for different distributions for q are shown in Figures 4.11 and 4.12, respectively. We observe that, for a given value of R_{max}, increasing the mean resource requirement leads to an increase in both E[R] and B. The explanation is similar to that for the case when we have different distributions for d.

The results for E[R] and B for different distributions for x are shown in Figures 4.13 and 4.14, respectively. We observe that, for a given value of R_{max}, the distribution of start time does not have a significant impact E[R] and B. The reason is that the start time is not related to resource usage, so it is not surprising to see that E[R] and B are insensitive to the start time.

The results for E[R] and B presented in this Chapter are consistent with those reported in [9]. These results, however, provide additional insight into the impact of resource availability and input parameters such as duration and resource

Figure 4.12: B plotted against R_{max} for varying q

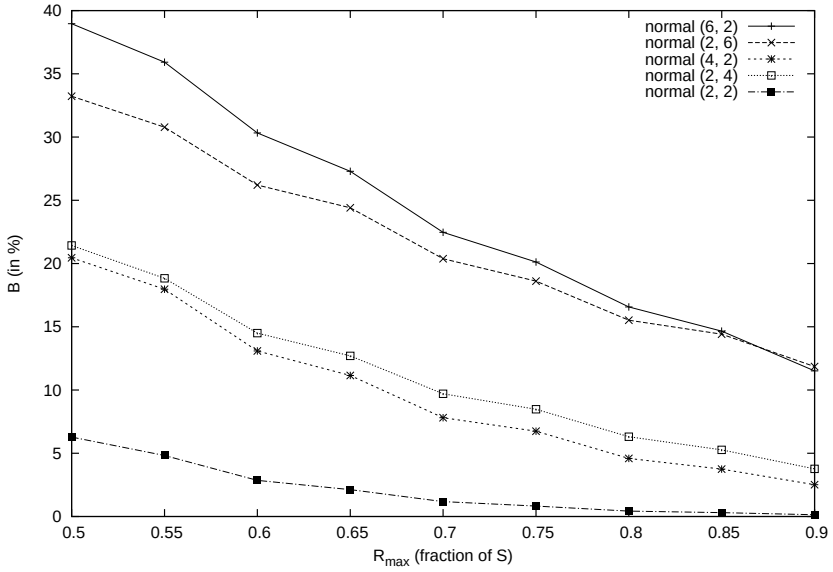


Figure 4.13: E$[R]$ plotted against R_{max} for varying x

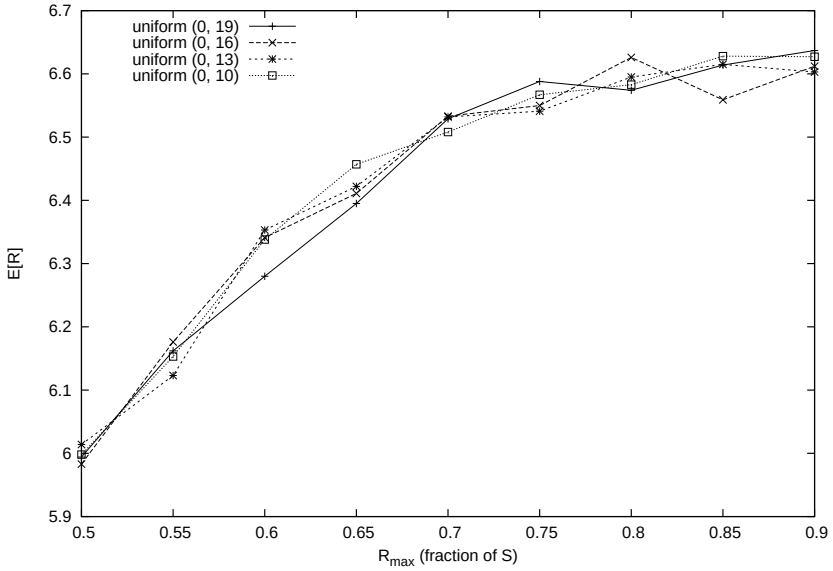
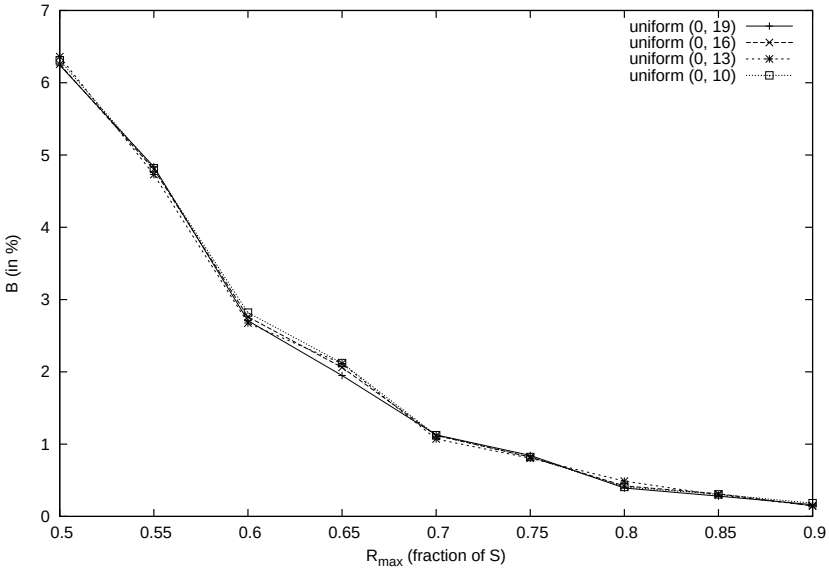


Figure 4.14: B plotted against R_{max} for varying x



requirement on performance.

Chapter 5

Performance Improvement for Reservation Class

We observe from the results in Chapter 4 that the blocking probability is large for quite a few cases. We also observe that the amount of reserved resources is small compared to R_{max}, the maximum amount allowed. For example, the results in Figures 4.9 and 4.10 show that when $R_{max} = 0.5S$ (or 15) and d is normal (6, 3), E[R] and B are approximately 7.4 and 11.8%, respectively. Of interest to our investigation are strategies that can be used to improve the values of both B and E[R] such that better utilization of resources is achieved. Two such strategies are presented in this chapter: use of start period and non-uniform resource allocation. The performance improvements resulting from these strategies are evaluated by simulation.

5.1 Use of Start Period

For the first strategy, the start time of a reservation request can be delayed (to some maximum value). If resources are not available when this maximum is reached, the

request is rejected. The period during which start times are acceptable is referred to as the start period. The rationale for using a start period is as follows. If the start time of a reservation request can be delayed, the chance of finding a start time that has sufficient resources could be improved. This should result in improved performance for the reservation class. An important assumption for the start period strategy is that the requester is willing to accept an alternative start time if the resources at the requested start time are not available.

Our base model in Chapter 3 is extended to include start period. The length of the start period is modeled by a probability distribution and we use b to denote this distribution. We further assume that b is uniform (0, z), a discrete uniform distribution between 0 and z, taking on integer values only.

We now present simulation results to show the performance improvement resulting from the use of start period. In our first set of experiments, three distributions for b are considered, namely uniform (0, 5), uniform (0, 10) and uniform (0, 20). The means of these distributions are 2.5, 5, and 10, respectively. Two scenarios are evaluated. For the first scenario, the input parameters are given by the default values in Table 4.1. The results for E[R] and B are shown in Figures 5.1 and 5.2, respectively. Results for the case where no start period is used, taken from Figures 4.9 and 4.10, are also shown. We observe improvements in E[R] and B for all values of R_{max}. The amount of improvement increases with the mean of b. This is not surprising because a larger mean should lead to a higher probability of accommodating a new request.

For the second scenario, d is normal (6, 3) instead of the default values of normal (4, 3). As to the other parameters, the default values in Table 4.1 are used. The results for E[R] and B for the three distributions for b are shown in Figures 5.3 and 5.4. Again, results for the case of no start period are also shown. We observe the same effects as for the case where the default value of d was used.

Figure 5.1: E[R] plotted against R_{max} for varying b with default d

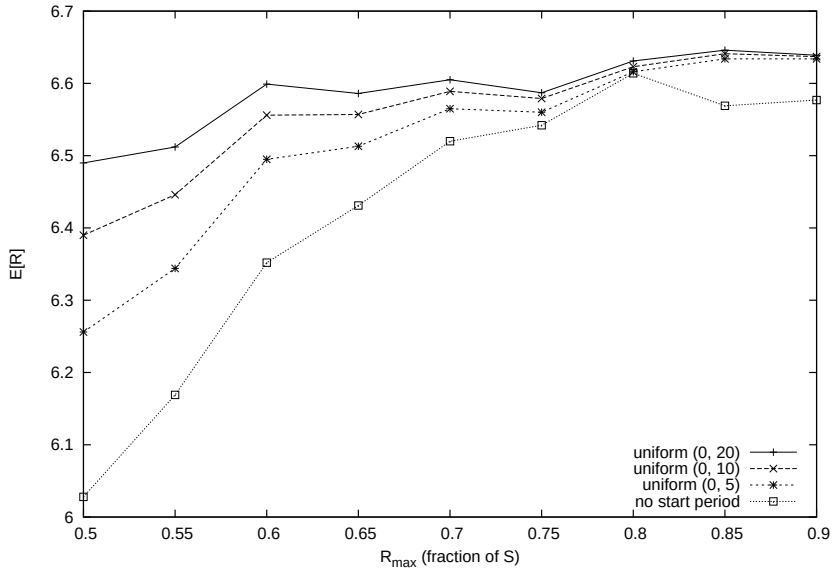


Figure 5.2: B plotted against R_{max} for varying b with default d

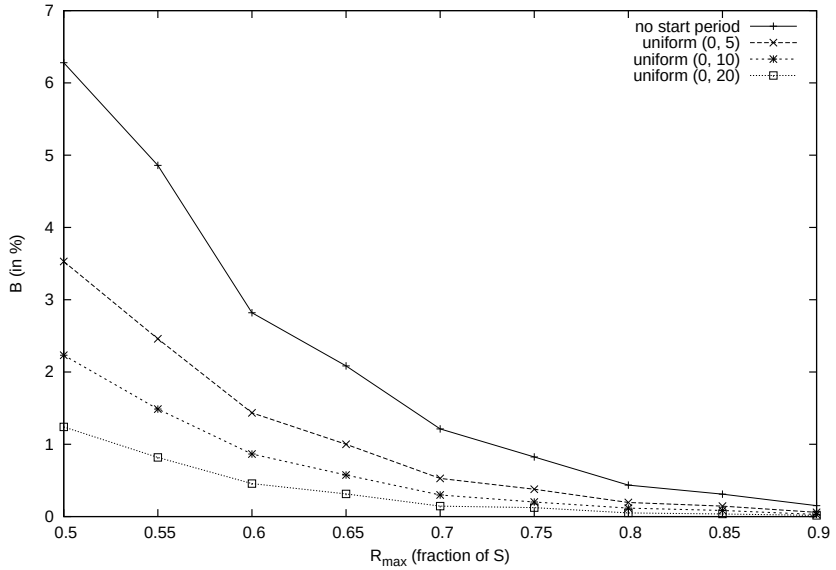


Figure 5.3: E[R] plotted against R_{max} for varying b when d is normal (6, 3)

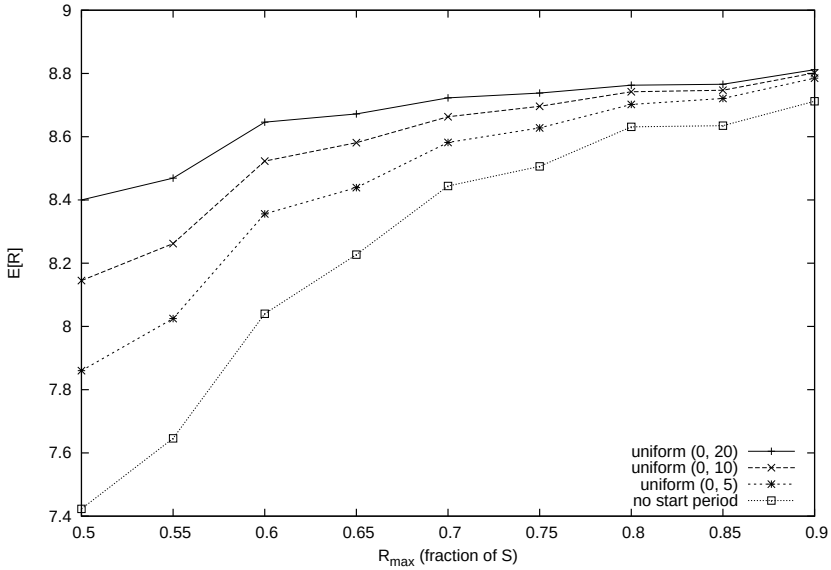
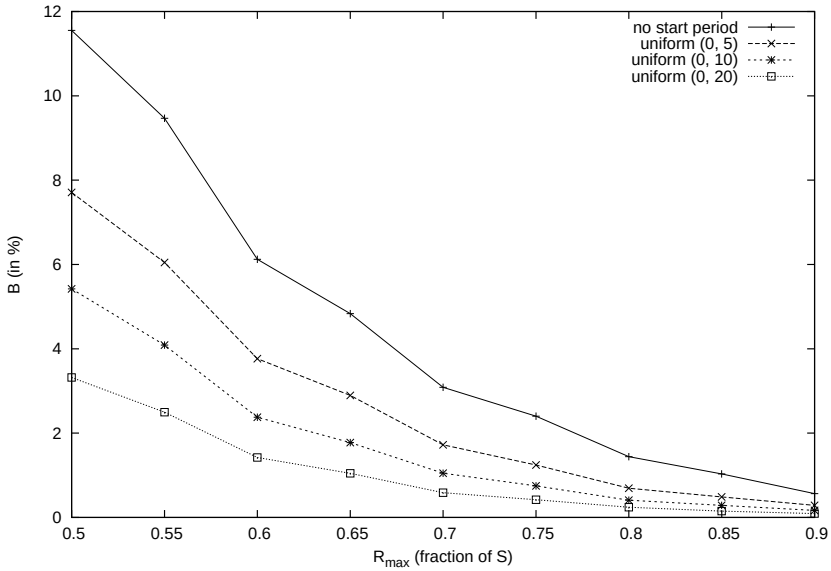


Figure 5.4: B plotted against R_{max} for varying b when d is normal (6, 3)



In our next set of experiments, we investigate the impact of the length of the start period on improvements in E[R] and B. The metrics used in our evaluation are f_R and f_B, the percentage improvements in E[R] and B, respectively. f_R and f_B are defined as follows.

$$f_R = \frac{(E[R]\_with\_start\_period - E[R]\_with\_no\_start\_period) * 100\%}{E[R]\_with\_no\_start\_period}$$

$$f_B = \frac{(B\_with\_start\_period - B\_with\_no\_start\_period) * 100\%}{B\_with\_no\_start\_period}$$

We note from the results in Figures 5.1 to 5.4 that significant improvements in E[R] and B are possible when $R_{max} = 0.5S$, but smaller improvement are realized when $R_{max} = 0.75S$. These two values of R_{max} are used in our evaluation. In Figures 5.5 and 5.6, we plot the percentage improvements f_R and f_B as a function of the mean start period for two different distributions for d, namely, normal (4, 3) and normal (6, 3). We observe that improvements in E[R] and B are larger when R_{max} is smaller. This is consistent with the results in Figures 5.1 to 5.4. We also observe that the amount of improvement is more significant when the mean duration is larger.

The corresponding results for two different distributions for q (normal (2, 2) and normal (4, 2)) are shown in Figures 5.7 and 5.8. We again observe that improvements in E[R] and B are larger when R_{max} is smaller. Similar observations are made in Figures 5.9 and 5.10 where we show the results for f_R and f_B for two different distributions for x. These start time distributions are uniform (0, 10) and uniform (0, 16), respectively.

The results in this section confirm that the strategy of start period can lead to improvements in E[R] and B. This improvement is gained at the expense of the requester having to accept an alternative start time.

Figure 5.5: f_R plotted against mean start period for varying d and R_{max}

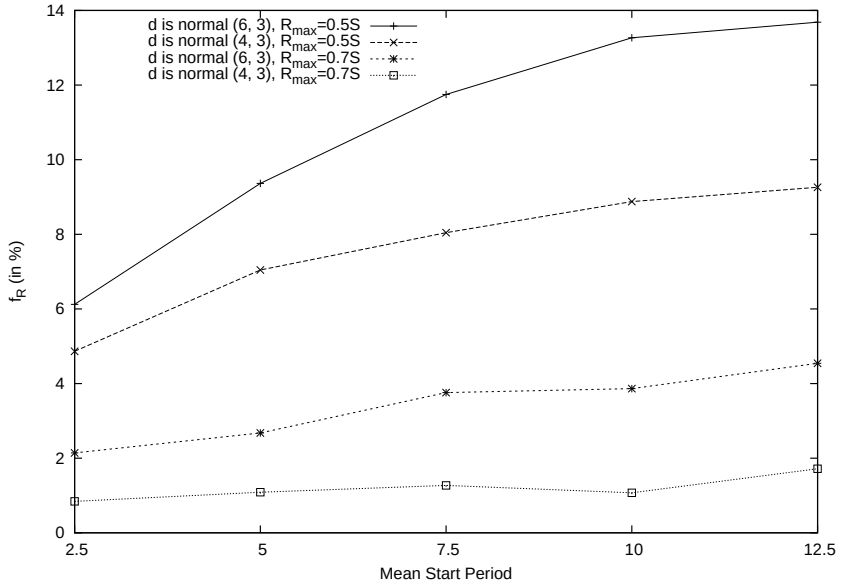


Figure 5.6: f_B plotted against mean start period for varying d and R_{max}

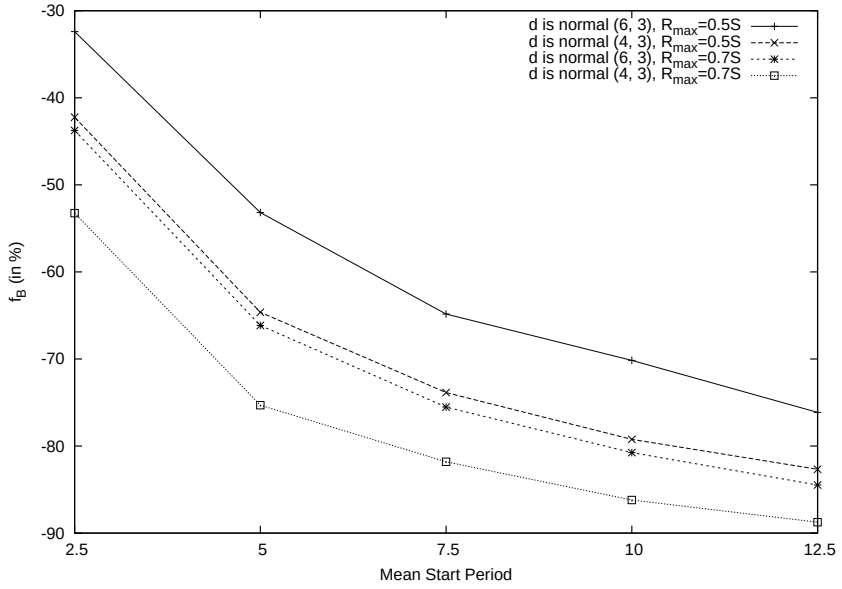


Figure 5.7: f_R plotted against mean start period for varying q and R_{max}

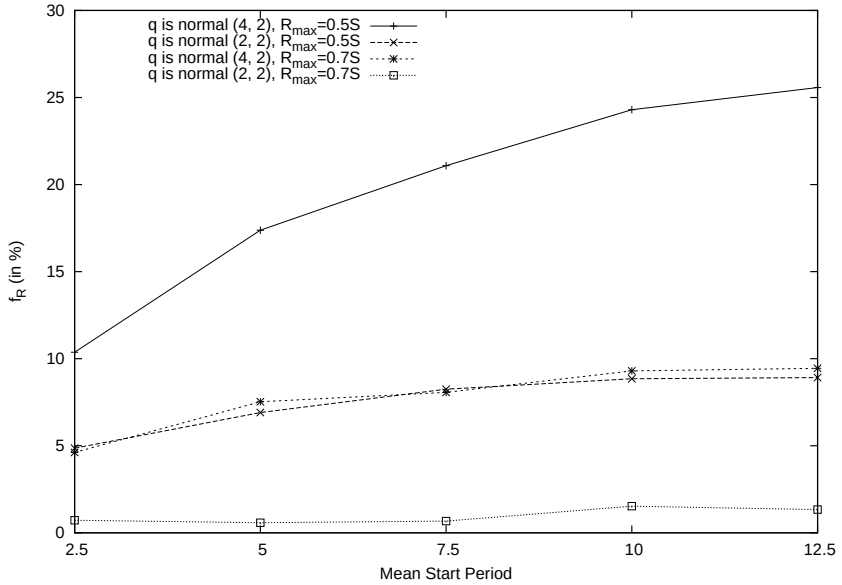


Figure 5.8: f_B plotted against mean start period for varying q and R_{max}

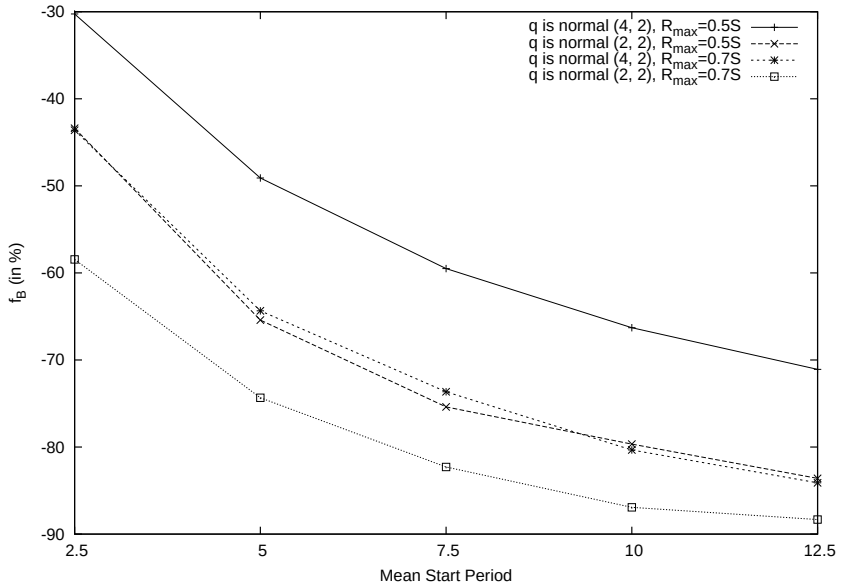


Figure 5.9: f_R plotted against mean start period for varying x and R_{max}

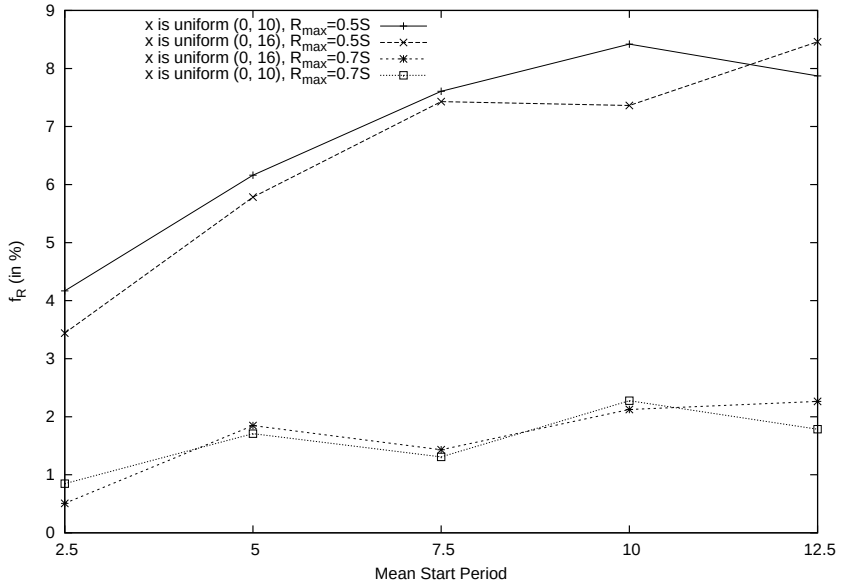
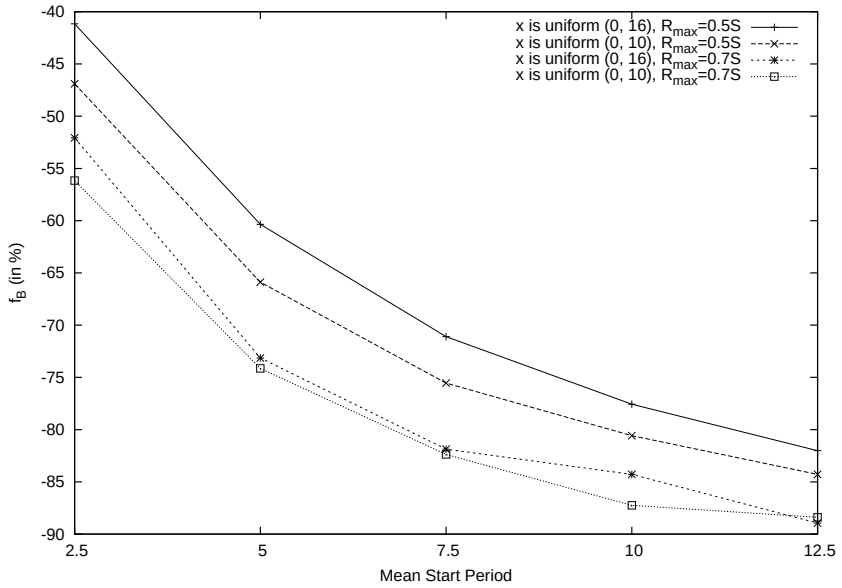


Figure 5.10: f_B plotted against mean start period for varying x and R_{max}



5.2 Non-uniform Resource Allocation

In this section, we consider our second strategy to improve E[R] and B. This strategy is based on the concept that the amount of resources allocated is not uniform over time, but the average amount is the same as that requested. The rationale for using such a concept is as follows. If the system has the flexibility of allocating resources to the reservation class in a non-uniform manner, a request can be accommodated even though the requested amount is not available at some time instants within the duration requested. This should result in improved performance for the reservation class.

Our base model in Chapter 3 is extended to include non-uniform resource allocation to a reservation request. For such a request, let u be number of server nodes required, t_0 be the start time and t_1 - t_0 be the duration. Let *G(t)* be the number of server nodes available at time t, $t_0 \leq t \leq t_1$. Our algorithm to accept or reject request is described in Algorithm 1 below. The variables used in this algorithm are defined as follows:

M - remaining resource requirements in (number of server node) * time

L - set of slots in (t_0, t_1) where server nodes are still available for allocation

N - size of L in number of slots

E - the average remaining resource requirement over the slots in L

Algorithm 1

1. If there are any time instants t within t_0 and t_1 where $G(t) = 0$, reject the request.
2. If the total available resources from t_0 to t_1 is less than (t_1 - t_0)*u, reject the request.

3. Set M to (t_1 - t_0)*u, N to number of slots in t_1-t_0, and E to M/N.
4. Allocate all available server nodes in those slots where $G(t) \leq E$ and allocate floor(E) nodes in slots where $G(t) > E$.
5. $M = M$ - amount of resources allocated in Step 4.
6. If $M = 0$, done.
7. Update $G(t)$, L and N, and re-compute $E = M/N$.
8. If $E \geq 1$, go to Step 4; otherwise allocate one server node for the first M slots in L.

To illustrate this algorithm, the shaded area in Figure 5.11 represents the resources available for reservation. A reservation request with $u = 3$ $t_0 = 2$ and $t_1 = 6$ is handled as follows. At step 2, the total available resources from t_0 to t_1 is 13 which is larger than $(t_1 - t_0) * u = 12$. Step 3 sets the variables M, L and E to their respective initial values. E is given by 3. At step 4, 2 server nodes in slots 3 and 4 are allocated since $G(t) = 2$ during these slots, and 3 server nodes in slots 2 and 5 are allocated because floor(E) = 3. This is shown as area 1 in Figure 5.12. At steps 5, M is reduced to 2. As a results of the updates at Step 7, $G(t) > 0$ in slots 2 and 5 only, $N = 2$ and $E = 1$. Returning to Step 4, one server node is allocated in slots 2 and 5. This is shown as area 2 in Figure 5.12. M then becomes zero and the algorithm terminates.

Figure 5.13 shows another example starting with the same resources available as shown in Figure 5.11, but the request is for 3 server nodes ($u = 3$) starting at $t_0 = 2$ and duration = 7. The algorithm works similarly as in the previous example, where the resources allocated at each iteration are shown.

We now present results that show the performance advantage of the non-uniform resource allocation strategy. In Figures 5.14 and 5.15, we plot the percentage improvements f_R and f_B for two different distributions for d, namely, normal (4, 3)

Figure 5.11: Scenario used for non-uniform resource allocation

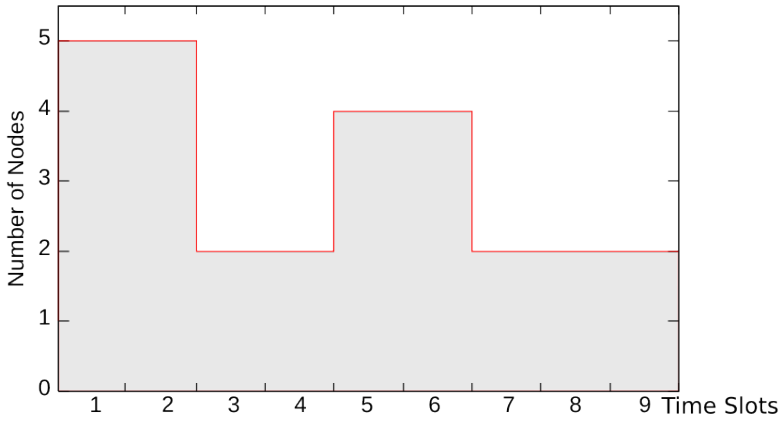


Figure 5.12: First example of non-uniform resource allocation

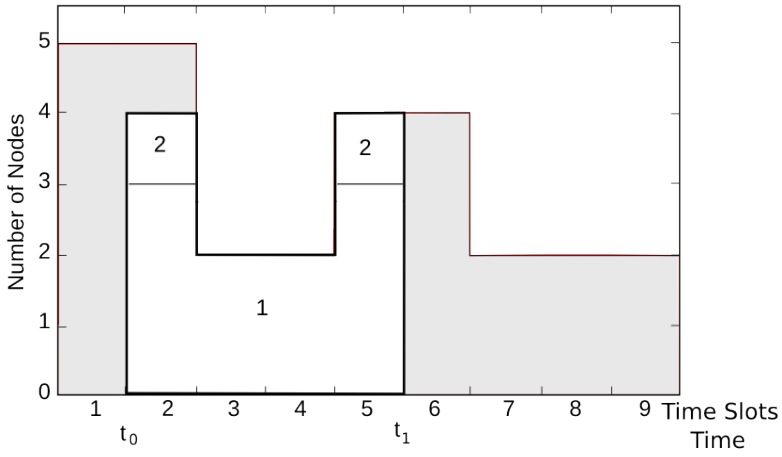
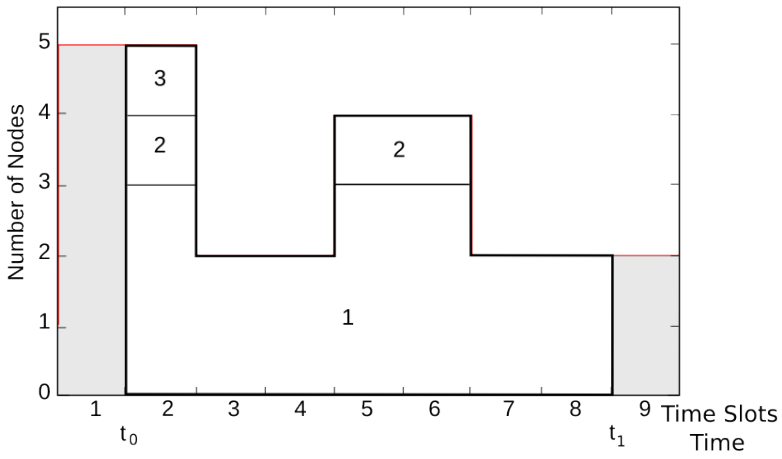


Figure 5.13: Second example of non-uniform resource allocation

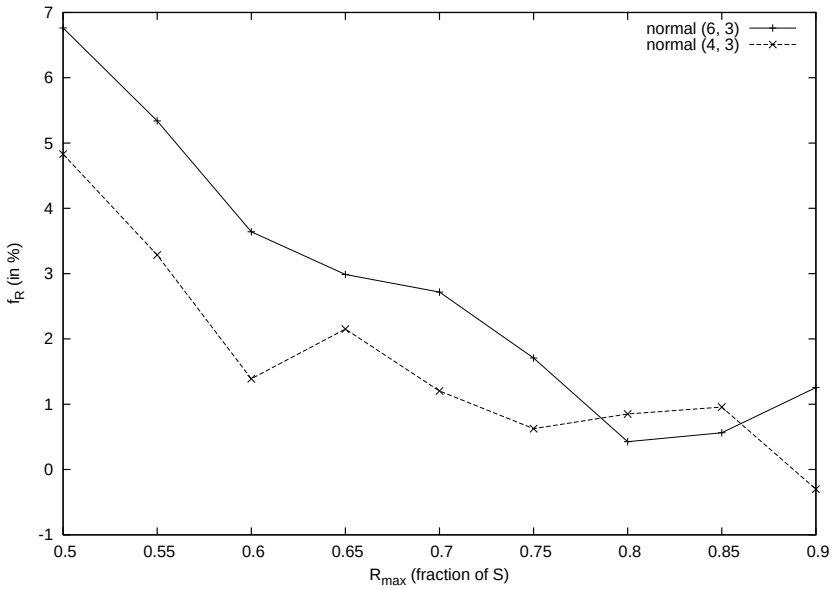


and normal (6, 3). As to the other input parameters, the default values in Table 4.1 are used. We observe improvements in E[R] and B, and the amount of improvement is larger when R_{max} is smaller. We also observe that amount of improvement is more significant when the mean duration is longer. The corresponding results for two different distributions for q (normal (2, 2) and normal (4, 2)) are shown in Figures 5.7 and 5.8. We again observe that improvements in E[R] and B and the amount of improvement are larger when R_{max} is smaller.

Finally, we show in Figures 5.18 and 5.19 the results for f_R and f_B for two different distributions for x. These distributions are uniform (0, 10) and uniform (0, 16), respectively. Contrary to the cases of varying d or q, no improvements in E[R] and B are observed for the entire range of R_{max} considered. This is consistent with the results in 4.13 and 4.14 where the start time distribution does not have a significant impact on the performance seen by the reservation class.

The results in this section confirm that the strategy of non-uniform resource allocation can lead to improvements in E[R] and B. The amount of improvement is smaller than that for the start period strategy. However, non-uniform resource allocation does not require the use of an alternative start time. This represents a

Figure 5.14: f_R plotted against R_{max} for varying d



better service offering in terms of not making changes to the requested start time. Additional results on the performance of the two strategies will be presented in the next section.

5.3 Performance Evaluation

The results in Sections 5.1 and 5.2 indicate that improvements in E[R] and B are possible if the following strategies are used: start period or non-uniform server node allocation. In this section, we discuss, by means of examples, the performance difference of these two strategies as well as the case where both strategies are not used. We will refer to this case as the basic strategy, which was investigated in Chapter 4. For the start period strategy, we assume that b is uniform (0, 10).

We start with scenarios where the blocking probability B is not larger than

Figure 5.15: f_B plotted against R_{max} for varying d

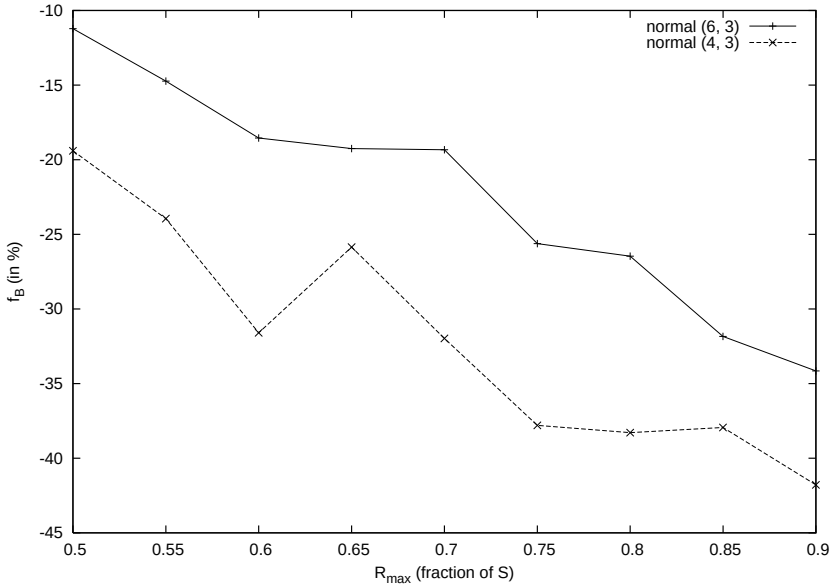


Figure 5.16: f_R plotted against R_{max} for varying q

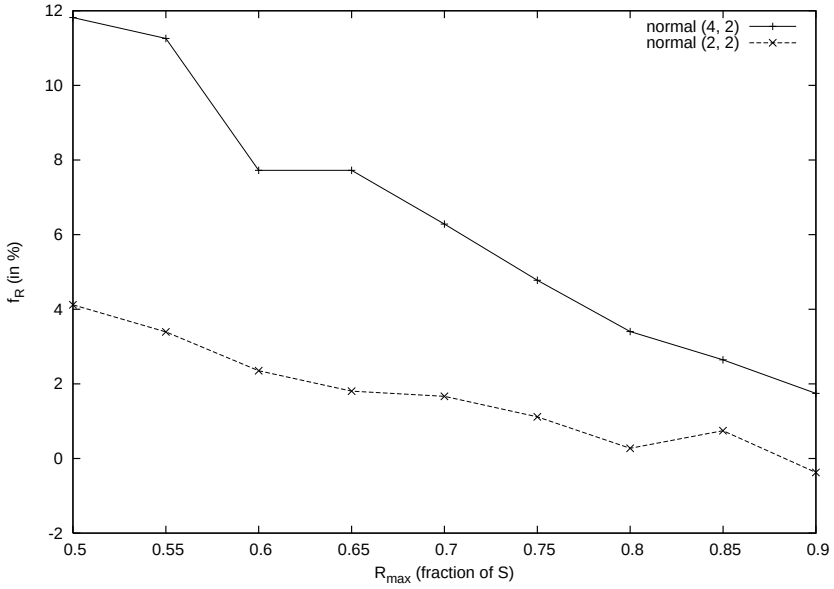


Figure 5.17: f_B plotted against R_{max} for varying q

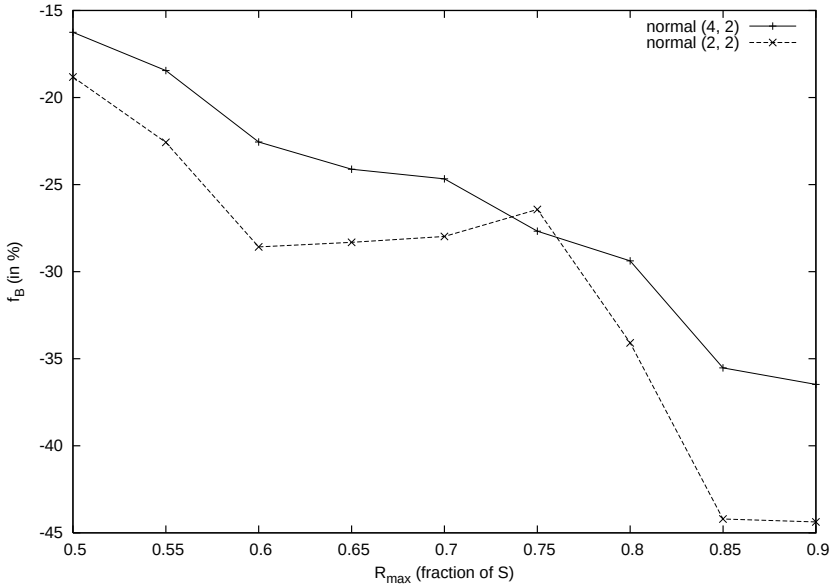


Figure 5.18: f_R plotted against R_{max} for varying x

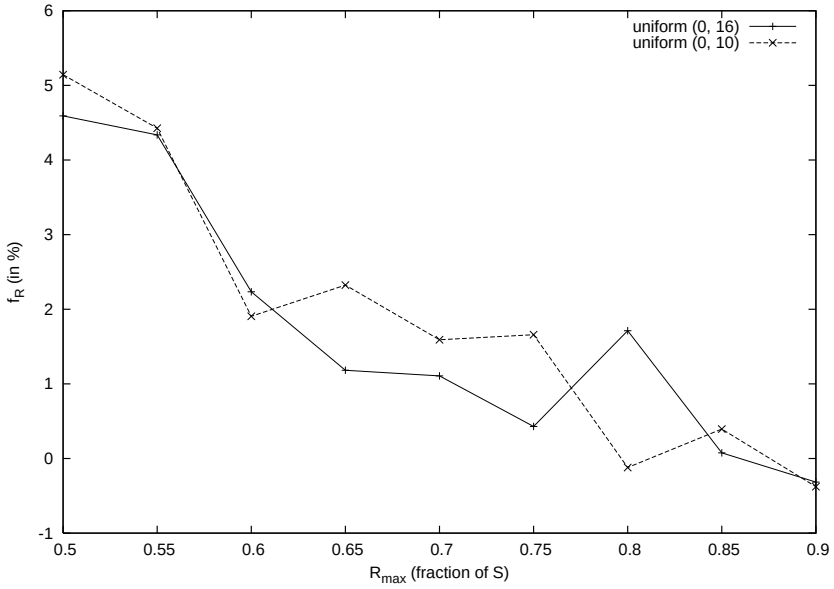
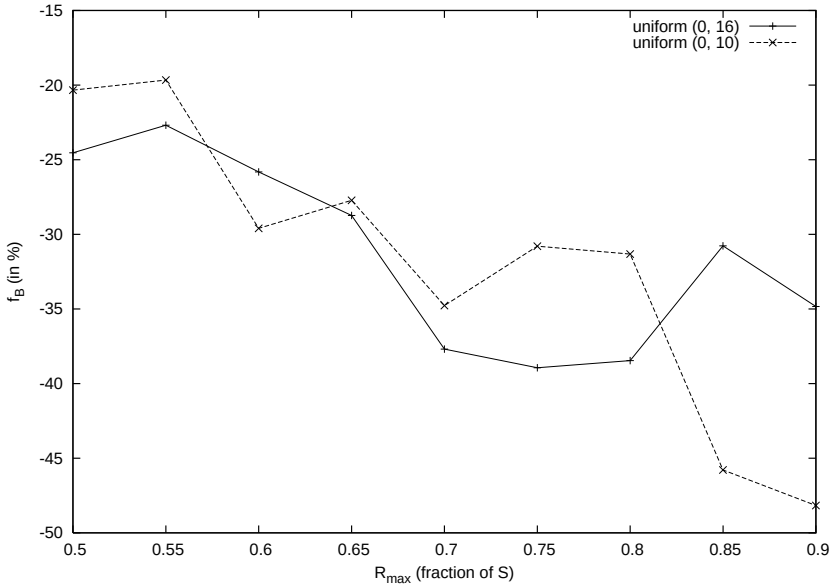


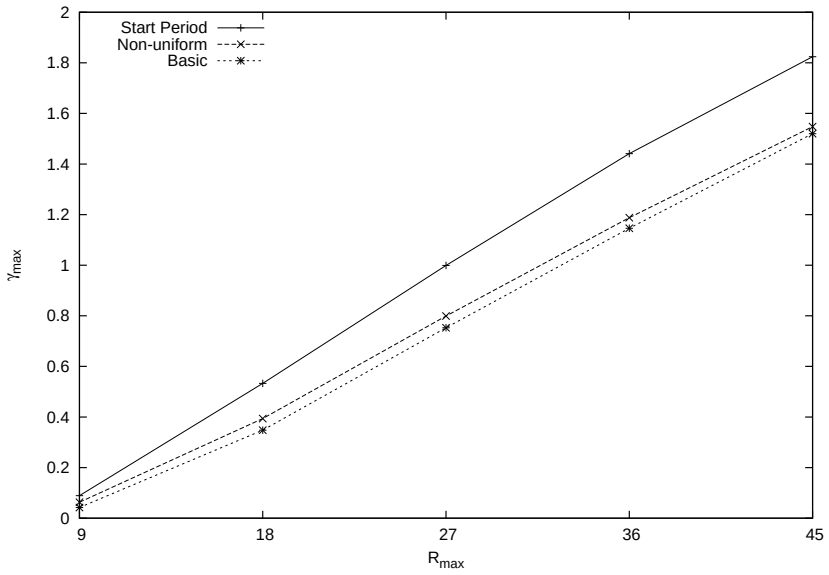
Figure 5.19: f_B plotted against R_{max} for varying x



B_{max}. B_{max} can be viewed as a parameter related to the quality of service provided to the reservation class. In our first set of experiments, two values of B_{max} are used: 1% and 0.1%. We consider these values to be sufficiently small to reflect good quality of service.

For a given B_{max}, we are interested in the highest request arrival rate that the system can support such that $B \leq B_{max}$. Let this highest rate be γ_{max}. We consider the scenarios in Chapter 4 where $S = 10, 20, ..., 50$ and $R_{max} = 0.9S$. The corresponding values of R_{max} are 9, 18, 27, 36, and 45. As to the other parameters, the default values shown in Table 4.1 are used. The results for γ_{max} as a function of R_{max} for $B_{max} = 1.0\%$ are shown in Figure 5.20. The corresponding results for $B_{max} = 0.1\%$ are shown in Figure 5.21. We observe that as R_{max} is increased, γ_{max} increases proportionally for all three strategies. The start period strategy yields the largest γ_{max} for both values of B_{max}, followed by non-uniform resource allocation.

Figure 5.20: γ_{max} plotted against R_{max} for $B_{max} = 1.0\%$



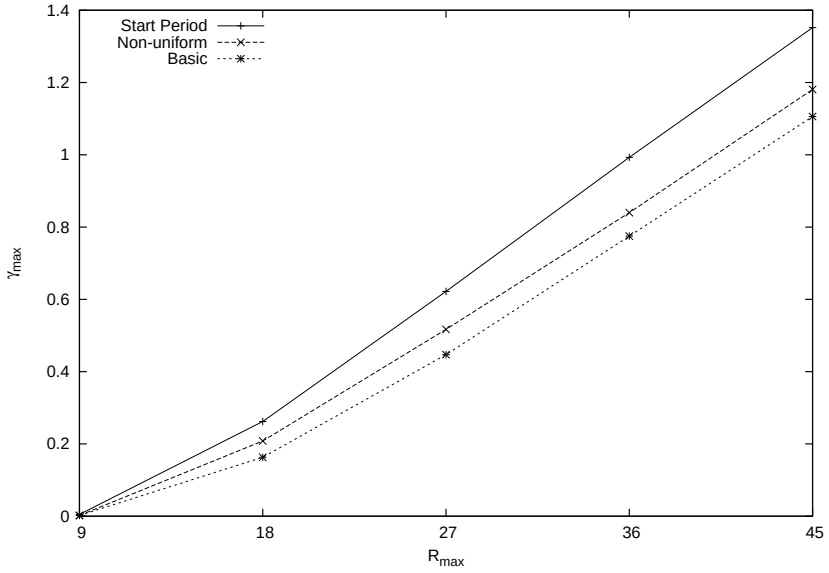
However, the improvement of the non-uniform allocation strategy over the basic strategy is much less significant when compared to the start period strategy. This is consistent with the results in Sections 5.1 and 5.2.

We observe that reducing B_{max} from 1.0% and 0.1% leads to a significant reduction in γ_{max}. For example, at $R_{max} = 27$, the amount of reduction for the start period strategy is close to 40% (from 1.0 to 0.62). We also observe that the improvement (in γ_{max}) of the non-uniform strategy over the basic strategy is more significant for the case of $B_{max} = 0.1\%$. This shows that non-uniform resource allocation is more effective when small values of B_{max} are required.

We are also interested in the utilization of resources by the reservation class. This performance metric, denoted by U, is defined as follows:

$$U = \frac{\gamma * E[d] * E[q] * (1 - B)}{R_{max}}$$

Figure 5.21: γ_{max} plotted against R_{max} for $B_{max} = 0.1\%$



where E[d] and E[q] are the mean duration and mean number of server nodes of a reservation request, respectively. The results for U for $B_{max} = 1.0\%$ and 0.1% are shown in Figures 5.22 and 5.23. In both figures, the start period strategy yields the largest value for U among the three strategies considered. However, when $B_{max} = 1.0\%$, the highest utilization reached using the start period strategy was about 32% (at $R_{max} = 45$) and about 24% when B_{max} is reduced to 0.1%. Consistent with the observations regarding γ_{max}, the results for U show that reducing B_{max} from 1.0% and 0.1% comes with a significant penalty in resource utilization.

An alternative approach to compare the three strategies is to determine R^*_{max}, the smallest value of R_{max} such that $B \leq B_{max}$ for a given value of γ. In Figures 5.24 and 5.25, we plot R^*_{max} as a function of γ for $B_{max} = 1.0\%$ and 0.1%. As to the other parameters, the default values shown in Table 4.1 are used. We observe that as γ is increased, the system requires a higher R^*_{max}. This is not surprising

Figure 5.22: U plotted against R_{max} for γ_{max} supported for $B_{max} = 1.0\%$

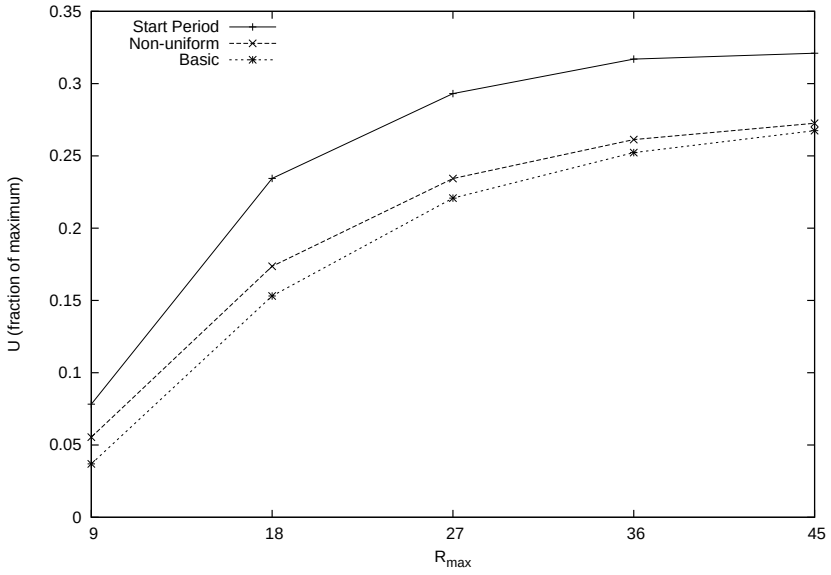


Figure 5.23: U plotted against R_{max} for γ_{max} supported for $B_{max} = 0.1\%$

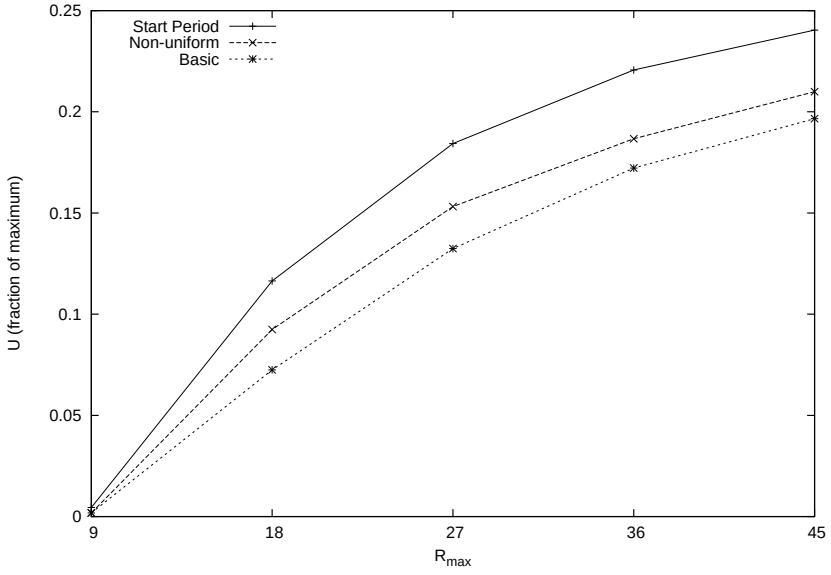
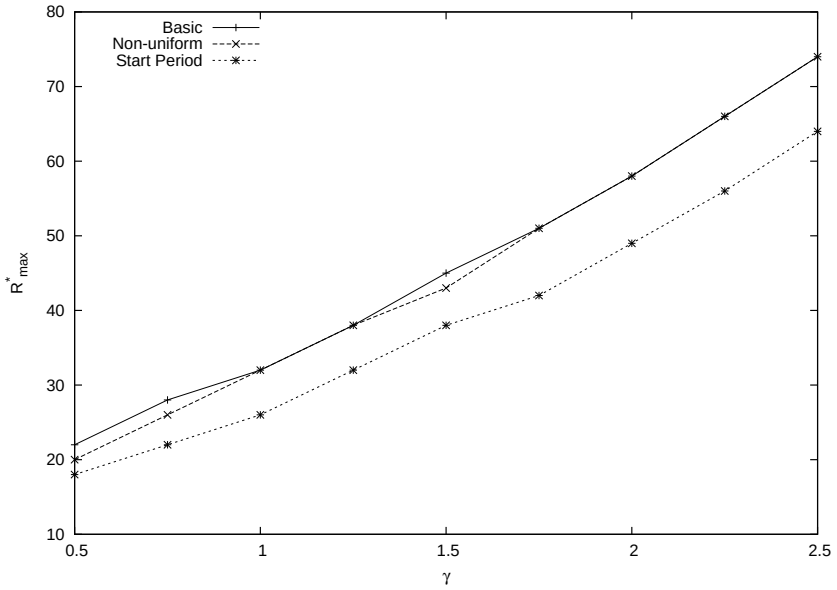


Figure 5.24: R^*_{max} plotted against γ for $B_{max} = 1.0\%$



because more resources are required to maintain the same level of performance. We again observe that the start period strategy has the best performance in terms of having the smallest R^*_{max}. The improvement over the basic strategy is significant. On the other hand, the non-uniform allocation strategy is only slightly better than the basic strategy. We also observe that when B_{max} is reduced from 1.0% to 0.1%, a larger R^*_{max} is required; this observation are consistent with those for γ_{max} and U.

We mentioned earlier that the highest observed utilization is about 32% for $B_{max} = 1.0\%$ and about 24% when $B_{max} = 0.1\%$. The low utilization is due to the requirement of a small blocking probability. If a higher B_{max} can be tolerated, then improvements in both γ_{max} and U can be realized. In our experiments, we evaluate the impact of B_{max} on γ_{max} and U. For these experiments, the default values of the input parameters shown in Table 4.1 are used.

Figure 5.25: R^*_{max} plotted against γ for $B_{max} = 0.1\%$

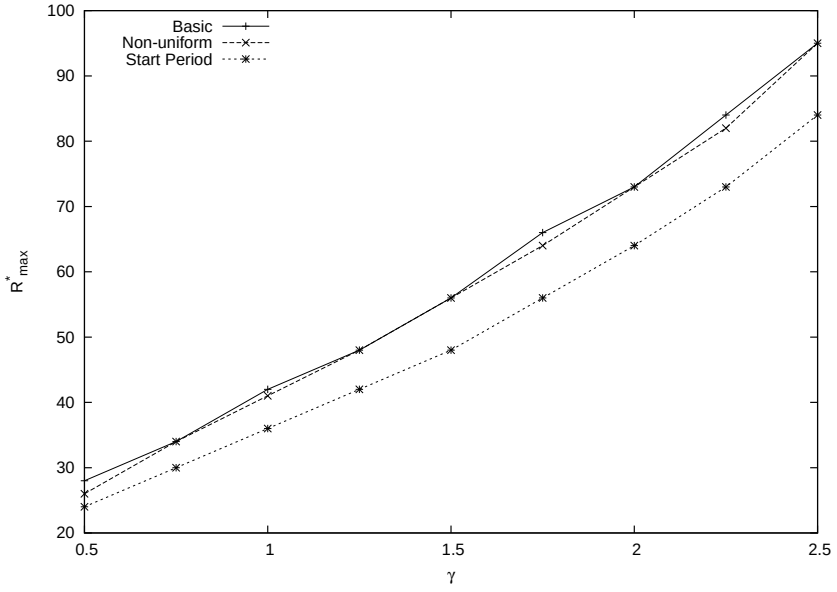


Figure 5.26: U plotted against γ for $B_{max} = 1.0\%$

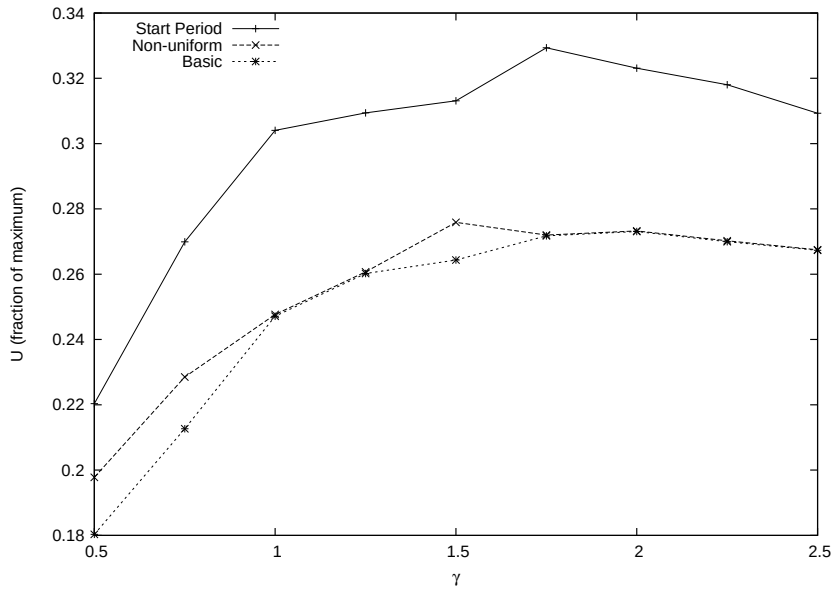
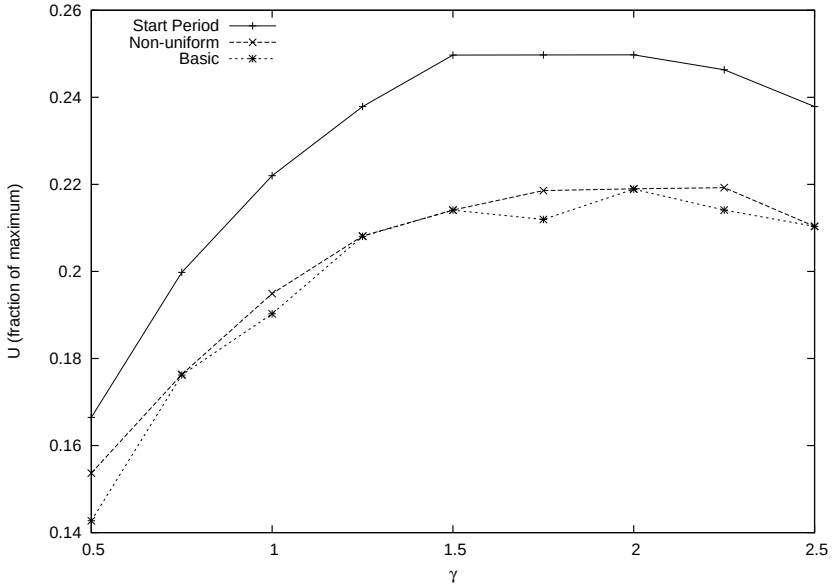


Figure 5.27: U plotted against γ for $B_{max} = 0.1\%$



The results for γ_{max} and U for different values of B_{max} are shown in Figures 5.28 and 5.29, respectively. Again, the start period strategy has the best performance. We also observe that as B_{max} increases, the system is able to handle a higher γ_{max}. For example, when B_{max} is changed from 1% to 10%, γ_{max} for the basic strategy is increased from 1.53 to 2.93, an increase of 92%. Similar increases are observed for the other two strategies. As to the utilization U, the corresponding increase is from 27% to 41%. U is above 50% for three strategies when B_{max} is 25%. These results show that we can get a higher U by tolerating higher B_{max}. However, the maximum value of U observed is still significant less than 100%. We believe this is a consequence of the characteristics of reservation system where a high request arrival rate will improve the chance of accommodating more requests and thereby improving the utilization U. Such high arrival rates will also result in a high blocking probability.

Figure 5.28: γ_{max} plotted against B_{max}

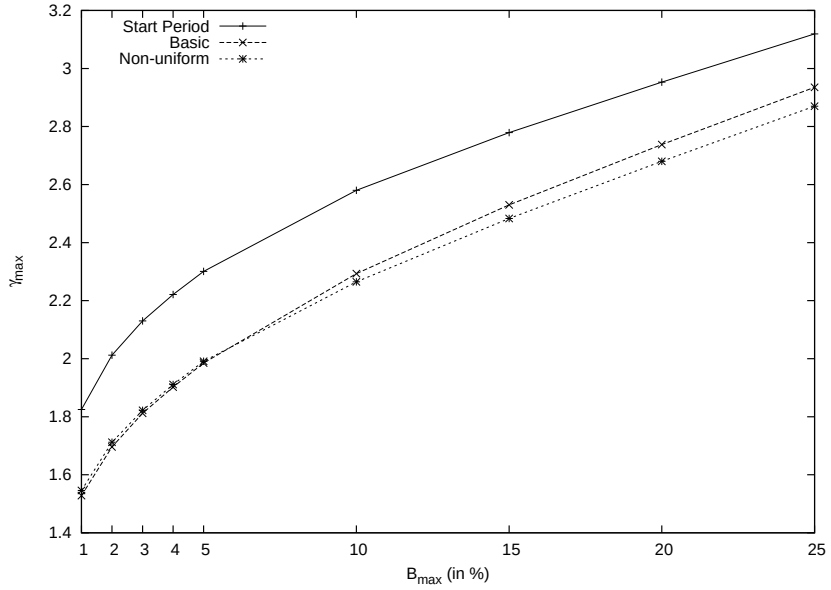
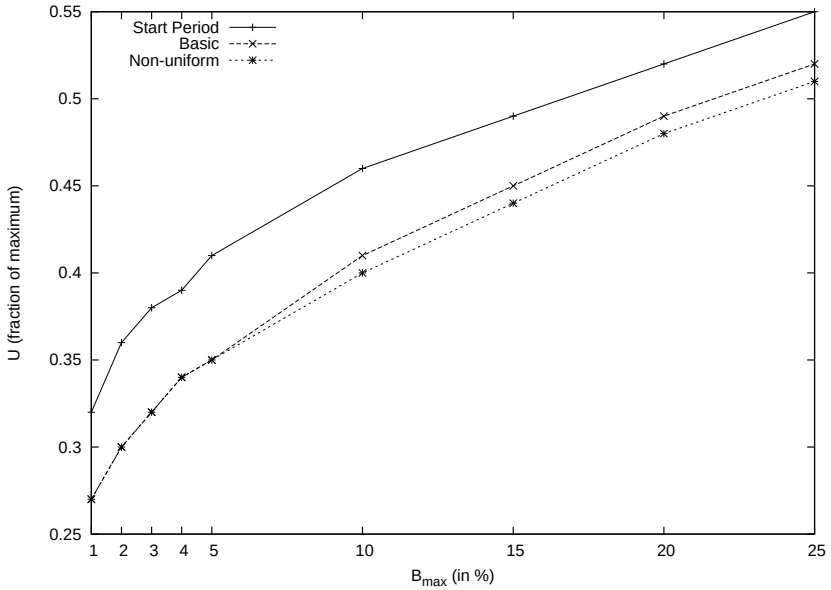


Figure 5.29: U plotted against B_{max} for γ_{max}



In the last set of experiments, we investigate the relative impacts of d and q by varying E[d], E[q] such that the product E[d] * E[q] is fixed. d and q are normal with means E[d] and E[q], respectively, and with a standard deviation of 2. We use simulation to obtain γ_{max} and U for six combinations of E[d] and E[q], namely (E[d], E[q]) = (1, 12), (2, 6), (3, 4), (4, 3), (6, 2), (12, 1) when $B_{max} = 1.0\%$. In the simulation experiments, $S = 50$ and R_{max} is set to 45. As to the other parameters, the default values shown in Table 4.1 are used. The results for γ_{max} and U are shown in Figures 5.30 and 5.31, respectively. We observe that the largest γ_{max} is observed when E[d] = 4 and E[q] = 3, and generally, γ_{max} is highest when E[d] and E[q] are about the same. This observation is true for all three strategies.

We also observe that γ_{max} and U are both lowest when E[d] = 1 and E[q] = 12. Furthermore, a larger E[d] compared to E[q] is more favourable in terms of higher values for γ_{max} and U. Finally, the results in Figures 5.30 and 5.31 again confirm that start period strategy has the best performance, followed by the non-uniform allocation strategy. However, for the case of small resource requirements, i.e., E[q] = 1, the non-uniform strategy and the basic strategy have similar performance. This is due to the high probability of encountering one or more time slots where resources are not available at step one of Algorithm 1.

Figure 5.30: γ_{max} plotted against d and q

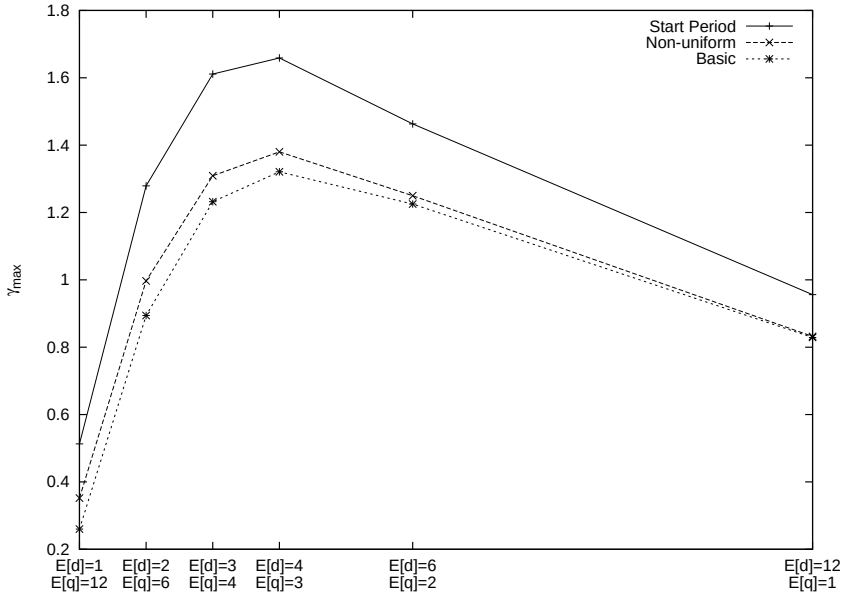
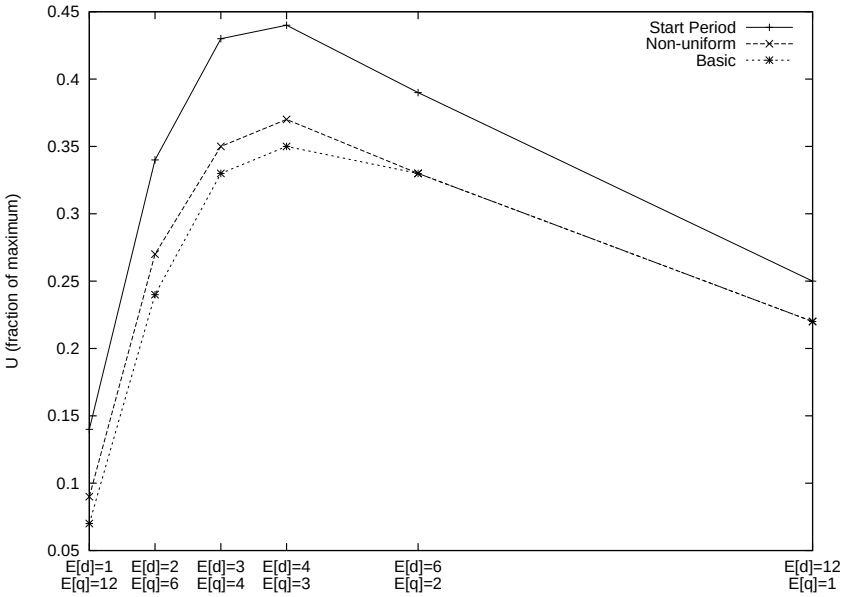


Figure 5.31: U plotted against d and q for γ_{max}



Chapter 6

Results for On-Demand Class

6.1 Initial Observations

Depending on the amount of resources committed to the reservation class, the resources available to the on-demand class at any time instant could be zero or more server nodes. Jobs for the on-demand class are therefore served by multiple servers, but the number of servers may change over time. For such jobs, the performance metric of interest is the mean response time. A suitable model to determine the mean response time is a multi-server model. The traditional model assumes a constant number of servers. Results for a time-varying number of servers are not available. This issue of time-varying capacity has been investigated in the context of a wireless channel in [18]. Our system is different from a wireless channel in the sense that we have multiple server nodes instead of a single channel. We are interested in the impact of resource availability and its variation on response time performance. In our investigation, we use the coefficient of variation of the number of server nodes available (denoted by $C_v[A]$) as our measure of variations in resource availability.

For the on-demand class, job arrivals are modeled by an interarrival time dis-

tribution that is independent of the state of the system. The service time for these jobs is modeled by a probability distribution. The following notation will be used:

α - distribution of interarrival time

β - distribution of service time

Consider first the case where α and β are both exponential and there is a constant number of servers. This is the *M/M/m* model and analytic result for the mean response time (denoted by E[T]) is available [13]. Let λ, μ, and m be the arrival rate, service rate, and number of servers, respectively. E[T] is given by:

$$E[T] = \frac{1}{\mu}(1 + \frac{\varrho}{m(1-\rho)})$$

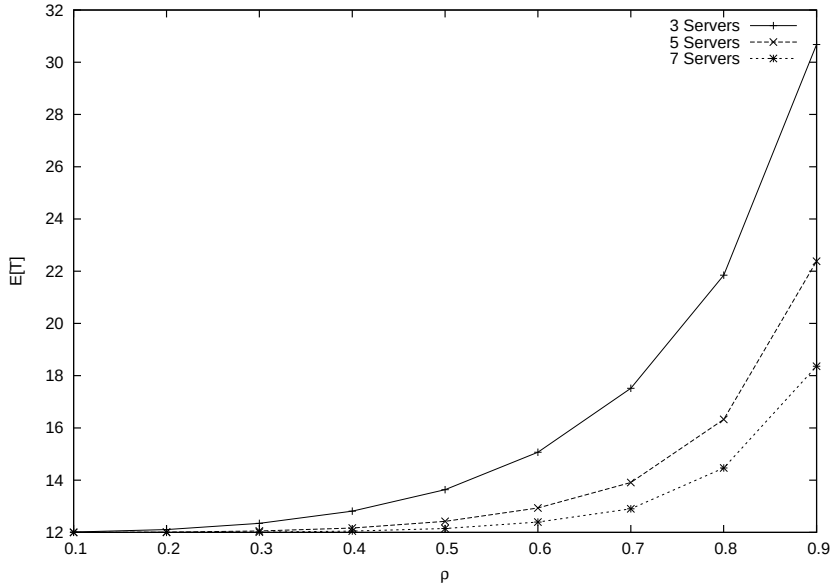
where ϱ = probability of queueing = $P(\geq$ m jobs$) = \frac{(m\rho)^m}{m!(1-p)}p_0$, p_0 = probability of zero jobs in the system = $[1 + \frac{(m\rho)^m}{m!(1-\rho)} + \Sigma_{n=1}^{m-1}\frac{(m\rho)^n}{n!}]^{-1}$, and ρ is the traffic intensity given by $\rho = \frac{\lambda}{m\mu}$.

As a numerical example, we set $\mu = 1/12$, and for a given m, select values of λ such that ρ = 0.1, 0.2, ..., 0.9. In Figure 6.1, we plot E[T] against ρ for m = 3, 5, and 7. We observe that the mean response time increases quickly when $\rho \geq 0.7$. We also note that for the system to be stable, ρ must be less than one and E[T] may become unacceptable when $\rho > 0.9$. We feel that results for the region of $0.7 \leq \rho \leq 0.9$ would provide valuable insights into the impact of input parameters on performance. Therefore, in our evaluation, emphasis is placed on scenarios where $0.7 \leq \rho \leq 0.9$.

6.2 Simulation Results for Mean Response Time

Consider again the base model investigated in Chapter 4. Simulation results for

Figure 6.1: E[T] plotted against ρ



E[R] (the mean number of server nodes allocated to the reservation class) for a range of values of the request arrival rate, and for a range of distributions of duration, resource requirement and start time were presented in Figures 4.1 to 4.14. From these results, one can determine E[A], the mean number of server nodes available to the on-demand class. E[A] is simply given by:

$$E[A] = S - E[R]$$

During the simulation, one can also collect data for $C_v[A]$, the coefficient of variation of the number of server nodes available and E[T], the mean response time of the on-demand class. $C_v[A]$ is given by:

$$C_v[A] = \frac{E[A]}{\sigma[A]}$$

We now present results that show the impact of E[A] and $C_v[A]$ on E[T]. In our simulation experiments, the total number of server nodes S considered are 10, 30, and 50. For the reservation class, the input parameters are selected as follows. γ, q and x are given by the default values in Table 4.1. R_{max} is varied from $0.5S$ to $0.9S$ and five distributions (normal (8, 3), normal (4, 7), normal (6, 3), normal (4, 5) and normal (4, 3) are used for d to achieve different values of E[A] and $C_v[A]$.

For the on-demand class, we assume again that α is exponential with rate λ and β is exponential with rate μ. For the case of time varying resource availability, the traffic intensity is given by

$$\rho = \frac{\lambda}{E[A] * \mu}$$

In the simulation experiments, μ is set to 1/12. For each value of S, λ is selected such that ρ is in the range of 0.7 to 0.9 when R_{max} and d are varied. The values of λ selected for S = 10, 30, and 50 are 0.42, 1.41, and 2.56, respectively. For each of the cases simulated, data for E[A], $C_v[A]$, and E[T] are collected.

In Figures 6.2, 6.3, and 6.4, we plot the results for E[T] against E[A] and $C_v[A]$ for S = 10, 30, and 50, respectively. As expected, E[T] increases as E[A] decreases because a smaller number of server nodes should lead to a longer response time. We also observe that when E[A] is large (or ρ is close to 0.7), the impact $C_v[A]$ on E[T] is not significant. However, when E[A] becomes small (or ρ is increased to 0.9), E[T] tends to increase with $C_v[A]$. It is not clear whether such increase is caused by the decrease in $C_v[A]$, or the increase in $C_v[A]$, or both, because $C_v[A]$ appears to be affected by E[A] also. Therefore, it is not easy to determine the relationship between $C_v[A]$ and E[T] from the results in Figures 6.2, 6.3, and 6.4. A further investigation of the impact of $C_v[A]$ on E[T] is required. The results of such an investigation are presented in the next section.

Figure 6.2: E[T] plotted against E[A] and $C_v[A]$ when $S = 10$

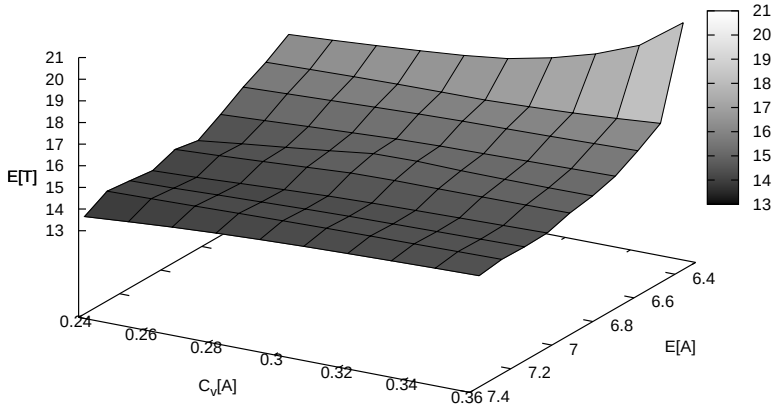


Figure 6.3: E[T] plotted against E[A] and $C_v[A]$ when $S = 30$

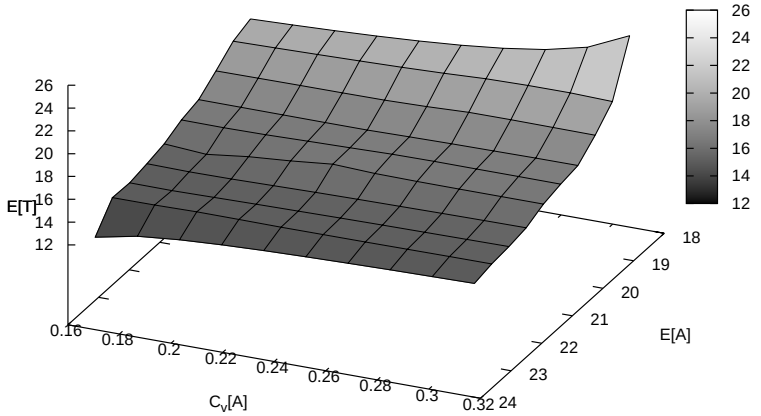
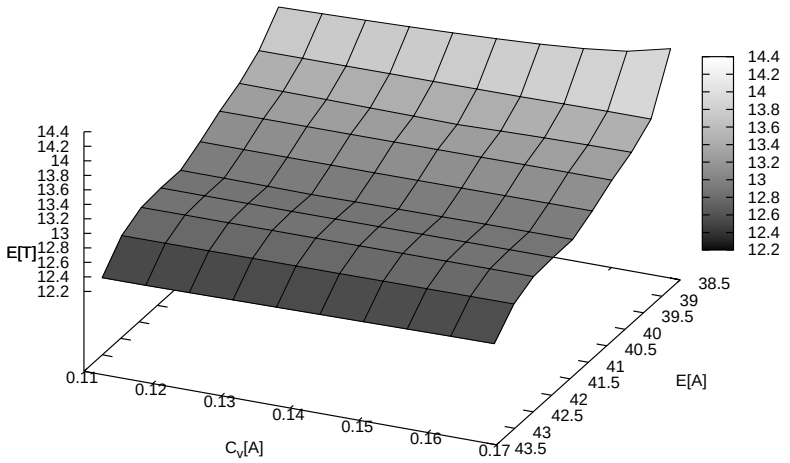


Figure 6.4: E[T] plotted against E[A] and $C_v[A]$ when $S = 50$

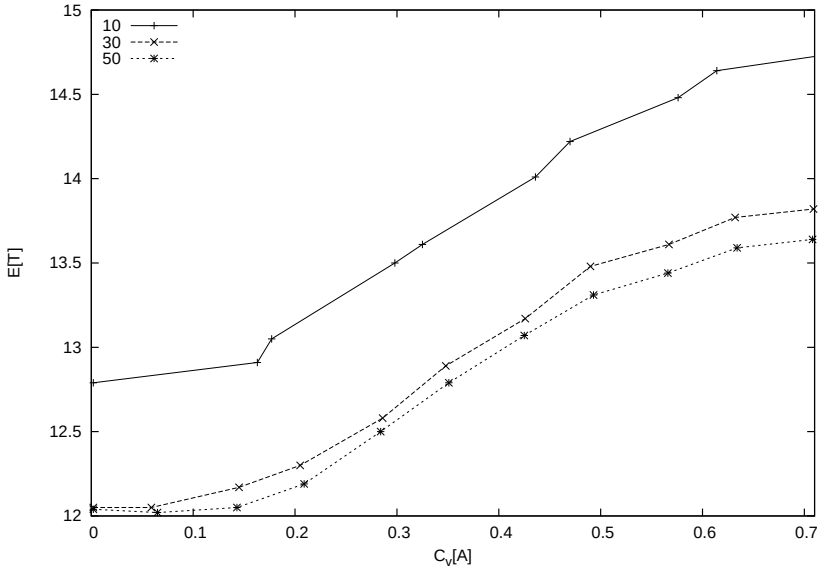


6.3 Impact of $C_v[A]$ on E[T]

Our approach is to create scenarios where E[A] is the same, but the $C_v[A]$'s are different and use them to evaluate the impact of $C_v[A]$ on E[T]. In general, it is not practical to create such scenarios by experimenting with different values of R_{max}, γ, d, and q. Instead, we use simulation to set up time-varying resource availability such that E[A] is the same, but the $C_v[A]$s are different. This is an efficient method to come up with given values of E[A] and $C_v[A]$. We feel that it will not affect our conclusion in terms of the impact of $C_v[A]$ on E[T].

In our simulation experiments, three different values of S, namely $S = 10$, 30, and 50, are considered. The corresponding values of E[A] are 5, 15, and 25 (i.e., E[A] = $0.5S$). Variations in resource availability were generated such that $C_v[A]$ has values ranging from 0 (no variation) to 0.7. We first consider the case where

Figure 6.5: E[T] plotted against $C_v[A]$ when α and β are exponential



α and β are both exponential. The service rate μ is again set to 1/12. For each of the three values of S, the arrival rate λ is selected such that $\rho = 0.7$. Based on this criterion, the values of λ are 0.29, 0.86, and 1.44, when $S = 10$, 30, and 50, respectively. The results for E[T] as a function of $C_v[A]$ are shown in Figure 6.5. We observe that as $C_v[A]$ increases, E[T] remains practically unchanged up to a point and then begins to increase. The point at which E[T] begins to increase is $C_v[A] = 0.1$ for $S = 30$ and 50 (or E[A] = 15 and 25) or $C_v[A] = 0.2$ for $S = 10$ (or E[A] = 5).

Consider next the case of non-exponential distributions for α and β. It was mentioned in [6, 12, 14] that both interarrival time and service time tend to have long tail distributions. We therefore consider the case where α and β are both

Pareto which has probability density function:

$$f(x) = \frac{kx_m^k}{x^{k+1}}$$

where x_m and k are the parameters. We denote such a distribution by Pareto (x_m, k). Let E[P] and $C_v[P]$ be the mean and coefficient of variation of the Pareto distribution. For $k > 2$, we have [20]:

$$E[P] = \frac{kx_m}{k-1}$$

$$C_v[P] = \frac{1}{\sqrt{k(k-2)}}$$

Two Pareto distributions for α and β are used in our simulation experiments. For the first distribution, $k = 2.2$ and the corresponding $C_v[P] = 1.5$. β is assumed to be Pareto (6.547, 2.2). For $S = 10$, 30, and 50, α is Pareto (0.376, 2.2), Pareto (0.627, 2.2), and Pareto (1.882, 2.2), respectively. With these values for α, the traffic intensity $\rho = 0.7$. The results for E[T] as a function of $C_v[A]$ are shown in Figure 6.6. We again observe that as $C_v[A]$ increases, E[T] remains practically unchanged up to a point and then begins to increase. This point is $C_v[A] = 0.1$ for E[A] = 15 and 25 or $C_v[A] = 0.2$ for E[A] = 5. The same observation is made for the second Pareto distribution where $k = 2.12$ ($C_v[P] = 2$). The results are shown in Figure 6.7. These results are for β equal to Pareto (6.349, 2.12). To obtain a traffic intensity of 0.7, α is Pareto (0.365, 2.12), Pareto (0.608, 2.12), and Pareto (1.825, 2.12), for $S = 10$, 30, and 50, respectively.

We conclude from the results in Figures 6.5 to 6.7 show that when $C_v[A]$ is above a particular value, there will be noticeable increase in E[T]. This value is affected by the value of E[A], namely, 0.1 when E[A] = 15 and 25 and 0.2 when E[A] = 5, but is not sensitive to the distributions used for α and β.

Figure 6.6: E[T] plotted against $C_v[A]$ when α and β are Pareto with $C_v[P] = 1.5$

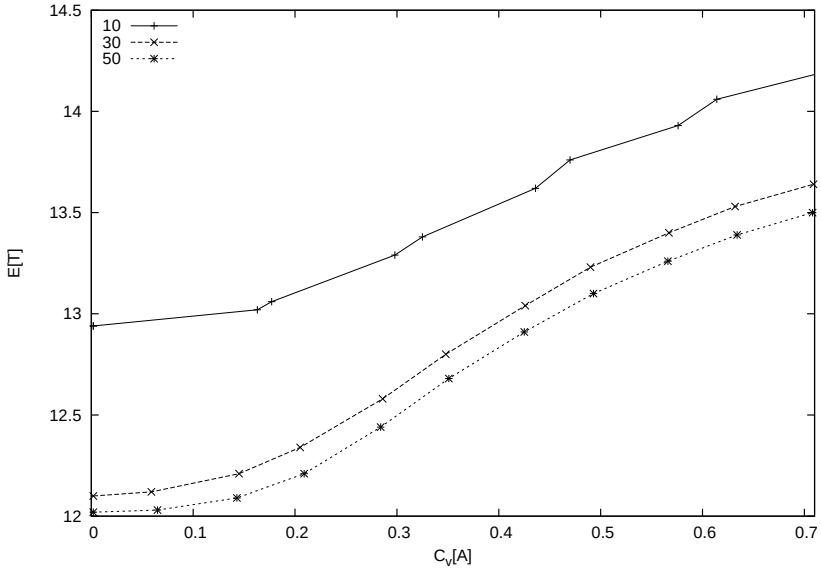
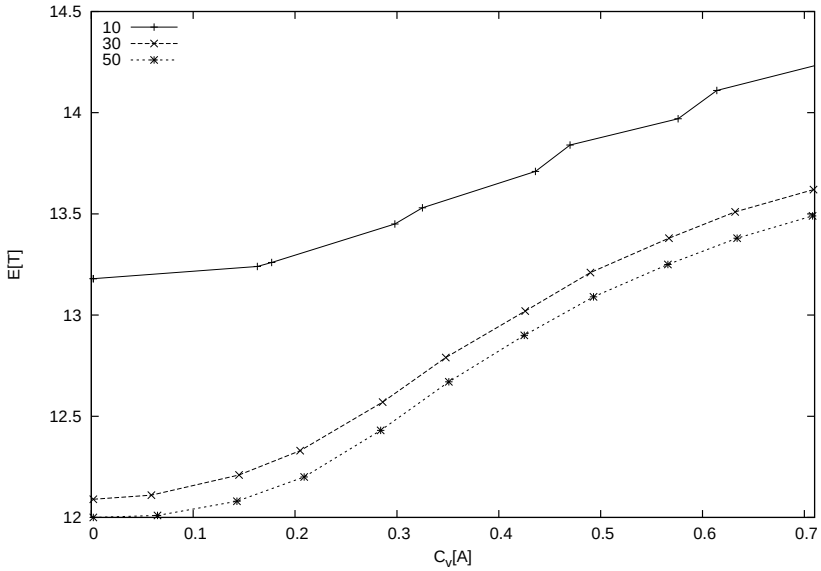


Figure 6.7: E[T] plotted against $C_v[A]$ when α and β are Pareto with $C_v[P] = 2$



6.4 Additional Remark

The results in Figures 6.5 to 6.7 provide useful guidelines for non-uniform resource allocation for the reservation class. Consider, for example, a reservation request with resource requirement of 5 server nodes. In Algorithm 1, the number of server nodes allocated varies over time, but its mean (denoted by E[N]) is the same as the resource requirement of the reservation request. For a given allocation, we can also compute $C_v[N]$, the coefficient of variation of the number of nodes allocated. Suppose the reserved server nodes are used to support interactive users and mean response time is the performance metric of interest. Our results in Figures 6.5 to 6.7 show that if $C_v[N] \leq 0.2$, the mean response time is not affected by the variation in resource availability. Non-uniform allocation is therefore an effective strategy in terms of meeting the response time performance.

In the case where $C_v[N] > 0.2$, the mean response time will be larger than that for the case of no variation. The increase in mean response time can be viewed as not allocating enough resources. A possible strategy is to allocate more server nodes on average. As an example, when α and β are both exponential, one can interpret the results in Figure 6.5 as E[T] = 12.8 when E[N] = 5 and there is no variation in resource allocation. One can also interpret that E[T] is noticeable larger than 12.8 when $C_v[N] > 0.2$. When this happens, a larger value of E[N] can be used to realize a mean response time of 12.8.

Consider the case of $C_v[N] = 0.3$, the results in Figure 6.5 show that E[T] is 13.5. Using simulation, we found that if E[N] is set to 6 instead of 5, the mean response time becomes 12.8, which is the same as the case of E[N] = 5 with no variation. This represents a 20% increase in resource allocation on average. As another example, when $C_v[N] = 0.44$, E[T] becomes 14 for E[N] = 5. Our simulation results show that to get $E[T] \leq 12.8$, we would require E[N] = 7, which means a 40% increase

in resource allocation. It seems that the additional resources required to alleviate the increase in mean response time due to a large value of $C_v[N]$ may be too costly. Nevertheless, if $C_v[N] \leq 0.2$, no additional resources are required and non-uniform resource allocation is a very effective strategy.

Chapter 7

Conclusions and Future Research

7.1 Conclusions

In this thesis, we used simulation to investigate the impact of variations in workload of reservation requests and on-demand access on performance. Performance metrics considered included resource utilization and blocking probability of reservation requests, and mean response times of on-demand access.

For reservation requests, the basic strategy is to impose a limit on the number of server nodes that could be reserved at any time instant. Our results indicated that resource usage was low compared to the reservation limit if the blocking probability were to be kept at an acceptable level. Two other strategies to improve system performance for reservation requests were evaluated. The first strategy, referred to as the start period strategy, allowed the start time of reservations to be delayed up to some maximum value. The second strategy allowed resources to be allocated in a non-uniform manner as long as the mean number of server nodes allocated is the same as the amount of resources required. Simulation results show that both strategies lead to increase in resource usage and decrease in blocking probability when compared to the basic strategy. Between the two, the start period strat-

egy has significantly better performance, but the requesting user may be required to accept an alternative start time for the reservation. Despite the performance improvement, the maximum resource utilization observed is about 32% when the blocking probability is kept below 1.0%.

We also investigated scenarios where higher blocking probabilities can be tolerated. For such scenarios, we observed increase in resource utilization for a given arrival rate, or increase in request arrival rates that can be supported. The highest resource utilization observed was about 55% when the blocking probability can be as high as 25%. Our conclusion was that a high request arrival rate will improve the chance of accommodating more requests and thereby improving the utilization, but will also result in a high blocking probability. Even with a blocking probability of 25% which is considered to be unacceptable, the resource utilization was below 60%.

With resources committed to reservations, the number of server nodes available for on-demand access may change over time. Initial results showed that the mean response time of on-demand access was increased when the mean resource availability was reduced, but the impact of variations in resource availability was not clearly shown. Additional results showed that the mean response time remained practically unchanged up to a certain value of the coefficient of variation of the number of server nodes, and then begins to increase. This value was 0.1 or 0.2 for the cases considered. These results indicated that the non-uniform resource allocation strategy for reservation requests may lead to degraded performance (in case the reserved resources are used for interactive access) if the variations in resource allocation over the duration of the reservation is too large. These results can be used to develop guidelines for the non-uniform resource allocation strategy.

7.2 Future Research

In our investigation, a reservation request is accepted if the requested resources are available, and rejected otherwise. In some cases, it may be advantageous to reject a request even though there are sufficient resources because such an action may lead to better resource usage in the future. Of interest is an investigation of the conditions under which such advantages can be realized.

In our investigation, no restriction was placed on the start time of a reservation request. A possible extension is to impose a minimum start time for all requests, known as the notice period [3]. This would allow the system to batch reservation requests and make better use of resources based on the requirements of these requests. The drawback is that the requesting user may not be informed immediately about whether the request was accepted or not. Extension of our work to include this feature is a topic worthy of investigation.

In some applications, the reserved resource may not be required towards the end of the duration of the reservation request. Normally, a request is not accepted if sufficient resources are not available. The possibility of an earlier end time allow for accepting more requests than normal with the hope that resources are available at the required times. Our model can be extended to estimate that probability of meeting a requests resource requirement for different strategies for accepting requests.

For on-demand access, we have used the mean response time as the performance metric of interest. In the financial world, a better measure of risk is VAR, or *value at risk*. Other metrics may include Prob[response time $\leq$ x], which is the probability that the response time is less than or equal to a given value. An example of a conditional VAR measure is "in the worse 1% of the cases, the average response time is Z", which is measured with this metric as Prob[response time $\leq$ Z] = 99%.

An investigation of this metric would provide additional insight into the system performance for on-demand access.

We have evaluated the performances of the reservation class and the effects of the reservation class on the performance of on-demand access. A related work is to evaluate the tradeoffs in performance between these two classes. This may include finding parameters that allow for optimal performance to both of these classes at the same time.

References

[1] Suman Basuroy, Dung Nguyen, and Richard E. Chatwin. Multiperiod airline overbooking with a single fare class. *Oper. Res.*, 46(6):805–819, 1998. 5

[2] Dimitris Bertsimas and Sanne de Boer. Simulation-based booking limits for airline revenue management. *Oper. Res.*, 53(1):90–106, 2005. 5

[3] E. G. Coffman, Jr. and Predrag Jelenković. Threshold policies for single-resource reservation systems. *SIGMETRICS Perform. Eval. Rev.*, 28(4):9–10, 2001. 61

[4] Domenico Ferrari, Amit Gupta, and Giorgio Ventre. Distributed advance reservation of real-time connections. *Multimedia Syst.*, 5(3):187–198, 1997. 4

[5] The Apache Software Foundation. Apache vcl. http://cwiki.apache.org/VCL/, April 2009. 1

[6] Mark W. Garrett and Walter Willinger. Analysis, modeling and generation of self-similar vbr video traffic. *SIGCOMM Comput. Commun. Rev.*, 24(4):269–280, 1994. 54

[7] Matthias Grossglauser and David N. C. Tse. A time-scale decomposition approach to measurement-based admission control. *IEEE/ACM Trans. Netw.*, 11(4):550–563, 2003. 6

[8] J. J. Harms and J. W. Wong. Performance modeling of a channel reservation service. *Computer Networks and ISDN Systems*, 27(11):1487 – 1497, 1995. 4

[9] Janelle J. Harms. *Performance evaluation of scheduling algorithms for reservation systems with applications to high-speed networks*. PhD thesis, University of Waterloo, Waterloo, Ont., Canada, Canada, 1991. 2, 4, 5, 20

[10] Janelle J. Harms. Video distribution to the home using advance reservation. In *LCN '97: Proceedings of the 22nd Annual IEEE Conference on Local Computer Networks*, pages 170–178, Washington, DC, USA, 1997. IEEE Computer Society. 2, 5

[11] J.J. Harms and J.W. Wong. Performance of scheduling algorithms for channel reservation. *Computers and Digital Techniques, IEE Proceedings* -, 141(6):341–346, Nov 1994. 4, 5

[12] Carl M. Harris, Percy H. Brill, and Martin J. Fischer. Internet-type queues with power-tailed interarrival times and computational methods for their analysis. *INFORMS J. on Computing*, 12(4):261–271, 2000. 54

[13] Raj Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons, 1991. 49

[14] Vern Paxson. Empirically derived analytic models of wide-area tcp connections. *IEEE/ACM Trans. Netw.*, 2(4):316–336, 1994. 54

[15] Wilko Reinhardt. Advance reservation of network resources for multimedia applications. In *IWACA '94: Proceedings of the Second International Workshop on Multimedia*, pages 23–33, London, UK, 1994. Springer-Verlag. 4

[16] Gurmeet Singh, Carl Kesselman, and Ewa Deelman. Adaptive pricing for resource reservations in shared environments. In *GRID '07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing*, pages 74–80, Washington, DC, USA, 2007. IEEE Computer Society. 5

[17] J. Siwko and I. Rubin. Connection admission control for capacity-varying networks with stochastic capacity change times. *IEEE/ACM Trans. Netw.*, 9(3):351–360, 2001. 5

[18] Hongxia Sun and Carey Williamson. On effective capacity in time-varying wireless networks. *Simulation Series*, 38(3):111–120, 2006. 2, 5, 48

[19] Hongxia Sun, Qian Wu, and C. Williamson. Impact of stochastic traffic characteristics on effective capacity in cdma networks. *Local Computer Networks, Annual IEEE Conference on*, 0:793–800, 2006. 5

[20] Wikipedia. Pareto distribution — Wikipedia, the free encyclopedia, 2009. [Online; accessed 23-June-2009]. 55

[21] Lars C. Wolf, Luca Delgrossi, Ralf Steinmetz, Sybille Schaller, and Hartmut Wittig. Issues of reserving resources in advance. In *NOSSDAV '95: Proceedings of the 5th International Workshop on Network and Operating System Support for Digital Audio and Video*, pages 28–38, London, UK, 1995. Springer-Verlag. 4

[22] Charlie Xu. Advance resource reservation in networks. Master's thesis, University of Waterloo, 1998. 2, 4, 5