SCHEDULING ON E-COMMERCE SYSTEMS

By

Yinjing Wang

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements
for the degree of

**Master of Science
in Information and Systems Science**

Department of Systems and Computer Engineering

Carleton University
Ottawa, Canada, K1S 5B6
December 2003

© Copyright 2003, Yinjing Wang

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services

Acquisitions et
services bibliographiques

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-93992-8

Our file Notre référence

ISBN: 0-612-93992-8

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this dissertation.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de ce manuscrit.

While these forms may be included in the document page count, their removal does not represent any loss of content from the dissertation.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Performance in e-commerce is usually tracked by traditional performance metrics such as response time and throughput. However, one of the most important performance metrics for an e-commerce Web site is profit and revenue. Thus, resource management policies for e-commerce systems should improve the business-oriented performance metrics, such as profit and revenue, while maintaining the more system-oriented performance metrics such as response time at acceptable levels.

Four scheduling policies for e-commerce systems are described and their performances are investigated in this thesis through a performance prototype. These policies use one or more of a number of customer and system characteristics for determining the priority of a customer. These characteristics include, the class of a customer, the current state of a customer, the total time a customer has spent in the system and the amount of money the customer has in her/his shopping cart. A workload generation tool that includes a request manager was developed in order to evaluate the performances of these four business-oriented dynamic policies. The results of our experiments show that at a high workload, the dynamic priority policies proposed in this thesis can increase business-oriented metrics, such as revenue per second, by as much as 50% over a conventional First Come First Served policy. E-commerce Web sites making use of these policies will be able to improve revenue earned at high workloads, while maintaining the same server capacity as observed on the conventional systems.

Acknowledgements

I would like to gratefully acknowledge the enthusiastic supervision of Professor Shikharesh Majumdar, who has been a constant source of support and inspiration during the research of this thesis. I would also like to give my thanks to all my colleagues in the RADS Lab for their help.

I would like to thank the faculty members of the RADS lab group for their help and support.

Finally, I am forever indebted to my parents and my husband, Hui, for their understanding, patience and encouragement when it was most required.

TABLE OF CONTENTS

LIST OF TABLES.....	V
LIST OF FIGURES	VI
GLOSSARY OF TERMS AND SYMBOLS	IX
CHAPTER 1 INTRODUCTION	1
1.1 THESIS MOTIVATION AND OBJECTIVE	1
1.2 THESIS CONTRIBUTIONS	2
1.3 THESIS ORGANIZATION.....	4
CHAPTER 2 LITERATURE OVERVIEW.....	6
2.1 EXISTING SCHEDULING POLICIES	6
2.1.1 Arrival Time-Based Scheduling	6
2.1.2 Service Time-Based Scheduling.....	7
2.2 RESOURCE MANAGEMENT POLICIES.....	9
2.2.1 Cluster Reserves.....	9
2.2.2 Priority – Based Scheduling Policy	11
2.2.3 Integrated Resource Management.....	12
2.2.4 Integrated Real-time Resource Scheduler (IRS).....	14
2.3 PROFIT-IMPROVING POLICIES FOR E-COMMERCE SYSTEMS	16
2.3.1 Usage-Based Web Personalization	16
2.3.2 Profit-Oriented Feedback Control System.....	19
2.3.3 Recommender System	21
CHAPTER 3 SCHEDULING POLICIES FOR E-COMMERCE SYSTEMS	24
3.1 CUSTOMER BEHAVIOR MODEL GRAPH OF DIFFERENT CLASSES OF CUSTOMERS	24
3.1.1 Workload Characterization	24
3.1.2 CBMG of Different Classes of Customers	26
3.2 LIMITATIONS OF CURRENT RESOURCE MANAGEMENT POLICIES	28
3.3 POLICY BASED ON CUSTOMER’S CLASS (PONLY).....	29
3.4 POLICY BASED ON CUSTOMER’S SHOPPING STATE	29
3.5 POLICY BASED ON CUSTOMER’S TOTAL TIME SPENT IN SYSTEM	30
3.6 INTEGRATION OF DIFFERENT PRIORITY-BASED POLICIES	31
3.6.1 Class-based and State-based (Sonly)	31
3.6.2 Class-based and Time-based (Tonly).....	32
3.6.3 Class-based, State-based and Time-based (T&S)	32
CHAPTER 4 PERFORMANCE PROTOTYPE.....	35
4.1 THE HTTPERF WORKLOAD GENERATION TOOL	35
4.2 SCALABLE-CLIENT (S-CLIENT) GENERATION TOOL.....	36
4.3 SCALABLE URL REFERENCE GENERATOR (SURGE)	37
4.4 CBMG-BASED PERFORMANCE PROTOTYPE	39

4.4.1 Closed Model vs. Open Model	39
4.4.2 Performance Prototype Description.....	39
4.4.2.1 Request Generating Component	41
4.4.2.2 Request Managing Component.....	43
4.5 APACHE-BASED SERVER.....	44
4.6 WORKLOAD PARAMETERS AND SYSTEM PARAMETERS	47
4.6.1 Workload Parameters.....	47
4.6.2 System Parameters	48
4.7 PERFORMANCE METRICS.....	50
4.8 VALIDATIONS.....	52
4.8.1 Applying Little's Law.....	52
4.8.2 Same Navigation Patterns	54
4.8.3 Same Priority Settings vs. Different Priority Settings	56
4.9 ACCURACY OF MEASUREMENT.....	58
CHAPTER 5 PERFORMANCE OF SCHEDULING POLICIES	60
5.1 EXPERIMENTAL DESIGN.....	60
5.2 EFFECT OF WORKLOAD AND SYSTEM PARAMETERS ON END-TO-END TIME	64
5.2.1 Effect of Arrival Rate.....	64
5.2.2 Effect of Think Time	69
5.2.3 Effect of Angry Time.....	78
5.2.4 Effect of Time Threshold.....	87
5.2.5 Effect of CBMG.....	92
5.3 EFFECT OF WORKLOAD AND SYSTEM PARAMETERS ON ANGRY FACTOR	95
5.3.1 Effect of Arrival Rate.....	95
5.3.2 Effect of Think Time	96
5.3.3 Effect of Angry Time.....	98
5.3.4 Effect of Time Threshold.....	99
5.3.5 Effect of CBMG.....	101
5.4 EFFECT OF WORKLOAD AND SYSTEM PARAMETERS ON REVENUE	102
5.4.1 Effect of Arrival Rate.....	102
5.4.2 Effect of Think Time	104
5.4.3 Effect of Angry Time.....	105
5.4.4 Effect of Time Threshold.....	106
5.4.5 Effect of CBMG.....	108
5.5 DISCUSSION	109
CHAPTER 6 CONCLUSIONS	113
6.1 SUMMARY	113
6.2 CONCLUSIONS	115
6.3 FUTURE WORK	118
REFERENCES	120

List of Tables

Table 4-1: Apache Server Configuration Parameters	45
Table 4-2: Transition Probabilities	48
Table 4-3: Priority Settings for Three Classes	49
Table 4-4: Time Thresholds for Different Classes	50
Table 4-5: Difference Percentages.....	54
Table 5-1: Levels of Factors	60
Table 5-2: Transition Probabilities of CBMG	61
Table 5-3: The Default Parameter Values	62
Table 5-4: Excepted Mean Residence Times (ms) and Average Money a Customer Paid (\$/customer)	63
Table 5-5: Time Threshold Settings for Three Customer Classes (ms).....	87
Table 5-6: Transition Probabilities of Default CBMG	92
Table 5-7: Transition Probabilities of Alternate CBMG	92
Table 6-1: Effect of Proposed Policies Compared to FCFS Policy	115

List of Figures

Figure 2-1: Cluster Reserves.....	10
Figure 2-2: Service Cluster with Integrated Resource Management Policy [37]	13
Figure 2-3: Runtime Environment of a Service Node [37].....	13
Figure 2-4: The Software Architecture of the IRS Policy	15
Figure 2-5: General Architecture for Usage-based Web Personalization [26]	18
Figure 2-6: Architecture of the Profit-Oriented Feedback Control System [8]	19
 Figure 3-1: Workload Characterization Methodology.....	 25
Figure 3-2: CBMG for Three Classes of Customers	27
Figure 3-3: Policy Based on Customer's Class.....	29
Figure 3-4: Policy Based on Customer's Total Time Spent in System	30
Figure 3-5: Policy Based on Class and State of Customers.....	31
Figure 3-6: Policy Based on Class and Time Customer Spends in System.....	33
Figure 3-7: Policy Based on Class, State and Time.....	34
 Figure 4-1: Open Model Performance Prototype.....	 40
Figure 4-2: Effect of Using the Same Navigational Patterns on End-to-End Time....	55
Figure 4-3: Effect of Using the Same Navigational Patterns on the Average Number of States Customers Visited	55
Figure 4-4: Performance Achieved with the Same Priority Setting (90-90-90)	56
Figure 4-5: Performance Achieved with Different Priority Settings	57
 Figure 5-1: Effect of Arrival Rate on End-to-End Time of Gold Customers	 65
Figure 5-2: Effect of Arrival Rate on End-to-End Time of Silver Customers.....	65
Figure 5-3: Effect of Arrival Rate on End-to-End Time of Bronze Customers.....	66
Figure 5-4: Discrimination Factor for Silver Customers	67
Figure 5-5: Discrimination Factor for Bronze Customers	67
Figure 5-6: Maximum Arrival Rates for Scheduling Policies	68
Figure 5-7: Effect of Think Time on End-to-End Time of Gold Customers with Ponly Policy	70
Figure 5-8: Effect of Think Time on End-to-End Time of Gold Customers with Sonly Policy	70
Figure 5-9: Effect of Think Time on End-to-End Time of Gold Customers with Tonly Policy	71
Figure 5-10: Effect of Think Time on End-to-End Time of Gold Customers with T&S Policy	71
Figure 5-11: Effect of Think Time on End-to-End Time of Silver Customers with Ponly Policy.....	72
Figure 5-12: Effect of Think Time on End-to-End Time of Silver Customers with Sonly Policy.....	73
Figure 5-13: Effect of Think Time on End-to-End Time of Silver Customers with Tonly Policy.....	73
Figure 5-14: Effect of Think Time on End-to-End Time of Silver Customers with T&S Policy.....	74

Figure 5-15: Effect of Think Time on End-to-End Time of Bronze Customers with Ponly Policy	74
Figure 5-16: Effect of Think Time on End-to-End Time of Bronze Customers with Sonly Policy	75
Figure 5-17: Effect of Think Time on End-to-End Time of Bronze Customers with Tonly Policy	75
Figure 5-18: Effect of Think Time on End-to-End Time of Bronze Customers with T&S Policy	76
Figure 5-19: Effect of Think Time on End-to-End Time for All Customer Classes at High Arrival Rate	77
Figure 5-20: Effect of Angry Time on End-to-End Time of Gold Customers with Ponly Policy	78
Figure 5-21: Effect of Angry Time on End-to-End Time of Gold Customers with Sonly Policy	79
Figure 5-22: Effect of Angry Time on End-to-End Time of Gold Customers with Tonly Policy	79
Figure 5-23: Effect of Angry Time on End-to-End Time of Gold Customers with T&S Policy	80
Figure 5-24: Effect of Angry Time on End-to-End Time of Silver Customers with Ponly Policy	81
Figure 5-25: Effect of Angry Time on End-to-End Time of Silver Customers with Sonly Policy	82
Figure 5-26: Effect of Angry Time on End-to-End Time of Silver Customers with Tonly Policy	82
Figure 5-27: Effect of Angry Time on End-to-End Time of Silver Customers with T&S Policy	83
Figure 5-28: Effect of Angry Time on End-to-End Time of Bronze Customers with Ponly Policy	83
Figure 5-29: Effect of Angry Time on End-to-End Time of Bronze Customers with Sonly Policy	84
Figure 5-30: Effect of Angry Time on End-to-End Time of Bronze Customers with Tonly Policy	84
Figure 5-31: Effect of Angry Time on End-to-End Time of Bronze Customers with T&S Policy	85
Figure 5-32: Effect of Angry Time on End-to-End Time for All Customer Classes at High Arrival Rate	86
Figure 5-33: Effect of Time Threshold on End-to-End Time of Gold Customers with T&S Policy	88
Figure 5-34: Effect of Time Threshold on End-to-End Time of Silver Customers with T&S Policy	88
Figure 5-35: Effect of Time Threshold on End-to-End Time of Bronze Customers with T&S Policy	89
Figure 5-36: Effect of Time Threshold on End-to-End Time of Gold Customers with Tonly Policy	90
Figure 5-37: Effect of Time Threshold on End-to-End Time of Silver Customers with Tonly Policy	90

Figure 5-38: Effect of Time Threshold on End-to-End Time of Bronze Customers with Tonly Policy.....	91
Figure 5-39: Effect of CBMGs on End-to-End Time for Gold Customers	93
Figure 5-40: Effect of CBMGs on End-to-End Time for Silver Customers.....	94
Figure 5-41: Effect of CBMGs on End-to-End Time for Bronze Customers.....	94
Figure 5-42: Effect of Arrival Rate on the Angry Factor	95
Figure 5-43: Effect of Think Time on the Angry Factor	97
Figure 5-44: Effect of Angry Time on the Angry Factor.....	98
Figure 5-45: Effect of Time Threshold on the Angry Factor with Tonly Policy.....	100
Figure 5-46: Effect of Time Threshold on the Angry Factor with T&S Policy	100
Figure 5-47: Effect of Different CBMGs on the Angry Factor	102
Figure 5-48: Effect of Arrival Rate on Revenue.....	103
Figure 5-49: Effect of Think Time on Revenue.....	105
Figure 5-50: Effect of Angry Time on Revenue per Second	106
Figure 5-51: Effect of Time Threshold on Revenue per Second with T&S Policy..	107
Figure 5-52: Effect of Time Threshold on Revenue per Second with Tonly Policy	108
Figure 5-53: Effect of CBMG on Revenue per Second.....	109

Glossary of Terms and Symbols

AF	Angry Factor
AR	Arrival Rate
AT	Angry Time
CBMG	Customer Behavior Model Graph
CGI	Common Gateway Interface
D	Discrimination Factor
E	Mean End-to-End Time
ET	Experiment Time
FCFS	First-Come First-Served
HTTP	HyperText Transfer Protocol
IRS	Integrated Real-Time Resource Scheduler
LCFS	Last-Come First-Served
P	Priority of a Customer Class
Pi	Probabilities of Transition in CBMG
R	Revenue Earned per Second
RR	Round Robin
RS	Random Scheduling
S	Average Number of States Customers Visited
S-Client	Scalable-Client
SJF	Shortest Job First
SRPT	Shortest Remaining Processing Time
SURGE	Scalable URL Reference Generator
Th	Think Time
URL	Uniform Resource Locator
V	Number of Valid Threads

Chapter 1 Introduction

1.1 Thesis Motivation and Objective

As the number of e-commerce websites continues to increase, the competition between e-commerce websites has become more and more intense. Poor performance has been recognized as a major obstacle to the success of an e-commerce website. A “buy” transaction that takes more than a few seconds may cause a customer to cancel the transaction, and the website may therefore lose the revenue generated by this transaction, or even lose this customer. Hence, providing a high quality of service has become more and more important.

The quality of service of e-commerce websites has usually been managed by the allocation of resources such as processors, disks, and network bandwidth. Tracking traditional performance metrics, such as response time, throughput and availability, is important in such a context. For example, QoS requirements can be controlled by dynamically tuning the configuration parameters, such as the maximum number of connections, number of threads and cache size, through tracking the site’s workload and the values of the performance metrics [22].

However, the metrics that are most important to online stores are revenue and profits. Therefore, the aim of resource management schemes for e-commerce websites should be optimizing business metrics instead of traditional performance metrics. Resource management strategies based on “business-oriented” policies, using priorities that are dynamically assigned as a function of the state of a customer and as a function of the amount of money in his/her shopping cart, have been investigated in [23]. However, that

study considered a single customer class, whereas this thesis focuses on systems with multiple classes of customers.

Both of the approaches described in the previous paragraph are useful. However, neither of them makes any differentiation between customer classes. For example, an e-commerce website may want to distinguish customers who have a higher probability of making a purchase. E-business is categorized into five categories [24]: 1) Business-to-Business (B2B); 2) Business-to-Customer (B2C); 3) Customer-to-Customer; 4) Government-to-Business; and 5) Government-to-Citizens. Customers can be categorized according to their registration status, payment for memberships, and purchasing history. Due to the increasing demands for faster response and higher availability from e-business websites, online companies must provide good performance to avoid loss of sales and customers. On any e-business website, it is important to guarantee the quality of service for customers in order to maximize revenue, and especially for important customers, such as heavy buyers, frequent partners.

This thesis focuses on the B2C type of e-business. The main objective of this thesis is to achieve a high revenue generated by a website and support its business goal, while providing higher responsiveness for the higher priority customer classes. This is achieved by a family of dynamic resource management policies. The priorities of different customer classes change dynamically according to the state a customer is in and the amount of time the customer has spent on the website.

1.2 Thesis Contributions

In this research a performance prototype was developed for analyzing the performance of various scheduling strategies. A workload generation tool based on an

open queuing model was constructed for generating the workload for the prototype. This workload generation tool uses a state transition graph called a Customer Behavior Model Graph (CBMG) to simulate a customer session. In this workload generation tool, a thread scheduler is introduced and manages the activities of every customer on the website.

Based on measurements made over a network of PCs, the performance of the prototype was investigated for various workload parameters. Using this measurement data, we captured the effect of different resource management policies on system performance and revenue generated for the online store.

The contributions made by the thesis are discussed briefly.

- Four priority-based scheduling policies are proposed:
 1. Policy based on a customer's class.
 2. Policy based on a customer's class and the shopping state of the customer.
 3. Policy based on a customer's class and the total time a customer spends in the online store.
 4. Policy based on a customer's class, the shopping state of this customer and total time spent by this customer in the online store.
- The relative performances of these four scheduling policies are analyzed in detail. The thesis presents insights into system behavior based on this performance analysis.
- The thesis shows that a significant performance benefit for customers in the higher priority classes is obtained through the use of the four proposed policies. In addition, a significant benefit with regard to revenue for the online store is produced by these four scheduling policies, as compared to a First-Come First-

Served policy. The results of the thesis are useful primarily for the designers of e-commerce systems.

1.3 Thesis Organization

Chapter 2 first presents a description of the existing scheduling policies from the literature. Several resource management policies for e-commerce systems are then discussed in detail. Finally, some profit-improving policies for e-commerce systems are presented.

In Chapter 3, the concept of the Customer Behavior Model Graph (CBMG) [25] is explained. Next, the limitations of resource management policies for e-commerce systems currently available in the literature are discussed. Four new priority-based resource management policies for e-commerce systems are then proposed.

The workload generation tool developed for this thesis is described in Chapter 4. First, brief descriptions of existing Web workload generation tools are given, after which the difference between open model and closed model is discussed. Finally, a detailed description of the implementation of the tool is presented.

The system architecture, workload and system parameters, and experimental design are presented in Chapter 5. In addition, the results of the validation of the workload generation tool are provided through the use of various approaches.

Chapter 6 presents the experimental results. The results demonstrate the impact of different priority-based scheduling policies on three performance metrics: the End-to-End Time of different customer classes, the Angry Factor of our system and the revenue earned by our e-commerce website. A detailed explanation of the experimental results is also given in this chapter.

Chapter 7 provides the conclusions of the thesis and suggests directions for future research.

Chapter 2 Literature Overview

This chapter discusses existing job scheduling and resource management strategies. A set of general scheduling policies is described first. A discussion of strategies for management of e-commerce systems is presented in the next section.

2.1 Existing Scheduling Policies

Scheduling is concerned with the effective management of a resource. The criteria of scheduling include: fairness, efficiency, throughput, and response time. It should be ensured that all processes get the CPU time they need and high system performance is achieved.

In this section, two categories of common scheduling policies will be discussed: arrival time-based scheduling policies and service time-based scheduling policies.

2.1.1 Arrival Time-Based Scheduling

The first group of algorithms to be discussed is arrival time-based. These algorithms are relatively easy to understand and implement. Customers enter the queue and are serviced according to their arrival times. A number of arrival-time-based scheduling strategies are presented next.

□ First Come First Served (FCFS)

FCFS is a very basic run-to-completion scheduling policy. Arriving jobs are placed in a first-in-first-out queue. The first job in the queue is blocked (as are all jobs arriving later) until at least as many resources as it requested are free. The job is then removed from the

queue and run. Whenever a job is completed and there are new resources available for assignment, the process is repeated for the next job in the queue [10].

❑ **Last Come First Served (LCFS)**

The last-come, first-served (LCFS) queuing algorithm is also a run-to-completion strategy as FCFS. The difference is that once the server has finished with a customer, the most recent arrival in the queue is processed next. This means that new entries are handled with very little wait. The obvious shortcoming, however, is that a customer that has been on the queue for a while may never get a chance at a server if new customers keep coming in [28].

❑ **Random Scheduling (RS)**

In Random scheduling each process is chosen randomly with an equal probability [16] from the ready queue and then executed. Sometimes a recent addition to the customer queue will be selected for a server, and at other times an older member of the queue will be chosen.

2.1.2 Service Time-Based Scheduling

One problem with arrival time-based scheduling policies is that if the customer or job in service has a large service demand, the waiting times for the other customers may be very long. This problem can be avoided by using a different family of scheduling policies, called service time-based scheduling policies.

❑ **Shortest Job First (SJF)**

The Shortest-Job-First (SJF) scheduling policy uses the service time of the customers in scheduling. The shorter the service time, the higher is the priority of a customer. The

SJF method is used to get around the main obstacle in FCFS, namely that the order of the queue can lead to long average waiting times.

SJF provides a better average response time than the arrival time-based non-preemptive scheduling policies [1]. The difficulty with the SJF policy lies in determining how long it will take for a server to process a customer.

❑ **Shortest Remaining Processing Time (SRPT)**

The SRPT scheduling policy selects a customer in the queue, which has the least remaining service time [11]. The policy is preemptive, so that if a new customer arrives in the system with shorter service time than that remaining for the customer currently in service, the scheduler immediately switches to service the newly arriving customer. The SRPT policy is provably optimal with respect to mean response time: it guarantees the lowest mean response time for the system as a whole [6] [34] [36].

❑ **Round Robin (RR)**

One of the problems of service time-based algorithms such as SJF is that they can lead to starvation of customer with large service times. Round Robin is a widely used scheduling algorithm that controls this unfairness of SJF. A small unit of time called a quantum is defined. All jobs are kept in a queue. New customers are added to the tail of the queue. The scheduler picks the first job from the queue and runs it for one quantum. If the service of the job is not completed at the end of the quantum, the job is pre-empted and added to the tail of the queue. The scheduler picks up the next job in the ready queue and runs it for a quantum. When all the customers in the queue have received a quantum of service, the first customer is scheduled again.

RR scheduling ensures that each customer will be processed fairly [39]. Although it does not explicitly use service times in scheduling, customer service times are implicitly used. The waiting time of a customer with RR scheduling is proportional to its service time.

2.2 Resource Management Policies

A number of resource management policies that are available in the literature is briefly discussed. Many of these strategies are used in the context of internet-based systems.

2.2.1 Cluster Reserves

Cluster Reserves is “a facility for achieving performance isolation between service classes hosted on a cluster. Cluster reserves are cluster-wide resource principals obtained by logically combining the resource principals on individual cluster nodes” [3]. Performance isolation means a specified proportion of server resources is reserved for a service, independent of the current demand for other services.

“A resource is any shared system entity needed for execution by a service and multiplexed by the system between the various services it hosts” [3]. CPU time is an example of resource. Resources will be allocated to a cluster reserve, and then allocated to resource containers on the individual nodes. One or more nodes in the cluster act as front-ends; these nodes act as points of contact for the clients. A client machine sends a request to one of these front-ends nodes across the Internet. The choice of a front-end node to contact can be made by asking the client which “mirror” server to contact.

“Resource containers are a first-class operating system abstraction for resource principals.” A resource container can be used to ensure effective performance isolation in single node Web servers. Using resource containers, a web server can associate, for instance, a client network connection, a server thread and a CGI process with a single resource principal that corresponds to the client request being served. This principal competes with other principals representing other client requests for server resources.

Figure 2-1: Cluster Reserves

Figure 2-1 shows a cluster with 3 nodes and 2 cluster reserves A and B. The resources assigned to A and B are dynamically split into resource assignments for the corresponding resource containers on each node. Resource allocations to the corresponding resource containers on each node add up to the desired allocation for the associated cluster reserve. For example, the resources assigned to A1, A2 and A3 add up to the desired allocation for cluster reserve A.

Cluster reserves are an effective solution for providing performance isolation in cluster-based Web servers. It extends existing mechanisms for performance isolation in single-node servers to a cluster environment. Hosting multiple Web services on a joint set

of cluster nodes can result in higher resource utilization and improved performance than the state-of-the-art approach of hosting different services on dedicated server nodes [3].

2.2.2 Priority – Based Scheduling Policy

The aim of a priority-based scheduling policy on a Web server is to provide a differentiated quality of service by assigning priorities to requests, based on the requested documents. Requests are assigned priorities based on which documents they are requesting [2]. In other words, customers who have paid for a high level of quality of service are provided faster Web access to their documents. There are two important aspects with regard to the scheduling policy [2]:

- ❑ **Sleep policy:** upon receiving a request, the scheduling policy must decide whether to process the request immediately, or to postpone the execution. A request may be postponed due to high system load, or the low priority of that request.
- ❑ **Wakeup policy:** the scheduling policy must also decide when a postponed request must be allowed to continue. The policy's scheduler will only allow a request to continue in the place of a completed request. When a request has been completed, the scheduling policy must decide which of the postponed requests, if any, should be selected to execute in its place.

These two policies are implemented by using thresholds for the maximum number of requests that can be handled concurrently in each priority level. Therefore, there are a fixed number of slots for each priority level, and each arriving request must either occupy a slot or wait in a queue until it is allowed to execute. Controlling the number of

processes that are allowed to run concurrently at each priority level is a simple and effective strategy for obtaining differentiated performance.

2.2.3 Integrated Resource Management

Integrated Resource Management provides cluster-based network services to many intranet or Internet users [37]. Inside those service clusters, several services are usually partitioned, replicated, aggregated, and then delivered to external clients through protocol gateways. For example, a service cluster may deliver search services to customers through Web servers and CGI gateways.

In a two-level request distribution and scheduling architecture, each external service request enters the service cluster through one of the gateways and is classified into one of the service classes, according to rules specified by the service providers. The gateway node then accesses one or more (in the case of service aggregation) internal services to fulfill the request. Inside the service cluster, each service can be made available to multiple nodes through replication. Each service node can process requests from all service classes. Each service request is directed to one of the replicated service nodes through a cluster-level request distribution. Figure 2-2 shows the architecture of a service cluster with the Integrated Resource Management policy.

At the cluster level, a class-aware load-balancing scheme is employed. This scheme evenly distributes requests for each class to all the servers.

There is a multi-queue (one per service class) scheduler inside each node. Whenever a service request arrives, it enters the appropriate queue for the service class it belongs to. When resources become available, the scheduler dequeues a request from one of the queues for service. Figure 2-3 shows the runtime environment of a node.

Figure 2-2: Service Cluster with Integrated Resource Management Policy [37]

Figure 2-3: Runtime Environment of a Service Node [37]

This integrated resource management framework for cluster-based network services is flexible in allowing service providers to express desired service qualities based on the service response time. At the cluster level, the scalable centralized request distribution architecture shown in Figure 2-2 ensures a prompt and smooth response to service demand spikes and server failures. Inside each node, an adaptive multi-queue scheduling scheme is employed to achieve efficient resource utilization and to provide service

differentiation. This framework is used primarily for resource management for single-tier services.

2.2.4 Integrated Real-time Resource Scheduler (IRS)

IRS is based on a two-level resource-scheduling architecture, which consists of a global scheduler and a set of local schedulers, each corresponding to a particular resource.

The main focus of IRS is to advocate the importance of an integrated framework for the allocation and scheduling of multiple resources in a real-time system in order to support application level performance guarantees [12]. Figure 2-4 (adapted from [12]) illustrates the software architecture of the IRS policy.

Real-time applications are programmed using an IRS API and linked with a user-level IRS library that uses a set of IRS specific system calls to interact with the kernel. The *global scheduler* is responsible for making sure that a task's dependencies are all satisfied before it is placed in the request queue of the corresponding resource. *Local schedulers* manage individual resources and perform scheduling decisions based on both task deadlines and resource utilization efficiency. To the *global scheduler*, individual resource schedulers are black-box building blocks whose specific scheduling algorithm is completely hidden.

The *CPU scheduler* in Figure 2-4 uses a simple Earliest Deadline First scheduling policy for real-time tasks and a priority based scheduling policy for non-real-time tasks. The disk I/O system uses a Deadline Sensitive SCAN Algorithm [13]. The *admission controller* makes admission decisions and performs task deadline assignments when a new application registers itself as a real-time process. The *resource usage monitor* keeps

track of the resource consumption of the tasks in order to detect applications that are misbehaving.

Figure 2-4: The Software Architecture of the IRS Policy

IRS is a two-level, real-time resource scheduler framework that respects task dependencies when dispatching tasks, performs the coordination of local resource schedulers, and supports the ability to dispatch multiple concurrent tasks to local schedulers before blocking. The framework allows the local resource schedulers to be any deadline-based schedulers.

Currently the IRS framework is applied to provide QoS guarantees for Web server clusters, where the clients can request not only throughput guarantees (requests/sec) but also per-request response time guarantees [12]. Serving each Web request typically involves access to multiple resources such as CPU, disk and network link. IRS is an essential building block for providing bounded response time guarantees in such a system,

where it is important to maximize the number of clients admitted by the Web server cluster.

2.3 Profit-Improving Policies for E-Commerce Systems

There are many resource management policies that manage the allocation of resources, such as the CPU, disks, and networks, in order to improve the overall performance of the e-commerce system. However, those policies do not focus on the most important business-oriented performance metrics: revenue and profits. Therefore, e-commerce systems need resource management policies that will improve their profits and revenue.

2.3.1 Usage-Based Web Personalization

As the number of on-line shoppers continues to increase, Web Personalization is becoming more and more important. Customers who access e-commerce Web sites come from different backgrounds, with different expertise and preferences, so Web sites have to provide information about their products (e.g. recommendations about a specific product) in such a way that the customer's needs are best matched.

Web personalization includes several procedures as follow [26]:

1. Modeling of Web objects (pages, etc.) and users.
2. Categorization of Web objects and users.
3. Matching objects and/or users.
4. Determination of the set of actions to be recommended for personalization.

One approach to Web personalization is based on collaborative filtering, which relies heavily on a user profile built through inputs from users [18] [14] [35]. Another

approach to personalization deals with the specific aspects of Web usage mining for the purpose of automatically discovering user profiles.

Usage-Based Web Personalization makes heavy use of Web data mining techniques, which make the personalization process automatic and dynamic. In this way the recommendations made to a user can truly match the user's needs. This approach is described in the architecture shown in Figure 2-5.

The overall process of usage-based Web personalization can be divided into two components [26] that are described next.

❑ **Offline Component**

The offline component is comprised of the data preparation tasks, which result in a user transaction file, and the specific usage mining tasks, which involve the discovery of association rules and the derivation of URL clusters. These two components are shown in the batch process in Figure 2-5.

❑ **Online Component**

The online component, shown in the online process in Figure 2-5, uses the frequent item sets and the URL clusters to provide dynamic recommendations to customers, based on their current navigational activity.

The online component is comprised of a recommendation engine and the HTTP server. The Web server keeps track of the active customer session as the customer browser makes HTTP requests. This can be accomplished by a variety of methods, such as URL rewriting [21], or by temporarily caching the Web server access logs. The recommendation engine considers the active user sessions in conjunction with the URL clusters and the discovered association rules to compute a set of

recommended URLs. The recommendation set is then added to the last requested page as a set of links before the page is sent to the customer's browser.

Recommendations based on this personalization process categorize web objects as well as users and match customers with the products they are likely to buy. Therefore, the profitability of the e-commerce website should be increased.

Figure 2-5: General Architecture for Usage-based Web Personalization [26]

2.3.2 Profit-Oriented Feedback Control System

The Profit-Oriented Feedback Control System automates the admission control decisions by controlling the loss of revenue due to rejected work and the penalties occurred if admitted work has a long response times. Therefore, this system can directly optimize profit for e-commerce Web sites and is robust to changes in workload, the target system and profit model (the underlying service level agreement) [8].

A fuzzy controller is employed to construct a profit-maximizing feedback loop, which avoids skill-intensive alternatives such as building a specific queuing model or handcrafting a proportional integral controller. Figure 2-6 (adapted from [8]) shows the internals of the profit-based fuzzy controller.

Figure 2-6: Architecture of the Profit-Oriented Feedback Control System [8]

The feedback loop includes a profit model component that computes the profit for each sampling interval [8]. The inputs that are used to compute the profit are the throughput (completed transactions) and the response time. The profit is computed and

maximized based on sampling intervals. The sampling interval is not too long, so the controller can adapt quickly to changes in the operating environment.

Another component in the feedback path is a differentiator (shown in Figure 2-6), which outputs the change in profit value between the current interval and the last. The feedback is used to change the MaxUsers value (the limit on the number of users allowed to connect to an application). The fuzzy controller automatically adjusts to changes in the profit model.

The fuzzy controller's actions are guided by a set of fuzzy rules. The value of the MaxUsers is optimized and the fuzzy rule set implements a hill-climbing procedure to find this optimal value. The rules are stored in a rule database (or rule base), a part of the controller, which is shown inside the fuzzy controller in Figure 2-6.

The fuzzification component of the fuzzy controller (bottom of Figure 2-6, inside the fuzzy controller) converts an input number into a linguistic equivalent. The Fuzzy inference mechanism evaluates fuzzy rules and yields a linguistic output [29]. The defuzzification component is complementary to fuzzification and converts the linguistic output yielded by fuzzy inference to a numerical value. Finally, the output of the fuzzy controller is an adjustment to MaxUsers for the next sampling interval.

Workloads change frequently in e-commerce environments, and profit models must often be adjusted as a result of service level agreement negotiations (which may be done automatically in Web service environments). The number of completed transactions determines service providers' revenues, and there are penalties for service-level agreement violations (e.g. exceeding response time guarantees). Thus, situations arise in which service providers must make trade-offs between losing revenue (e.g. as a result of

admission control that denies access to some customers) and incurring penalties (e.g. because admitted work cannot be completed within the service agreement-dictated response time).

By using a fuzzy control system, the number of customers allowed in the system (workload) can be adjusted dynamically, as a result of which the number of completed transactions can be maximized. This leads to the maximization of profit.

2.3.3 Recommender System

E-commerce websites provide products that may well match customers' interests, and e-commerce websites are also eager to get their customers to browse for more products that may attract them. A recommender system will recommend products that customers might be interested in. Recommender systems are being used by many of the largest e-Commerce websites nowadays.

By using recommender systems, the products can be recommended based on the top overall sellers on a site, the demographics of the customer, or an analysis of the past buying behavior of the customer as a prediction for future buying behavior. Broadly, these techniques fall under personalization on a site, because they help the site adapt itself to each customer. Recommender systems automate personalization on the Web, enabling individual personalization for each customer.

Recommender systems enhance e-commerce sales in three ways [33]:

- ❑ Converting browsers into buyers: Visitors to a website often look over the site without ever purchasing anything. Recommender systems can help customers find products they wish to purchase.

- ❑ Cross-sell: Recommender systems improve cross-sell by suggesting additional products for the customer to purchase. If the recommendations are suitable, they may increase the average order size.
- ❑ Loyalty: In a world where a site's competitors are only a click or two away, gaining customers' loyalty is an essential business strategy [30] [31]. Recommender systems improve loyalty and create a value-added relationship between the site and the customer. Sites invest in learning about their users, and use recommender systems to operationalize that learning. Customers repay these sites by returning to the ones that best match their needs. The more a customer uses the recommendation system – teaching it what they want – the more loyal they are to the site. “Even if a competitor were to build the exact same capabilities, a customer... would have to spend an inordinate amount of time and energy teaching the competitor what the company already knows.” [17]. Finally, creating relationships between customers can also increase loyalty.

There are three recommendation techniques used by most e-commerce sites that are discussed next.

1. Attribute-Based Recommendation:

Attribute-based recommender systems recommend products to customers based on the syntactic properties of the products. An example of an attribute-based recommendation would be when an e-commerce site responds with a recommended list of three related books after a customer has done a search for a JAVA book. Attribute-based recommendations are often manual.

2. Item-to-Item Correlations:

Item-to-Item Correlation recommender systems recommend products to customers based on a small set of products the customers have expressed interest in. The system, for instance, may recommend complementary products in order to increase the order size after a customer has placed a few products in his shopping cart. Item-to-Item Correlation recommender systems can be automatic, if they are based on observations of the customer's unchanged behavior. They can also require some manual effort, if the customer must explicitly type in several items of interest in order to generate a recommendation.

3. User-to-User Correlations:

User-to-User Correlation recommender systems recommend products to a customer based on correlations between that customer and other customers who have purchased products from the e-commerce site. This technology is often called "collaborative filtering," because it originated as an information-filtering technique that used group opinions to recommend information items to individuals [32]. User-to-User correlation recommender systems are close to being automatic, since the recommendations themselves are generated automatically by the system. The system does have to learn over time from customers, however.

Recommender systems are a keyway to automating mass customization for e-commerce sites. They will become increasingly important in the future. By recommending a set of products that a specific customer is likely to buy, these systems attempt in improving the profit of the website.

Chapter 3 Scheduling Policies for E-Commerce Systems

3.1 Customer Behavior Model Graph of Different Classes of Customers

3.1.1 Workload Characterization

Customers interact with e-commerce sites through a series of consecutive and related requests, which are called sessions. Some examples of requests an online shopper might make are: browse, search, select, add to shopping cart, user registration and pay. In addition, different customers exhibit different navigational patterns. A Customer Behavior Model Graph (CBMG) is a state-transition graph that is used to describe the behavior of groups of customers who exhibit similar navigational patterns [25]. The graph has one node for each possible state (e.g. home page, browse, search, select, add and pay). A probability is assigned to each transition. Different customers may be characterized by different CBMGs in terms of their transition probabilities.

As shown in Figure 3-1, two steps are required to obtain a CBMG. In the first step, a conventional HTTP log is taken as input and a session log is generated. Figure 3-1 shows that the request log is scanned sequentially. A determination is made for each request in this log as to whether it constitutes a request that starts a new session, or a request from an existing open session. A request is considered to start a new session if there is no open session for that customer, or if the last request from this customer occurred more than a specified amount of time ago. If it is a request for a new session, any existing open session for the same customer must be closed. If the request relates to an existing open

session, the relevant data for that customer are updated. Every time a session is closed, a new entry is written to the session log.

Figure 3-1: Workload Characterization Methodology

In the second step a clustering analysis is performed on the session logs produced in the first step. This results in a synthetic workload that consists of a relatively small number of CBMGs. These CBMGs are used as a compact representation of the e-commerce workload. The centroid of a cluster determines the characteristics of the CBMG. Any number of clustering algorithms can be used. The goal of clustering algorithms is to identify natural groups of customers, based on similar navigational patterns. A detailed discussion of the process of CBMG generation is provided in [25].

3.1.2 CBMG of Different Classes of Customers

A CBMG has $n+1$ states. State 0 is always the Entry State, while state n is always the Exit State. Transitions may occur among these states, with certain probabilities. For example, a transition from state k to state j may occur if a customer sends a request, or if a customer leaves the website from any state (except the Entry State) to the Exit State. A think time can be associated with each type of state in the CBMG. For example, the think time associated with the state “select” indicates how long it takes a customer to submit the next request to the server. In this thesis we have assumed a customer “thinks” in the “login”, “select”, “confirm” and the “pay” states.

In this study, we distinguish between three customer classes: Gold, Silver and Bronze. Gold customers are the highest priority customers. They typically have the highest probability of buying if they see a product that interests them at a suitable price. Silver customers are assigned a medium priority level. They are less likely to buy products than Gold customers. Bronze customers have the lowest priority. They typically use the website to find out about existing products, such as new books or best fares, but most of the time do not buy anything from the website. Each customer class is characterized by a separate set of transition probabilities. Figure 3-2 shows the CBMG that can be used for the three classes of customers. Depending on the system, each class may be characterized by a separate or the same CBMG.

The five states shown in Figure 3-2 are adequate for describing customer behavior in our system. Other systems may have more states. The nodes of the CBMG, represented by ellipses, describe each state a customer is in during a visit to the e-commerce site. The arrows connecting the states indicate possible transitions between the states. A half-circle

with an arrow denotes "thinking" (described in more detail in Section 4.6.1) by the customer in a given state. The Entry State is a special state that immediately precedes a customer's entry to the online store. This state is shown as part of the CBMG for convenience of modeling, and does not correspond to any action carried out by the customer. The Exit State refers to customers leaving the site from any state. It is possible to transition to the Exit state from all states except the Entry state.

Figure 3-2: CBMG for Three Classes of Customers

P_i refers to the transition probabilities between states, where $i = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. The transition probabilities are determined by the input workload parameters. For the sake of simplicity, a uniform distribution is used to control the transitions between states by comparing them with the transition probabilities provided.

A customer in the experiments described in this thesis can get "angry" if the time spent in the system is too long. We have assumed that such an angry customer exits the system without completing the purchase. This issue is discussed further in Section 4.6.2.

3.2 Limitations of Current Resource Management Policies

E-commerce Web sites sometimes face very high traffic and the server capacity is not able to handle all the customers trying to enter the site. Usually, all customers who attempt to visit the site during these peak periods experience a poor quality of service, typically indicated by long waiting times to interact with the Web server and refused connections. Current resource management policies do not make any distinction between customers and do not provide differentiated quality of service. For instance, an online store may want to offer the best quality of service to customers who really intend to purchase something. However, with the systems available at present, the resources are shared equally between customers who are making a purchase and customers who are just browsing.

The most important goal of an online store is maximizing the revenue generated by the website. General-purpose operating systems and Web servers do not provide adequate support for the resource management of e-commerce sites. They only consider throughput in assigning resources to multiple processes, and ignore the relative importance of the processes themselves and their meaning with regard to the goals of the business goals. Therefore, resource management policies should be designed to support business goals. Resources should be allocated on an economic basis.

A number of resource management policies that attempt to improve the revenue generated by an e-commerce website are investigated in this thesis. Descriptions of these strategies are provided in the following sub-sections.

3.3 Policy Based on Customer's Class (Ponly)

This is the simplest policy. In this policy, Gold customers have the highest priority, and Bronze customers have the lowest. A similar policy was investigated in [20]. As shown in Figure 3-3, the priorities of all customers are assigned once they enter the system and will never be changed during their visit to the website. Within a given class, a First Come First Served (FCFS) scheduling policy is used.

Figure 3-3: Policy Based on Customer's Class

3.4 Policy Based on Customer's Shopping State

This policy will adjust a customer's priority in the system based on the state a customer is visiting in the CBMG. All customers enter the system with the same priority, and are characterized by the same CBMG. When a customer visits a "pay" state, and sends a request associated with making a payment, this customer will be given the highest priority in the system, and other customers will have to wait until this customer has

finished the payment request. If two customers are in the “pay” state, they will be served on a FCFS basis.

3.5 Policy Based on Customer's Total Time Spent in System

In this policy, all customers enter the website with a high priority. Real logs from an online store were analyzed in [23], and an interesting pattern was observed: the longer a customer stays in a Web store, the less likely it is that this customer will buy an item. Therefore, after a customer's session length, that is the total time the customer has spent in the system has exceeded a threshold T_1 (see Figure 3-4), the customer's priority is lowered to medium. This customer will remain at this priority level until the session length reaches the second threshold $T_1 + T_2$. When the second threshold is reached, the customer's priority is reduced to “low level”. Notice that if T_2 is long enough, the priority policy can be reduced to a two-level priority policy. In addition, if T_1 is long enough, a customer will be assigned the same priority all the time. Thus, by varying the values of T_1 and T_2 , one can generate several priority policies.

Figure 3-4: Policy Based on Customer's Total Time Spent in System

3.6 Integration of Different Priority-Based Policies

3.6.1 Class-based and State-based (Sonly)

This policy combines the class-based and state-based policies into one policy. Gold customers have a high priority; the Silver customers have a medium priority, and the Bronze customers the lowest, as in the class-based strategy described in Section 3.3. However, when a customer is about to buy something, i.e. enter the “pay” state in the CBMG, no matter which class this customer comes from, he/she will be given the highest priority in the system, even higher than that of the Gold customers. For example, Gold customers have a priority of 30; Silver customers 20; and Bronze customers 10. The larger the number, the higher the priority. When a Bronze customer enters the “pay” state, the priority of this customer will be increased to 40, so that this pay request will be answered fastest.

Figure 3-5: Policy Based on Class and State of Customers

3.6.2 Class-based and Time-based (Tonly)

This policy combines the class-based policy and time-based policy. As described in Section 3.5, customers' priorities will be changed based on how much time they spend on the website. The longer the time spent, the lower the priority. Each class has three levels of priority: high, medium and low. When a customer enters the system, he/she has the highest priority of his/her class.

A Gold customer enters the Web store with a priority of 90. After T_1 , the first time-based threshold, he/she will have a priority of 60, which is the medium priority level within the Gold class. After $T_1 + T_2$, the sum of the first and the second time threshold, customer's priority will be changed to 30, which is the lowest level priority within the Gold class. The priorities of Silver and Bronze customers are changed in a similar manner (see Figure 3-6). In Figure 3-6, the three bold-bordered boxes represent the three priority levels. Within each of these boxes, there are three classes of customers. The numbers associated with the different class of customers refers to their priorities in the system. For example, in the box at the lowest priority level, Gold customers have a priority of 30, Silver customers 20 and Bronze customers 10. A customer may exit the Web store from each priority level.

3.6.3 Class-based, State-based and Time-based (T&S)

This policy is more dynamic than the one described in Section 3.6.2. Customers' priorities will be changed based on their class, how much time they spend on the website as well as which state they are in. The time policy aspect remains the same as described in Section 3.6.2. Note that a policy based on time and state is discussed in [23]. However,

while the T&S policy discussed in this thesis that deals with multiple customer classes, the policy described in [23] handled only a single customer class.

Figure 3-6: Policy Based on Class and Time Customer Spends in System

What is different with the T&S policy is that as soon as a customer adds an item to his/her shopping cart, i.e. visits the “confirm” state in the CBMG, his/her priority will be increased to (or maintained at) the highest priority available to his/her class (see Figure 3-7). For example, if a Bronze customer’s priority has already been reduced once, and is at 40, if this customer adds one item to his/her shopping cart, his/her priority will be increased to 70, which is the highest level available to a Bronze customer. In addition, as soon as a customer enters the “pay” state, this customer will have the highest priority of

100 in the whole system, even higher than a newly entered Gold customer, who has a priority of 90.

Figure 3-7: Policy Based on Class, State and Time

As described in Section 3.6.2, customers may exit the online store at any of the priority levels.

In this thesis we have investigated the performance of the Ponly, Sonly, Tonly and T&S policies. A detailed discussion is provided in Chapter 6.

Chapter 4 Performance Prototype

The performance prototype used in this thesis includes a workload generation component and a resource management component.

A synthetic workload generated by an artificial tool, the characteristics of which are similar to those of the real workload and can be applied repeatedly in a controlled manner, has been developed and used for this thesis. The reasons for using a synthetic workload are: 1) it is a model of the real workload; 2) the workload can be easily modified without affecting operations; 3) it can be ported to different systems with ease, due to its small size; and 4) it can have built-in measurement capabilities [15]. In the next three sections, three existing synthetic workload tools are discussed. Why these tools cannot be used directly for this research is explained. After that, a performance prototype for our CBMG-based e-commerce Web application is presented in Section 4.4.

4.1 The Httpperf Workload Generation Tool

Httpperf is a tool that provides flexible facility for measuring Web server performance. It supports both HTTP/1.0 and HTTP/1.1 [9] protocols and offers a variety of workload generators. Httpperf consists of three parts, which are as follows [27].

- The core HTTP engine

The HTTP engine handles all communication with the server, connection management, HTTP request generation and the reply from the server.

- Workload generation

The workload generator initiates appropriate HTTP calls at certain times so that a desirable workload is generated to the server.

- ❑ Statistics collection

Statistics collection is responsible for measuring various quantities and producing performance statistics.

The current version of Httpperf supports two kinds of workload generators: one is a request generator that creates sessions deterministically and at a fixed rate, and the other is a URL generator that simply generates the specified sequence of URLs repeatedly.

The Httpperf tool is suitable for measuring Web server performance. However, it does not work well for capturing the performance metrics (e.g. the revenue-per-second of a website) related to the e-commerce application under review in this thesis. In addition, Httpperf cannot be used to demonstrate customers' behavior based directly on a CBMG model. Therefore, we cannot use Httpperf as a standalone tool for this research.

4.2 Scalable-Client (S-Client) Generation Tool

The most basic operation of the S-Client Generation tool is to generate heavy concurrent traffic with a temporal behavior similar to that of real Web traffic [4]. A number of S-Client processes run on a client machine, each S-Client consisting of a pair of processes connected to server. One process in the S-Client is responsible for generating HTTP requests at a certain rate and with a certain request distribution.

After the S-Client sends several non-blocking requests to the server, the S-Client checks if any of these requests has completed or a certain time has elapsed after a request has been sent. If any of these requests has completed, the S-Client hands off this connection to the other process of the S-Client, and then initiates another connection to the server. Otherwise, the S-Client closes the request's connection to the server, and then initiates another connection.

The S-Client allows for the generation of request rates beyond the capacity of the server, with a reasonable number of clients, by using non-blocking connects and a closing socket if no connection is established within a certain time. In addition, the S-Client ensures that the generated request rate is independent of the rate at which the server handles requests. As long as the client machine is not saturated, the S-Client can generate as many requests as the operating system allows. However, the S-Client does not capture any performance metric; in addition, we cannot use the S-Client to generate requests, which follow a CBMG model. Therefore, we cannot use S-Client as a standalone tool for this research.

4.3 Scalable URL Reference Generator (SURGE)

The Scalable URL Reference Generator (SURGE) from Boston University is a WWW workload generator [7]. It generates Web traffic and simulates the behavior of real Web users. The requests generated by SURGE have six statistical properties:

- ❑ **Server File Size** – The server file system must be able to handle highly variable file sizes.
- ❑ **Request Sizes** – The file size transferred over the network from the server should match the empirical measurements.
- ❑ **Popularity** – The relative number of requests made to individual files on the server is another property of the workload. Popularity measures the distribution of requests on a per-file basis.
- ❑ **Number of Embedded Object References** – This parameter refers to the characterization of the number of embedded object references in Web objects, and affects the distribution of Active OFF periods. An Active OFF period

corresponds to the processing time the browser spends parsing Web objects and preparing to start new TCP connections [7].

- ❑ **Temporal Locality** – This refers to the likelihood that once requested, a file will be requested in the near future. The distribution of the request stack distance is used as a measure of the temporal locality.
- ❑ **Off Times** – The distribution of the inactive OFF time, which reflects the bursty nature of an individual Web user's requests, and the active OFF time, which reflects the transfer of Web Objects.

SURGE generates a sequence of URL requests that matches the distributions specified for each of their attributes. It provides a sample database of documents from which to generate requests. Therefore, URLs cannot be specified, and we cannot use SURGE to generate a CBMG-based request. In addition, similar to the two earlier workload tools, SURGE does not capture the performance metrics related to this study. So we cannot use it as a standalone tool for this research.

A number of other workload generation tools, such as SPECWeb [41] and WebStone [38], are available in the literature, but none of them directly supports a CBMG model. A workload generation tool that can generate customer arrivals at a controllable rate, model customer behavior specified by a CBMG and compute the performance metrics of interest is needed for this research. Since none of the tools described earlier can achieve this without alteration of their source codes, a CBMG-based workload generation tool was developed for the performance prototype used in this research. A discussion on the implementation details of the performance prototype that includes the tool is provided next.

4.4 CBMG-Based Performance Prototype

4.4.1 Closed Model vs. Open Model

There are two kinds of models for systems: the closed model and the open model [5]. Closed models have a fixed number of clients. An open model allows requests to arrive, go through the various resources (e.g., Web server, application server and database server in an e-commerce site) and leave the system. An open model thus represents an e-commerce site with no limit on the number of customers who typically arrive, obtain services, and leave the system.

We have chosen an open model for this study, since the number of total participants required by the closed model is not known *a priori* and can vary from time to time.

4.4.2 Performance Prototype Description

A CBMG-based performance prototype has been developed for this research. The prototype uses the HTTP/1.1 protocol to offer different workloads based on the CBMG. It keeps track of a number of performance metrics, such as End-to-End Time and Revenue per second, to gain insights into performance of the scheduling policies. A number of parameters, such as the different transition probabilities, Angry Time and Think Time, are used to control the behavior of customers.

A multi-threaded Java environment was chosen to implement the CBMG-based open model performance prototype that deployed multiple threads (see Figure 4-1).

There are two components to this performance prototype: Request Generating Component and Request Managing Component.

- ❑ Request-Generating Component: The Request Generating Component generates a synthetic workload with controllable parameters. This is responsible for generating all the requests required in a certain experiment. The inter-arrival times between customers follow an exponential distribution.

Figure 4-1: Open Model Performance Prototype

- ❑ Request Managing Component: This component manages the system resources and all the activities of every single customer in the system, including running, suspending, resuming and clearing customers who have stayed in the system for more than a certain time (called the Angry Time described in Section 4.6.2).

These two major components will be discussed in detail in the next two sub-sections.

4.4.2.1 Request Generating Component

□ Main Thread

The main thread shown in Figure 4-1 is used to control the generation of client requests. The tasks of the main thread are to: 1) generate one request thread at a time, 2) give a specific global ID to every thread, 3) assign a request thread to a customer class, 4) compute and output average End-to-End Time for each customer class and the whole system, 5) output total experiment time and the total number of arrivals.

Each request is handled by a separate thread. When a request arrives, a request thread is created and the request is assigned to a customer class (Gold, Silver or Bronze) characterized by a specific CBMG. In this thesis, we assume that one third of the total number of customers that arrive belong to a given class. Since threads dispatched at the beginning, when the system has not yet reached a steady state, may run on a lower system load than those dispatched later, they may generate unreliable results [15]. For this reason we have excluded them from our results. The main thread sets an ID for each thread, and those threads that have an ID number greater than a specific value become “valid threads”. Note that only the requests corresponding to the valid threads are used in the performance computations.

The upper part of Figure 4-1 illustrates how the main thread defines the threads that are valid for measurement. The threads dispatched at the beginning and at the end by the main thread are not valid and do not contribute to the results. The number of invalid threads dispatched at the beginning is set to a fixed number according to system load. In all our experiments, the number is set to 60 for handling different system loads.

The number of valid threads is an input parameter of this tool. The default value of the number of valid threads is 600. After the invalid threads have been dispatched at the beginning, the next consecutive threads are valid. Once the 600 valid threads have been dispatched, the main thread will keep dispatching invalid threads until all those 600 valid threads have exited the system, in order to maintain a steady running environment for those valid threads. If the system load is higher, the time needed to complete the experiment increases, and the number of invalid threads dispatched at the end can increase as well. The rationale behind the choice of the number of valid threads is presented in Section 4.9.

□ Request Thread

The request thread shown in Figure 4-1 is created by the main thread and controlled by the Request Managing Component. Each request thread corresponds to a customer: a customer may connect to the Web server several times, and therefore, each request thread handles several HTTP requests. Since different customers have different navigational behavior when accessing the Web site, the customer behavior models, i.e. CBMG, must be built into the request thread.

The implementation of the CBMG is performed in the Request Thread. Random numbers generated in the request thread are distributed uniformly within (0, 1). They are used to effect a state transition. The transition probabilities: P_0 , P_1 , P_2 , P_3 , P_4 , P_5 , P_6 , P_7 , P_8 , and P_9 in CBMG (see Section 3.1.2), are the input parameters of the tool. Note that a thread may go back to a prior state, or may exit the system entirely, according to the CBMG.

4.4.2.2 Request Managing Component

The request manager, shown in Figure 4-1, controls the activities of each request thread. The tasks of the request manager are to: 1) compare the priority of a new request thread with the requests already in the system; 2) start a request thread with the highest priority in the system; 3) interrupt all other request threads and put them into a ready queue; 4) clear all threads which have stayed in the system more than a certain time (Angry Time described in Section 4.6.2).

When a new request comes in, the request manager compares its priority with the priority of the thread running in the system. If the new request has a higher priority than the one that is running, the request manager will interrupt the running request thread and put it into a ready queue, then start the new request. Once a request thread exits the system, the request manager will check all threads in the ready queue, then start or resume the one with the highest priority.

After starting each request thread, the request manager will check the accumulated End-to-End Time of all threads in the ready queue. If the accumulated End-to-End Time of a request thread exceeds a certain amount of time (its angry time), the request manager will clear this request thread out of the system. This signifies the departure of an angry customer from the website. The angry time is an input parameter of this tool, and its default value is 30000 ms.

The implementation of the request management component of the CBMG-based Performance Prototype used in this thesis is discussed further in the following sections. Note that this prototype uses a single level of request managing component. On a real system an additional front-end component that performs the mapping of a customer

request to an appropriate priority level is required. If the system is multi-tiered and an end-to-end priority is to be associated with requests, a request manager is to be incorporated at each tier and the thread priorities are to be assigned on each node hosting the tier. A customer request has to be piggybacked with the priority value so that the same priority level used in one tier is passed on to the next.

4.5 Apache-Based Server

The request management component of the performance prototype is based on an Apache Web server [40]. Apache was originally based on code and ideas found in the very popular HTTP server, NCSA (the National Center for Supercomputing Applications). Apache has been the most popular Web server on the Internet since April 1996. The December 2000 Netcraft Web server survey found that over 60% of Web sites on the Internet make use of Apache [19]. Due to this wide usage, Apache (version 2.0.40) was chosen as the Web server for our prototype. Apache has a number of configuration parameters the suggested default values of which are given in [40]. Table 4-1 presents the Apache Web server configuration parameters that are changed from their default values. The other parameters are held at the suggested default values in this study. The changes were made at various points in time during prototype development to achieve stable behavior of the prototype at all system loads. Tuning of the parameters with respect to the final version of the prototype was not performed. This is not expected to affect the conclusions regarding the relative performance of the scheduling policies because each scheduling policy was evaluated with the same set of configuration parameters.

Table 4-1: Apache Server Configuration Parameters

Configuration parameters	Default Value	Setting Value
StartServers	5	20
MinSpareServers	5	20
MaxSpareServers	10	25
MaxClients	150	8000
MaxKeepAliveRequests	100	0 (unlimited)
KeepAliveTimeOut	15 seconds	150 seconds

A short discussion of the Apache server parameters shown in Table 4-1 is presented next.

□ **StartServers, MinSpareServers and MaxSpareServers**

StartServer is the number of server processes that are spawned at the beginning. MinSpareServers and MaxSpareServers represent the lower and upper limit respectively of the number of spare server processes. The number of processes waiting for requests is compared with these values within the Web server to ensure that enough server processes are available to handle the current load. Because increasing these values will increase the overhead within the server, the performance will differ with different settings [28]. In order to ensure that all our experiments could be handled under the same configuration setting, we increased their values over the defaults set by Apache. These values resulted in stable systems at various workload conditions in our experiments.

□ **MaxClients**

MaxClients limits the total number of server processes running (i.e., the number of clients that can simultaneously connect to the servers). Clients will be locked out if more servers than this are needed.

□ **MaxKeepAliveRequests**

MaxKeepAliveRequest denotes the maximum number of requests allowed during a persistent connection. It was set to 0 in our experiments to allow for an unlimited number of requests [40].

□ **KeepAliveTimeOut**

KeepAliveTimeOut is the number of seconds the server waits for the next request from the same client on the same connection. When the value was too small, an error was encountered in the experimental set-up. It was increased to 150 seconds to avoid the error.

The workload generation tool and Web server run under Red Hat Linux 6.2, kernel 2.2.16-3 on 266MHz Pentium II PCs with 64MB of RAM. The PCs are inter-connected by a 100MB Ethernet LAN. The PCs and LAN form the “quiet network” in the Real Time and Distributed System Lab, where experiments run without interference from other users.

Each of the scheduling policies described in Chapter 3 were implemented in the performance prototype. To assess the impact of the scheduling policies described in Section 3.6, we simulated an electronic bookstore. The activities of a customer in such a bookstore can be modeled by a CBMG used in this thesis (see Figure 3-2 for example). Upon arrival the customer is in the Login state and proceeds to the Select state for choosing a book. The book is added to the shopping cart in the Confirm state. The Pay state is visited by the customer when payment is made. While in the Thanks state the customer receives a “thank you” message from the web site. The Exit state is visited when the customer departs the system. Websites that can give rise to more complex

CBMG's are possible, but the CBMG used in this thesis is sufficient for the qualitative nature of the research presented in this thesis.

The distribution of book prices is a truncated Gaussian distribution. The books range in price from \$5 to \$100, and the mean value is \$50. The price range was obtained by analyzing the results of many searches performed in various existing online bookstores [23].

4.6 Workload Parameters and System Parameters

A number of different parameters were used to control the workload and the system configuration. These parameters are discussed in the following sub-sections.

4.6.1 Workload Parameters

□ Arrival Rate (customer/second) (AR)

We assume that customer arrival follows a Poisson distribution. The Arrival Rate in our experiments has four levels: 0.1 customers/second, 0.2 customers/second, 0.3 customers/second and 0.4 customers/second.

□ Number of Valid Threads (V)

Only valid threads have been measured in our experiments, as described in Section 4.2.2. Valid threads do not include arrivals that are dispatched at the beginning and at the end of an experiment. This ensures that measured threads always come from a system in a steady state. Since the system in all experiments achieves steady state with 600 valid threads, the number of valid threads is set to 600 in all the experiments.

□ Think Time (Th)

The customer may spend time thinking, inputting data, or clicking a button in one state before going to the next state in the CBMG. Different classes may have different think times. Within a class, the think time is assumed to be the same in each state of the CBMG. The default think times for each customer class are: Gold-800 ms, Silver-1000ms and Bronze-1200 ms.

□ **Pi**

These parameters are probabilities of transition from one state to another state, where i is an integer from $\{0,1,2,3,4,5,6,7,8,9\}$. The probabilities that Gold, Silver and Bronze customers will visit different states in the CBMG are different. For example, Gold customers typically visit pay states more often than the customers in other classes. Table 4-2 shows the default transition probabilities for three classes of customers.

Table 4-2: Transition Probabilities

	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
Gold	0.9	0.9	0.8	0.1	0.1	0.8	0.1	0.1	0.1	0.1
Silver	0.9	0.8	0.6	0.3	0.1	0.8	0.1	0.2	0.1	0.1
Bronze	0.9	0.7	0.4	0.5	0.1	0.8	0.1	0.3	0.1	0.1

Here we assume that customers may go back to the previous states.

4.6.2 System Parameters

□ **Priority Setting (P)**

There are three priority levels in each customer class: high, medium and low. At a certain priority level, Gold customers usually have the highest priority at each level; Bronze customers have the lowest at each level and Silver customers have medium

priority. Except for one situation described in the next paragraph, a Gold customer will have a higher priority than a Silver customer at the same level; also, a Silver customer will always have a higher priority than a Bronze customer at the same level.

The bigger the priority value, the higher the priority. Table 4-3 shows the priority settings for each class. Note that in the Sonly and T&S policies, if a customer is in the “pay” state, this customer has the highest priority of 100 irrespective of its class. Thus a Bronze customer in the pay state for example, will be assigned a priority that is even higher than a Gold or Silver customer who is not in the “pay” state.

Table 4-3: Priority Settings for Three Classes

	High	Medium	Low
Gold	90	60	30
Silver	80	50	20
Bronze	70	40	10

The Ponly policy uses only the priorities in the High column of Table 4-3 where as the Sonly policy uses the priority associated with the “pay” state in addition to the priorities in the High column of Table 4-3. The Tonly policy uses all the priorities listed in Table 4-3. In addition to all the priorities listed in Table 4-3, the T&S policy uses the priority associated with the “pay” state.

□ Time Threshold

The Tonly and the T&S policies use Time Thresholds. There are two time thresholds for each class of customers: T1 and T2. If the total time a customer spends in the system exceeds the threshold T1, the first Time Threshold, its priority is lowered to medium. This customer will remain at this priority level until the session length reaches the second threshold T1 + T2 (T2 being the second Time Threshold). When the second threshold has

been reached, the customer's priority is reduced to low level. Table 4-4 shows the default time thresholds in this thesis for the three classes of customers.

Table 4-4: Time Thresholds for Different Classes

	Gold	Silver	Bronze
T1	4000 ms	3000 ms	2000 ms
T2	6000 ms	5000 ms	4000 ms

□ **Angry time (AT)**

A customer will leave a website if performance is poor. Therefore after a customer's End-to-End Time has exceeded a certain value, called angry time, this customer will leave the website. The default angry time in this study is 30000 ms for all customer classes. The choice of this default value is discussed further in Section 5.1. A fixed Angry Time is used in the experiments described in the thesis. Conducting experiments in which Angry Times are proportional to the number of items in the shopping cart seems interesting. For a given scheduling policy, CBMG and arrival rate, E and R are expected to increase with such a dynamically changing AT whereas AF is likely to decrease. However we expect the general conclusions regarding the characteristics of the scheduling policies to remain the same.

4.7 Performance Metrics

The performance metrics used in this thesis are presented in this section.

□ **E: Mean End-to-End Time**

End-to-End Time is the time interval between a customer's arrival and his/her exit from the system. It includes all the Think Times in all the states the customer passes through. E is the mean End-to-End Time.

□ **R: Revenue earned per second**

This is the total amount of money (in dollars) all customers paid to the website, divided by the total experiment time.

□ **D: Discrimination Factor**

This is used to characterize the difference between the End-to-End Times achieved by the customers in different classes. The mean End-to-End Time for the Gold customers is used as the basis for comparison. We use the following three formulas to compute the Discrimination Factors [20].

$$D_{\text{Gold/Gold}} = \frac{\text{Mean End-to-End Time of Gold Customer}}{\text{Mean End-to-End Time of Gold Customer}}$$

$$D_{\text{Gold/Silver}} = \frac{\text{Mean End-to-End Time of Silver Customer}}{\text{Mean End-to-End Time of Gold Customer}}$$

$$D_{\text{Gold/Bronze}} = \frac{\text{Mean End-to-End Time of Bronze Customer}}{\text{Mean End-to-End Time of Gold Customer}}$$

Note that $D_{\text{Gold/Gold}} = 1$.

D is useful in quantifying the degree of differentiated service provided by a system. A high D may be useful in motivating customers to buy membership in a gold or silver class on websites that support such a membership system.

The Discrimination Factor defined in this section is appropriate for comparing the performances of the scheduling policies presented in this thesis. In other situations, using a ratio of End-to-End times each of which is normalized with respect to the no load residence time of that class (discussed in Section 5.1) may be useful.

□ **AF: Angry Factor**

This is the ratio between the number of customers exiting the website due to poor performance, and the total number of customers visiting the site.

□ **ET: Experiment Time**

Experiment time refers to the duration of time between the first customer entering the system and the completion of the last valid customer. This is used for the validation of the experimental results.

□ **S: Average number of states customers visited**

This is the total number of states visited by a given class of customers, divided by the total number of customers in that class. The average number of states customers visited is used in the validation of the experimental results.

4.8 Validations

4.8.1 Applying Little's Law

The validity of the experimental results was performed using Little's Law. As we know, there are two methods for obtaining the average number of jobs in the system. One method is to compute the average number of jobs in the system (N_0) as the rate of the sum of the residence times for all jobs, divided by the experiment time. Another method is to apply Little's Law: $N_1 = AR' * W$, where N_1 is the average number of jobs in the system, AR' is the average measured Arrival Rate, and W is the average residence time of jobs in the system. In a correctly functioning experiment, N_0 and N_1 will be very close to one another. The difference between N_0 and N_1 is obtained using the following formula:

$$DP_i = \frac{N_1 - N_0}{N_0} = \frac{AR'_i * E_i - T_i / X}{T_i / X} * 100\%$$

Where $i = 1$ is for Gold customers, $i = 2$ is for Silver customers, and $i = 3$ is for Bronze customers.

E_i : Average residence time or average End-to-End Time for a class i customer.

T_i : Sum of the End-to-End Times for all class i customers.

X : Experiment time.

AR'_i : Arrival rate for class i customers.

DP_i : Difference for a class i customer.

N_1 : The number of jobs in the system by applying Little's law.

$$DP_{overall} = \frac{N_1 - N_0}{N_0} = \frac{AR'_{overall} * E_{overall} - T_{overall} / X}{T_{overall} / X} * 100\%$$

where,

$E_{overall}$: Average residence time or average End-to-End Time for all customer classes.

$T_{overall}$: Sum of the End-to-End times for all customers.

X : Experiment time.

$AR'_{overall}$: Overall arrival rate for the system.

$DP_{overall}$: Overall difference (including all customer classes).

N_1 : The number of jobs in the system by applying Little's law.

This validation was performed for four sample experiments in which all factors have the default values (see Section 5.1). These experiments led to the results shown in Table 4-5.

Table 4-5: Difference Percentages

Difference Percentage	$AR'_{\text{overall}} = 0.098$ customers/second	$AR'_{\text{overall}} = 0.196$ customers/second	$AR'_{\text{overall}} = 0.303$ customers/second	$AR'_{\text{overall}} = 0.395$ customers/second
DP _{overall}	-0.014%	-0.011%	-0.009%	-0.011%
DP1	-0.011%	-0.012%	-0.005%	-0.010%
DP2	-0.013%	-0.011%	-0.007%	-0.019%
DP3	-0.017%	-0.009%	-0.011%	-0.003%

Note that the Arrival Rate AR'_{overall} and AR'_i are based on the measured Arrival Rate, which is slightly different from the input parameter AR. As shown in the table, the expected results computed by using Little's law are close to the measured values.

4.8.2 Same Navigation Patterns

In most of our experiments, customers from different classes have different navigational patterns in the CBMG. We have used two types of navigational patterns to validate the experimental results. Figure 4-2 demonstrates the End-to-End time achieved with the same navigational patterns for the different classes. Figure 4-3 shows the average number of states customers visited for this situation. In order to show the effects clearly, the same priority (90) and the same think time (1000 ms) are used for all customers. The other factors are set to their default values.

Figure 4-2: Effect of Using the Same Navigational Patterns on End-to-End Time

Figure 4-3: Effect of Using the Same Navigational Patterns on the Average Number of States Customers Visited

As expected, with the same priority and same navigational patterns, Gold, Silver and Bronze customers have very close End-to-End Times and visited a comparable number of states on an average.

4.8.3 Same Priority Settings vs. Different Priority Settings

The comparison between the same priority settings and different priority settings demonstrates that setting different priorities based on the classes of customers can provide differentiated services. If all customers are given the same priority, which means that Gold, Silver and Bronze all have the same priority of 90, they are treated no differently. Also, in order to show the effect of priority settings clearly, the Ponly policy is applied, when all customers have the same navigational patterns and the same think time (1000 ms) while other factors are held at their default values.

Figure 4-4: Performance Achieved with the Same Priority Setting (90-90-90)

As expected, the average End-to-End Times for all the classes shown in Figure 4-4 are close to one another.

Figure 4-5: Performance Achieved with Different Priority Settings

Also, as expected, the average End-to-End Times for different classes of customers are different if the priority setting is different (90 for Gold, 80 for Silver and 70 for Bronze). Gold customers have the shortest End-to-End Time in Figure 4-5. The reason for this is that Gold customers are always given the highest priority, which is “90” in this experiment. From Figure 4-5 we also observe that the mean End-to-End Time of Gold customers remains almost flat as load increases, and the Bronze customers have the lowest priority setting and perform worst. In Figure 4-5 we also find that when different priority settings are used, Gold and Silver customers have a much shorter End-to-End Time at a higher Arrival Rate in comparison to Bronze customers. The mean End-to-End

Time for Silver customers for a given Arrival Rate is in between those achieved by the Gold and Bronze customers.

Thus, the relative performances of Gold, Silver and Bronze customers agree with our expectations.

4.9 Accuracy of Measurement

As described in Section 4.4.2.1, the number of valid threads in each experiment is 600. To decide on the total number of valid threads, we ran a number of experiments with different total number of customers in our system using the FCFS policy and the default CBMG at a high arrival rate of 0.4 customers/second. Initially, as the total number of customers is increased, the mean End-to-End Time is observed to fluctuate. When the total number of customers is 660, the variation in mean End-to-End Time is observed to be very small.

In order to reduce the bias caused by an empty initial state and measure the data generated in a steady state of our system, we have to consider a number of customers that arrived at the beginning to be invalid. In this thesis, we use the method described in [5] to reduce the impact of the initial condition. In this method, the experimental run is divided into two phases: an Initialization Phase and a Data Collection Phase. The Initialization phase corresponds to the invalid threads that are not used in computation of the performance metrics. By using this statistical technique described in [5] for determining the Initialization Phase, the number of invalid threads was computed to be 60 for FCFS at an AR of 0.4 customers/second. Thus, the number of initial invalid threads in our experiments was set to 60. The number of valid threads was set at 600 that was achieved by subtracting 60 from 660.

Each experiment was repeated enough number of times such that all End-to-End Time measurements have a confidence interval of less than $\pm 5\%$ of the mean at a confidence level of 95%.

Chapter 5 Performance of Scheduling Policies

This chapter focuses on the results of our experiments. The experimental design is presented first. The following three sections present discussions on End-to-End Time, Angry Factor and Revenue per Second.

5.1 Experimental Design

In order to demonstrate the impact of each factor on performance, a simple one-factor-at-a-time design is used in this thesis. One of the factors is varied while others are held at their default values. Table 5-1 and Table 5-2 present the levels of parameters used in the experiments. The default parameter values used in the experiments are shown in Table 5-3. The default transition probabilities for the CBMG listed in Table 5-3 were used in [20].

Table 5-1: Levels of Factors

Parameter		Levels		
Arrival Rate (customers/second)		0.1, 0.2, 0.3, 0.4		
		Gold	Silver	Bronze
Think Time (milliseconds)		800, 1000	1000	1000, 1200
Angry Time (milliseconds)		25000, 30000	30000	30000, 35000
Time Thresholds (1000 milliseconds)	T1	3, 4, 4.5, 5	3	3, 2, 1.5, 1
	T2	5, 6, 6.5, 7	5	5, 4, 3.5, 3

Those probabilities not listed in the Table 5-2, such as P_0 , are the same as the default value as given in Table 5-3.

Note that the Think Times and Angry Times used in this thesis are chosen in such a way that the time required to run the experiments is reasonable. For experiments with larger Think Times the Angry Times should be increased proportionately. We believe

that such a change is unlikely to change the general conclusions regarding the performance of the scheduling policies. Investigating more detailed behaviors of the scheduling policies for larger values of Th and AT forms an important direction for future research. Fixed vales of Think Times and Angry Times are used in this thesis. Investigating systems with different distributions of Th and AT seems interesting.

Table 5-2: Transition Probabilities of CBMG

Probabilities	Level		
	Gold	Silver	Bronze
P1	0.9, 0.4	0.8, 0.2	0.7, 0.05
P2	0.8, 0.8	0.6, 0.5	0.4, 0.2
P3	0.1, 0.1	0.3, 0.2	0.5, 0.3
P7	0.1, 0.6	0.2, 0.8	0.3, 0.95
P8	0.1, 0.1	0.1, 0.3	0.1, 0.5

Time Thresholds are tuning parameters that need to be adjusted based on performance measurements made on the system. The default values of T1 (4000 ms) and T2 (6000ms) used for the Gold customers are chosen in such a way that most Gold customers complete their service before their residence times exceed these thresholds. Note that the longest E observed in the experiments described for both CBMG's are below 6000 ms. The default T1 and T2 for Silver customers are selected to be 1000 ms less than the corresponding thresholds for Gold customers. Similarly the default T1 and T2 for Bronze customers are selected to be 1000 ms less than the corresponding thresholds for Silver customers. In this way the system favors the Gold customers the most, followed by the Silver and Bronze customers. In order to investigate the effect of T1 and T2 on performance, we have used a number of different combinations of T1 and T2 values (see Table 5-1). The T1 and T2 for Silver customers are fixed at the default values. The T1 values for Gold and Bronze customers are always selected to be higher and lower than those of Silver respectively. Similarly, the T2 values for Gold and Bronze

customers are also selected to be higher and lower than those of Silver respectively. For all the different combinations, the sum of the T1 values and the sum of the T2 values for the three customer classes are constants.

Table 5-3: The Default Parameter Values

Parameter		Default Value		
Number of Valid Threads		600		
		Gold	Silver	Bronze
Priority Settings	High	90	80	70
	Medium	60	50	40
	Low	30	20	10
Angry Time (milliseconds)		30000	30000	30000
Think Time (milliseconds)		800	1000	1200
Time Thresholds (milliseconds)	T1	4000	3000	2000
	T2	6000	5000	4000
Probabilities of CBMG				
P0		0.9	0.9	0.9
P1		0.9	0.8	0.7
P2		0.8	0.6	0.4
P3		0.1	0.3	0.5
P4		0.1	0.1	0.1
P5		0.8	0.8	0.8
P6		0.1	0.1	0.1
P7		0.1	0.2	0.3
P8		0.1	0.1	0.1
P9		0.1	0.1	0.1

Mean Customer Residence Times and Average Money a Customer Paid at No load:

Two performance metrics achieved on the system in the absence of resource contention when there is only a single customer in the system aptly capture the characteristics of the CBMGs used. Both the mean customer residence time, E' , and the average money a customer paid, R' , at such a “no load” are presented in Table 5-4. E' and R' were measured on the performance prototype when a single customer in any given class is present on the system. As soon as the customer exits, it is replaced by a statistically identical customer. For both the CBMGs, the Gold customers spend the smallest amount of time on the system followed by the Silver and Bronze customers. In spite of the smallest mean E' , for both CBMGs, the R' corresponding to Gold is the highest followed by Silver and Bronze. This is because of the higher probability of a Gold customer visiting the “Pay” state (see Table 5-3). For any given customer class, smaller E' and R' are achieved with the alternate CBMG.

Table 5-4: Excepted Mean Residence Times (ms) and Average Money a Customer Paid (\$/customer)

	Gold		Silver		Bronze	
Performance Metrics	Default CBMG	Alternate CBMG	Default CBMG	Alternate CBMG	Default CBMG	Alternate CBMG
E'	2770	2057	3505	2199	4153	2402
R'	34.67	14.28	31.45	4.56	20.35	0.63

5.2 Effect of Workload and System Parameters on End-to-End Time

5.2.1 Effect of Arrival Rate

Figure 5-1 through Figure 5-3 show the effect of the Arrival Rate on the End-to-End Time of each customer class for a number of different scheduling policies. In these experiments, we use default values for all factors. From Figure 5-1, we observe that a Gold customer's End-to-End Time is always less than 6000 ms, even at higher Arrival Rates. However, in Figure 5-3, Bronze customers have an End-to-End Time five times as long as the Gold customer's at the higher Arrival Rate when the T&S and Tonly policies are applied, and the ratio is even as high as eight when the Ponly and Sonly policies are used.

In addition, the End-to-End Time of Bronze customers is increased all the time, while the Gold customers' End-to-End Time is increased slightly. This clearly shows that different classes of customers are treated differently.

Figure 5-1 through Figure 5-3 also show the effects of different scheduling policies on the End-to-End Time of all customer classes. There are five policies, including Ponly, Sonly, Tonly, T&S (described in Section 3.3 through Section 3.6) and FCFS.

For Bronze customers, the Tonly policy and T&S policy reduce the End-to-End Time by 11% and 18% respectively at the highest Arrival Rate in comparison to the Ponly policy (see Figure 5-3).

Figure 5-1: Effect of Arrival Rate on End-to-End Time of Gold Customers

Figure 5-2: Effect of Arrival Rate on End-to-End Time of Silver Customers

Figure 5-3: Effect of Arrival Rate on End-to-End Time of Bronze Customers

In contrast, these two policies produce higher End-to-End Times for Gold and Silver customers in comparison to those produced by the Ponly and Sonly policies. The reason for this is that these two policies dynamically change the priority of a customer, so that the Bronze customers have more of an opportunity to carry out their jobs. Figure 5-4 and Figure 5-5 show the discrimination factors of Silver and Bronze customers.

Figure 5-4 and Figure 5-5 show that the Tonly and T&S policies reduce the discrimination factor of Bronze customers by 70% and 60% respectively at the higher Arrival Rates, in comparison to the Ponly and Sonly policies.

All four of the policies investigated in our study (except the FCFS Policy) provide different system capacities. All factors in these experiments are held at default values.

Figure 5-4: Discrimination Factor for Silver Customers

Figure 5-5: Discrimination Factor for Bronze Customers

Figure 5-6: Maximum Arrival Rates for Scheduling Policies

Figure 5-6 shows the maximum Arrival Rate each policy can sustain before the average End-to-End Time of Bronze customers exceeds the Angry Time of Bronze customers. Such an arrival rate is a good measure of system capacity, operating the system above this arrival rate can lead to a large number of angry Bronze customers.

Figure 5-6 shows that in the same system, the T&S policy performs best with regard to system capacity. The capacity achieved with T&S policy is twice the capacity that the Ponly policy can provide. Therefore, the T&S policy is a reasonable choice when the workload in the system is high. Note that for the T&S policy, at an Arrival Rate of 0.8 customer/second, the End-to-End Time for Gold customers is still the best in our system, followed by Silver and Bronze customers.

5.2.2 Effect of Think Time

In Figures 5-7 through Figure 5-18, a “D” follows the name of a scheduling policy representing an experiment with a different Think Time setting, whereas an “S” means the experiment has the same Th setting. The same Th is when the Th for all customer classes is set to 1000 ms. The different Th setting is as follows: Gold customers-800 ms, Silver customers-1000ms, Bronze customers-1200 ms (see Section 5.1). The other factors are set to the default values.

Figure 5-7 through Figure 5-10 present the effect of Think Time on the End-to-End Time (E) for Gold customers. From these figures we can see that for Gold customers, at all Arrival Rates, the E for a given scheduling policy achieved with the same Think Time setting is greater than E achieved with a different Think Time setting. Also, we see that, compared to the Tonly and the T&S policies, the Sonly and the Ponly policies resulted in a better E for Gold customers when the Arrival Rate is high.

Figure 5-11 through Figure 5-14 show the effect of Think Time on E for Silver customers. For Silver customers, at lower Arrival Rates, the End-to-End Times with different Think Time settings are very close to those with the same Think Time settings. However, when the Arrival Rate, AR, is higher, the End-to-End Times with the same Think Time settings exceed those with different Think Time settings for all four scheduling policies.

Figure 5-7: Effect of Think Time on End-to-End Time of Gold Customers with Ponly Policy

Figure 5-8: Effect of Think Time on End-to-End Time of Gold Customers with Sonly Policy

Figure 5-9: Effect of Think Time on End-to-End Time of Gold Customers with Tonly Policy

Figure 5-10: Effect of Think Time on End-to-End Time of Gold Customers with T&S Policy

Figure 5-11: Effect of Think Time on End-to-End Time of Silver Customers with Ponly Policy

Figure 5-15 through Figure 5-18 show the effect of Think Time on the End-to-End Time (E) for Bronze customers. For Bronze customers, the same Think Time setting has a slight advantage over the different Think Time settings. However, at the highest AR, the same Think Time setting leads to a higher E than the different Think Time setting.

Figure 5-12: Effect of Think Time on End-to-End Time of Silver Customers with Sonly Policy

Figure 5-13: Effect of Think Time on End-to-End Time of Silver Customers with Tonly Policy

Figure 5-14: Effect of Think Time on End-to-End Time of Silver Customers with T&S Policy

Figure 5-15: Effect of Think Time on End-to-End Time of Bronze Customers with Ponly Policy

Figure 5-16: Effect of Think Time on End-to-End Time of Bronze Customers with Sonly Policy

Figure 5-17: Effect of Think Time on End-to-End Time of Bronze Customers with Tonly Policy

Figure 5-18: Effect of Think Time on End-to-End Time of Bronze Customers with T&S Policy

Figure 5-7 through Figure 5-18 show that with different Think Times for the three classes, a lower E is often achieved in comparison to a system with the same Think Time for all classes under our four scheduling policies, especially at higher Arrival Rates.

Figure 5-19 presents the effect of Think Time on E for all classes at the high Arrival Rate of 0.4 customers/second. The columns with diagonal markings represent experiments using the same Think Time setting for different customer classes (for example Ponly-S), and those without diagonal markings represent experiments using different Th setting for all classes, for example Ponly-D. The E's achieved with the different scheduling policies appear in the same sequence as the legends in the figure.

Figure 5-19: Effect of Think Time on End-to-End Time for All Customer Classes at High Arrival Rate

Figure 5-19 shows the effect of the different policies on E for all customer classes at a higher AR. For Gold and Silver customers, the Ponly and Sonly policies produced better E than the Tonly and T&S policies with both same and different Th settings. For Bronze customers, however, the result is just the opposite. The T&S policy gives the best E followed by Tonly, Sonly and Ponly with both the same and different Th settings. The reason for this is the same as mentioned earlier: the T&S and Tonly policies dynamically change the priority of a customer, so that the Bronze customers have more of an opportunity to carry out their jobs.

5.2.3 Effect of Angry Time

In Figures 5-20 through Figure 5-31, a “D” following the name of a scheduling policy represents an experiment with a different Angry Time, AT, setting, whereas an “S” means that an experiment has the same AT setting. The same AT setting means that the AT for all customer classes is set to 30000 ms. The different AT setting is as follows: Gold customers-25000 ms, Silver customers-30000 ms, Bronze customers-35000 ms (see Section 5.1). The other factors are set to the default values.

Figure 5-20: Effect of Angry Time on End-to-End Time of Gold Customers with Ponly Policy

Figure 5-20 through Figure 5-23 demonstrate the effect of Angry Time on the End-to-End Time for Gold customers. In these four figures we observed that for the AT values used, the AT settings have almost no effect on the E of Gold customers except for the Ponly policy.

Figure 5-21: Effect of Angry Time on End-to-End Time of Gold Customers with Sonly Policy

Figure 5-22: Effect of Angry Time on End-to-End Time of Gold Customers with Tonly Policy

Figure 5-23: Effect of Angry Time on End-to-End Time of Gold Customers with T&S Policy

Under the Ponly policy, the system with different AT's for the three classes produces a lower E in comparison to a system with the same AT for the three classes at low and medium Arrival Rates. At a high Arrival Rate of 0.4 customers/second, a slightly better E for Gold customers is achieved on the system with the same AT setting in comparison to the system with the different AT settings. .

Figure 5-24 through 5-27 show the effect of Angry Time on E for Silver customers. In these figures we observed that for Silver customers, there is no big difference between the same AT setting and the different AT setting at lower Arrival Rates.

Figure 5-24: Effect of Angry Time on End-to-End Time of Silver Customers with Ponly Policy

However, at a higher AR, the End-to-End Times based on the different Angry Time setting slightly exceed the End-to-End Time based on the same Angry Time setting except for the Ponly policy. Under Ponly policy, at a higher AR, the different AT settings produce a shorter E for Silver customers in comparison to the same AT setting (see Figure 5-24).

Figure 5-28 through Figure 5-31 show the effect of Angry Time on the End-to-End Time for Bronze Customers. For Bronze customers, the End-to-End Times achieved at a given Arrival Rate and a scheduling policy of different angry time settings are longer than those of the same angry time settings.

Figure 5-25: Effect of Angry Time on End-to-End Time of Silver Customers with Sonly Policy

Figure 5-26: Effect of Angry Time on End-to-End Time of Silver Customers with Tonly Policy

Figure 5-27: Effect of Angry Time on End-to-End Time of Silver Customers with T&S Policy

Figure 5-28: Effect of Angry Time on End-to-End Time of Bronze Customers with Ponly Policy

Figure 5-29: Effect of Angry Time on End-to-End Time of Bronze Customers with Sonly Policy

Figure 5-30: Effect of Angry Time on End-to-End Time of Bronze Customers with Tonly Policy

Figure 5-31: Effect of Angry Time on End-to-End Time of Bronze Customers with T&S Policy

Therefore, the same Angry Time setting is better than the different setting in reducing E. The reason for this is that reducing the Gold Customer's Angry Time in different AT settings has no significant effect on the End-to-End Time of Gold customers. However, increasing the AT of Bronze customers in different AT settings will result in more Bronze customers remaining in the system, and the waiting time for Bronze customers in the ready queue increases and the End-to-End Time of Bronze customers will increase as well.

Figure 5-32 shows the effect of AT on E for all classes at the high Arrival Rate of 0.4 customers/second. The columns without diagonal markings represent experiments using the same AT setting for different customer classes (for example Ponly-S), and those with

diagonal markings represent experiments using different AT setting for all classes, for example Ponly-D.

Figure 5-32: Effect of Angry Time on End-to-End Time for All Customer Classes at High Arrival Rate

Figure 5-32 shows the effect of different policies on E of all customer classes at a high Arrival Rate of 0.4 customers/second. For Gold and Silver customers, the Ponly and Sonly policies produce better E than the Tonly and T&S policies with both the same and different AT settings. For Bronze customers, the effects of the policies are different for a given AT setting. The T&S policy provides the best E followed by Tonly, Sonly, and Ponly with the same AT setting. For the different AT settings, the T&S policy is still the best policy followed by Tonly policy, however, the Ponly policy performs better than Sonly policy. The reason for this is as mentioned earlier, the T&S and Tonly policies dynamically change the priority of a customer, so that the Bronze customers have more of

an opportunity to carry out their jobs. Note that for Gold customers, the E for any given policy observed on a system with the same AT setting is comparable to that observed on the system with different AT settings.

5.2.4 Effect of Time Threshold

There are two scheduling policies related to Time Threshold: T&S and Tonly. This section contains a discussion of the effect of Time Threshold on the End-to-End Time of a customer class for the T&S and Tonly policies. In these figures, “same” signifies that all customer classes have the same T1 and T2 (3000 ms and 5000 ms). The settings for Time Threshold in our experiments are listed in Table 5-4, where “X-Y” means that the T1 of a customer class is X ms and the T2 of this customer class is Y ms. Other factors in our experiments are held at default values.

Table 5-5: Time Threshold Settings for Three Customer Classes (ms)

Time Threshold Settings	Gold	Silver	Bronze
Same	3000-5000	3000-5000	3000-5000
Setting1	5000-7000	3000-5000	1000-3000
Setting2	4500-6500	3000-5000	1500-3500
Setting3 (default)	4000-6000	3000-5000	2000-4000

Figures 5-33 through Figure 5-35 show that with regard to the End-to-End Time, the different Time Thresholds for different customer classes produce a better performance than the same Time Threshold setting. For Gold customers, the End-to-End Time with the same Time Threshold setting is greater than the End-to-End Time for all the three different Time Threshold settings at higher Arrival Rates.

Figure 5-33: Effect of Time Threshold on End-to-End Time of Gold Customers with T&S Policy

Figure 5-34: Effect of Time Threshold on End-to-End Time of Silver Customers with T&S Policy

Figure 5-35: Effect of Time Threshold on End-to-End Time of Bronze Customers with T&S Policy

For Silver customers, the End-to-End Time follows the same trend as the End-to-End of Gold customers. The difference is that the E for the same Time Threshold setting is closer to the E's for different Time Threshold settings. For Bronze customers, all four Time Threshold settings lead to comparable End-to-End Times except at an Arrival Rate of 0.4 customers/second Setting2 produces the best performance.

Therefore, with regard to the T&S policy, setting a different Time Threshold for different customer classes seems to be favorable to setting the same Time Threshold for different customer classes.

Figures 5-36 through Figure 5-38 show the effect of Time Threshold on the End-to-End Time of all customer classes using the Tonly scheduling policy.

Figure 5-36: Effect of Time Threshold on End-to-End Time of Gold Customers with Tonly Policy

Figure 5-37: Effect of Time Threshold on End-to-End Time of Silver Customers with Tonly Policy

Figure 5-38: Effect of Time Threshold on End-to-End Time of Bronze Customers with Tonly Policy

For Gold customers, the same Time Threshold settings give the longest End-to-End Time at all AR levels. For Silver and Bronze customers, the End-to-End Times with different Time Threshold settings and same Time Threshold settings are close to one another. For Silver customers, at a higher AR of 0.4 customers/second, Setting1 produces a slightly better E than other three settings. For Bronze customers, the same Time Threshold setting has the worst performance. Since the same Time Threshold settings lead to worse performance for all the customer classes, using a different Time Threshold for different customer classes is preferable.

5.2.5 Effect of CBMG

In this section, we compare the performances achieved using two different CBMGs. The difference between these two CBMGs lies in their transition probabilities. Table 5-5 shows the transition probabilities of the default CBMG and Table 5-6 shows the transition probabilities of the alternate CBMG.

Table 5-6: Transition Probabilities of Default CBMG

Probabilities	Level		
	Gold	Silver	Bronze
P0	0.9	0.9	0.9
P1	0.9	0.8	0.7
P2	0.8	0.6	0.4
P3	0.1	0.3	0.5
P4	0.1	0.1	0.1
P5	0.8	0.8	0.8
P6	0.1	0.1	0.1
P7	0.1	0.2	0.3
P8	0.1	0.1	0.1
P9	0.1	0.1	0.1

Table 5-7: Transition Probabilities of Alternate CBMG

Probabilities	Level		
	Gold	Silver	Bronze
P0	0.9	0.9	0.9
P1	0.4	0.2	0.05
P2	0.8	0.5	0.2
P3	0.1	0.2	0.3
P4	0.1	0.1	0.1
P5	0.8	0.8	0.8
P6	0.1	0.1	0.1
P7	0.6	0.8	0.95
P8	0.1	0.3	0.5
P9	0.1	0.1	0.1

The default CBMG represents the behavior of customers who visit an attractive online store, whereas the alternate CBMG represents a less attractive store. The transition

probabilities for a customer to visit the “confirm” and “pay” states in the default CBMG are larger than those in the alternate CBMG. In addition, the probabilities that a customer will exit this online store in the default CBMG are lower than those in the alternate CBMG.

Figures 5-39 through Figure 5-41 present the effects of different CBMGs on the End-to-End Time of all customer classes, where the other factors are set to the default values.

Figure 5-39: Effect of CBMGs on End-to-End Time for Gold Customers

The dashed lines in the three figures present the alternate CBMG. The solid lines represent the default CBMG.

From the above three figures, it is evident that although the two CBMGs have different transition probabilities for each customer class, the End-to-End Times of all customer classes follow the same patterns.

Figure 5-40: Effect of CBMGs on End-to-End Time for Silver Customers

Figure 5-41: Effect of CBMGs on End-to-End Time for Bronze Customers

For a given scheduling policy and Arrival Rate, the default CBMG leads to a higher End-to-End time for any given customer class. However, the CBMG itself makes no difference with regard to the rank of the different scheduling policies in terms of End-to-End Times at higher AR rates.

5.3 Effect of Workload and System Parameters on Angry Factor

5.3.1 Effect of Arrival Rate

Figure 5-42 shows the effect of the Arrival Rate on AF, the Angry Factor (defined in Section 4.7). AF is an average computed factor across all customer classes. From the figure, we can see that the higher the Arrival Rate, the more customers are angry. Here in our experiments, all other factors are held at default values.

Figure 5-42: Effect of Arrival Rate on the Angry Factor

As shown in Figure 5-42, the relative performances of our four policies are similar in terms of AF as in terms of the End-to-End Times. The Ponly and Sonly policies perform better than the Tonly and T&S policies, especially at higher Arrival Rates. At lower Arrival Rates, Angry Factor achieved with policies seems to be closer to one another. The reason for this result is that when the Ponly and Sonly policies are used, the Gold and Silver customers have higher priorities than the Bronze customers do. For this reason, they barely get angry. The Bronze customers are more likely to get angry. However, under the Tonly and T&S policies, Bronze customers have more of an opportunity to carry out their jobs, but there is more chance that the Silver customers will have to wait in the ready queue, and therefore more Silver customers will be angry, in addition to the angry Bronze customers, especially at higher Arrival Rates. Since the Gold customers have the highest priority in the system, they are not angry at all levels of AR. Therefore, the Ponly and Sonly policies perform better than the Tonly and T&S policies when the load of the system is higher.

5.3.2 Effect of Think Time

The effect of using two types of Th, Think Time settings is investigated. With the same Think Time setting, all customer classes have the same Think Time of 1000 ms, and the different Think Time setting means that Gold customers have 800ms, Silver customers 1000ms and Bronze customers 1200 ms of Think Time. The Angry Factors achieved by the different policies are presented in Figure 5-43. The solid lines represent experiments with the different Think Time setting and the dashed lines show experiments with the same Think Time setting.

Figure 5-43: Effect of Think Time on the Angry Factor

Figure 5-43 shows that for a given policy, a lower E is observed on a system with the different Think Time setting in comparison to the system with the same Think Time setting. The relative performance of the scheduling policies on the system with the same Think Time setting follows a trend similar to that achieved on the system with the different Think Time setting. The Ponly and Sonly policies perform better than the Tonly and T&S policies.

The reason for this is that when all customer classes have the same Th settings, Gold customers have a longer Th in comparison to the different Th settings. Thus they stay longer in each state in the CBMG. Therefore, more Silver and Bronze customers are put into the ready queue and have to wait longer. Thus, the different Th setting gives a better result than the same Th setting.

5.3.3 Effect of Angry Time

The effect of using two different Angry Time, AT, settings on the AF achieved by the different scheduling policies is shown in Figure 5-44. A “D” following the name of a scheduling policy represents an experiment with a different AT setting, whereas an “S” means that an experiment has the same AT setting. The dashed lines represent experiments with the same AT setting for all classes, and the solid lines represent experiments with the different AT settings. With the same AT setting, all customer classes have the same AT of 30000 ms, whereas in the different AT setting, Gold customers have an AT of 25000ms, Silver customers have an AT of 30000ms and Bronze customers have an AT of 35000 ms. (See Section 5.1)

Figure 5-44: Effect of Angry Time on the Angry Factor

Figure 5-44 shows that the different AT setting results in a higher AF for a given policy and AR. However, the performances achieved with one scheduling policy and the two AT settings are very close to one another. Even at the highest AR, the difference is no more than 6%. Also, the relative performance of the scheduling policies is the same for both AT settings. The Tonly and T&S policies are slightly worse than the Ponly and Sonly policies.

5.3.4 Effect of Time Threshold

Figures 5-45 and 5-46 show the effect of Time Threshold on the Angry Factor achieved by the Tonly and T&S policies. In these figures, “same” signifies that all customer classes have the same T1 and T2 (3000 ms and 5000 ms). The settings for Time Threshold in our experiments are listed in Table 5-4, where “X-Y” means that the T1 of a customer class is X ms and the T2 of this customer class is Y ms. Other factors in our experiments are held at default values.

It is evident that under both policies, at lower AR the performances of the same setting and different setting are very close. At higher AR, under the Tonly policy, the same Time Threshold setting produces a slightly worse performance than the different setting.

Under the T&S policy, the difference between the same setting and the different setting is as large as 17% at an Arrival Rate of 0.4 customers/second. Therefore, the same Time Threshold setting seems to be worse than the different Time Threshold setting for both policies, especially at high Arrival Rates.

Figure 5-45: Effect of Time Threshold on the Angry Factor with Tonly Policy

Figure 5-46: Effect of Time Threshold on the Angry Factor with T&S Policy

With the different Time Threshold setting, it takes longer for a Gold customer to drop to a lower priority (T1 for the same Time Threshold setting is the shortest: 3000ms). Thus Gold customers are most likely to finish their jobs before they need to go to the ready queue. Thus Gold customers exit the system quicker. For this reason, Silver and Bronze customers wait for a shorter period of time in the ready queue, and fewer of them get angry. Therefore, the different Time Threshold setting gives a better result than the same Time Threshold setting does. The best performance in the experiment is achieved with Setting3, the default Time Threshold setting (see Figure 5-45 and Figure 5-46).

5.3.5 Effect of CBMG

Figure 5-47 shows the Angry Factors achieved by the different scheduling policies when different CBMGs are used. The solid lines represent the default CBMG. The dashed lines in the figure represent the alternate CBMG (see Section 5.2.5). In these experiments, the other factors are held at the default values.

In Figure 5-47 we can see that although the two CBMGs have different transition probabilities for each customer class, the Angry Factor follows the same pattern. The Sonly policy performs the best, followed by the Ponly, T&S and Tonly policies.

Again, the CBMG itself has no effect on the relative performance of the different scheduling policies with regard to the Angry Factor. However, the alternate CBMG leads to a lower Angry Factor for a given scheduling policy and arrival rate. The possible reason is that customers exit this store more on their own, and fewer customers are angry.

Figure 5-47: Effect of Different CBMGs on the Angry Factor

5.4 Effect of Workload and System Parameters on Revenue

5.4.1 Effect of Arrival Rate

Figure 5-48 shows the effect of the Arrival Rate on R , the revenue earned per second by our online store. A description of the computation of R is presented in Section 4.7. From this figure we observe that as the Arrival Rate increases, the revenue generated per second increases. However, the slopes of the R lines are not constant. At a higher Arrival Rate, the rate of increase in the R earned by our system for the four proposed policies decreases slightly. When the Tonly policy is used, the line even becomes almost flat. As the Arrival Rates increase from low to medium, more and more customers arrive and increase the R for the shop. However, as the Arrival Rate increases further, the proportion

of angry customers in the system tends to increase. Since the angry customers do not contribute to R , the slopes of the graphs are reduced at higher Arrival Rates. It is interesting to observe that as AR is increased from 0.3 customers/second to 0.4 customers/second, R achieved by FCFS drops significantly. Gold customers do not receive a higher priority with FCFS. Thus, compared to the other strategies, a larger number of these customers leave the system without completing the purchase at this high arrival rate. Since Gold customers are heavy buyers, this leads to a drop in the revenue earned per second.

Figure 5-48: Effect of Arrival Rate on Revenue

In addition, when the Arrival Rate is lower, the R generated by the four policies are comparable. However, we found that the T&S policy gives the best result at the highest Arrival Rate. The Sonly policy is the second best policy in terms of earning R for the shop. The poorest policy in terms of R is the Tonly policy.

Both the T&S and Sonly policies take the state of a customer into account; a customer in the “pay” state has the highest priority in the shop, whereas the Tonly and Ponly policies do not differentiate between a customer making a purchase from the shop and a customer who is just browsing the shop.

Under the Tonly and Ponly policies, more customers (especially Silver and Bronze customers) who are about to make a purchase have to wait in the ready queue, and after a certain time they will be angry and exit the shop, and the shop loses R from such Silver and Bronze customers. Thus the T&S and Sonly policies produce more R than the Ponly and Tonly policies do.

5.4.2 Effect of Think Time

Figure 5-49 shows the effect of Think Time on R earned per second. In this figure, the dashed lines represent experiments with the same Think Time setting for all classes, and the solid lines represent experiments with the different Think Time setting. Other factors in these experiments are held in default values.

Figure 5-49 demonstrates that at lower Arrival Rates, the performances of all the policies are very close. At higher Arrival Rates, the T&S policy provides the largest revenue per second, using both the different and same settings, while the different setting has a slightly better result. Because the T1 for Gold customers is set longer with the different setting, Gold customers finish their jobs quicker. Thus Silver and Bronze customers have more of an opportunity to complete their jobs before they get angry, and have a better opportunity to complete their purchases. The Tonly policy performs the poorest in both cases.

Figure 5-49: Effect of Think Time on Revenue

Given a scheduling policy, the different Think Time setting and the same Think Time setting make no major difference with regard to revenue earned per second by our online store.

5.4.3 Effect of Angry Time

Figure 5-50 shows the effect of Angry Time, AT, on revenue earned per second. The dashed lines correspond to the experiment using the different AT setting, and the solid lines correspond to the experiments using the same AT setting.

Figure 5-50 shows that at lower AR, the R earned by the different scheduling policies using the different AT setting are very close. At higher AR values, the T&S policy leads to more R than the others for both AT settings. The Tonly policy results in the lowest R in both cases. With the different AT setting, the relative performance of the four policies

remains the same as when the same AT setting is used. Moreover, for any given policy and Arrival Rate, the performances achieved under the two angry time settings are comparable.

Figure 5-50: Effect of Angry Time on Revenue per Second

Therefore, the Angry Time settings used in our experiments do not seem to have a major effect on the R earned by our online store.

5.4.4 Effect of Time Threshold

Figures 5-51 and 5-52 demonstrate the effect of the different Time Threshold settings (described in Table 5-4) on the R earned per second under the T&S and Tonly policies. As these figures show, at the lower Arrival Rates, the amount of Revenue earned per second (R) using the different Time Threshold setting is very close to the R earned with

the same Time Threshold setting in both the Tonly and T&S policies. At a higher Arrival Rate, the same Time Threshold setting in both policies results in poorer R.

Figure 5-51: Effect of Time Threshold on Revenue per Second with T&S Policy

Under the T&S policy, the R gained by using the same setting is less than the different settings by at least 6%. With Setting3, the default Time Threshold setting, the amount of R earned is greater than the same setting by 26%. Under the Tonly policy, the difference of R between Setting2 and the other settings is at least 5%.

Therefore, the different Time Threshold setting has an advantage over the same Time Threshold setting.

Figure 5-52: Effect of Time Threshold on Revenue per Second with Tonly Policy

5.4.5 Effect of CBMG

Figure 5-53 shows the effect of using different CBMGs on revenue earned per second where other factors are held at default values. From this figure we observe that although the dashed lines show lower R at all Arrival Rates, the relative performances of the four policies remain the same with different CBMGs. The T&S policy still provides the best R results, followed by the Sonly, Tonly and the Ponly policies.

Figure 5-53: Effect of CBMG on Revenue per Second

Therefore, different customer behavior, as reflected by the different CBMGs, may change a website's R, but they do not seem to affect the relative performances of the different scheduling policies. The alternative CBMG represents a less attractive shop than the one represented by the default CBMG. Customers are more likely to leave when the alternative CBMG is used, and the shop is more likely to lose R. Therefore, the alternative CBMG produces less R than the default CBMG does.

5.5 Discussion

In this chapter, we present our experimental results to show the effects of different workload and system parameters on the End-to-End Time, the Angry Factor and the R of our online store. The results are summarized in this section.

- **Arrival Rate:** As the Arrival Rate increases, the End-to-End Time for all customer classes increases, whereas the E of Gold customers increases only very slightly. The Angry Factor increases sharply, especially at higher Arrival Rates. For the Angry Factor, the Ponly and Sonly policies produce better results than the Tonly and T&S do at higher Arrival Rates. For the E, the Ponly and Sonly policies perform worse than Tonly and T&S policies at higher Arrival Rate for Bronze customers, but better for Gold and Silver customers. The R generated by our online store increases; however, at higher Arrival Rates, the rate of Revenue increase is decreased. In addition, with regard to R, the T&S policy has the best result, followed by the other policies at high Arrival Rates.
- **Think Time:** Systems with different Think Time settings lead to a better performance than systems with the same Think Time setting with regard to the End-to-End Time for each customer class, as well as the Angry Factor. Neither the different setting nor the same setting has a major effect on the relative performance of the four policies. With regard to both End-to-End Time and the Angry Factor, the T&S policy and the Tonly policy performed worse than the Ponly and Sonly policies at higher Arrival Rates for Gold and Silver customers, but better for Bronze customers. With regard to the R earned, there is no major difference between systems with the same Think Time setting and systems with the different Think Time settings, and under both settings, the relative performance of the different scheduling policies are similar: the T&S policy performed the best, with the other policies following.

- **Angry Time:** Systems with the same Angry Time setting produce a comparable End-to-End Time with a system with the different Angry Time settings for Bronze customers under all four scheduling policies. For Gold customers and Silver customers, the End-to-End Times achieved on a system with the same AT setting and a system with the different AT settings are close to one another under Sonly, Tonly and T&S policies. The only exception is the Ponly policy. With the Ponly policy, a system with the different AT setting produces a shorter E for Gold and Silver customers at higher AR. Whether a system with the same AT setting or a the different AT settings is used, there is no difference in the ranks of the scheduling policies with regard to the Angry Factor and the R earned by the online store. In terms of R, the T&S policy gives the best R of all the four policies; however, in terms of the E and the Angry Factor, the Ponly and the Sonly polices perform better than the other two at higher Arrival Rates for Gold and Silver customers but poorer for Bronze customers.
- **Time Threshold:** For both the Tonly and T&S policies, the different Time Threshold setting is better than the same Time Threshold setting in terms of End-to-End Time, Angry Factor and the R brought in to the website at higher AR for Gold and Silver customers. For Bronze customers, the E produced by both same setting and different setting are close to each other.
- **CBMG:** Different CBMGs may lead to different End-to-End Times and the Angry Factor for a given scheduling policy. However, for the two CBMGs used, no significant difference is observed regarding the relative performances of the four scheduling policies.

Note that a policy that performs best with regard to the system performance is not necessarily the best one for business oriented performance metrics. Here, the T&S policy does not produce the shortest End-to-End Time and the smallest Angry Factor, but it does achieve the greatest R.

Chapter 6 Conclusions

This chapter summarizes this thesis and provides the conclusions drawn from the experimental results. The final section will contain a discussion of the directions for future work.

6.1 Summary

In this thesis, we proposed four different scheduling policies in order to improve the revenue generated by an online store, support its business goal and maintain a high system performance. The four policies are:

- ❑ Policy based on a customer's class only.
- ❑ Policy based on a customer's class and the shopping state of the customer.
- ❑ Policy based on a customer's class and the total time a customer spends in the online store.
- ❑ Policy based on a customer's class, the shopping state of this customer and total time spent by this customer in the online store.

The first policy is a static policy since it does not change a customer's priority once the customer has been assigned to a specific class (Gold, Silver, Bronze). The other three policies change a customer's priority dynamically according to not only a customer's class, but also the customer's state on the website.

We used a "simulated" online bookstore and developed a performance prototype to evaluate the performance of the four scheduling policies. Our performance prototype is based on the popular open source Apache Web server, running under Red Hat Linux 6.2,

kernel 2.2.16-3 on 266 MHz Pentium II PCs with 64MB of RAM. The PCs are interconnected by a 100 MB Ethernet LAN. We have used the Apache Web server and a synthetic open model workload generation tool to analyze the effect of different scheduling policies on e-commerce system performance. The reason for choosing this prototype and measurement-based approach is that they can capture the effect of system overheads, which is difficult to do accurately with simulation or analytic models.

A workload generation tool implemented in Java was developed in this thesis. This tool is used to simulate the workload produced by a customer's browser. It uses the HTTP 1.1 GET method to request a page from our virtual bookstore, and measures the End-to-End time, Angry Factor and Revenue Earned per Second for the online store. This workload generation tool is implemented based on an open model, since the open model bears more similarity to the real Web requests made to an e-commerce website than a closed model. In this tool, one thread represents one customer. In order to implement our dynamic priority policies, we developed our own thread scheduler. This scheduler can put a thread into a waiting state or wake it up, based on the priority of that thread. Also, our scheduler can change the priority of a thread according to its real-time state.

Our experiments are based on a Customer Behavior Model Graphic (CBMG). The End-to-End time measurements have a confidence interval of less than $\pm 5\%$ of the average End-to-End time at a confidence level of 95%.

A number of experiments were performed in order to compare the different scheduling policies. The experiments demonstrate the effects of different workload and system parameters on the performance of the scheduling policies. These parameters include the Arrival Rate, Think Time, Angry Time, Time Threshold, and the CBMGs

used for modeling customer behavior. This allowed us to obtain insights into the behavior and performance of our system.

The next section summarizes the insights derived from the results of the experiments presented in Chapter 6 of the thesis.

6.2 Conclusions

The insights into system behavior presented in this section are based on the experiments described in this thesis.

The experimental results shown in Chapter 6 demonstrate that the dynamic priority scheduling policies we proposed have better results than the static priority policy. Table 6-1 shows the performance of the four policies as compared with a system that uses a First-Come First-Served policy for scheduling at an Arrival Rate of 0.4 customer/second. The other parameters are set to default values (see Section 5.1). A positive number represents an increase in the corresponding performance metric and a negative number a decrease.

Table 6-1: Effect of Proposed Policies Compared to FCFS Policy

	T&S	Sonly	Ponly	Tonly
Revenue/Second	+121.5%	+105.6%	+93.3%	+71.6%
Angry Factor	-21.4%	-34.9%	-30.2%	-11.7%
End-to-End time (Gold)	-84.4%	-88.1%	-87.9%	-81.1%
End-to-End time (Silver)	-33.4%	-50.7%	-49.5%	-30.6%
End-to-End time (Bronze)	-7.1%	+10.9%	+13.4%	+1.2%

Table 6-1 shows that in most cases there is a gain in the business oriented performance metrics of an e-commerce site, as well as in the conventional system

performance metrics, when one of our dynamic priority-changing scheduling policies is used. There is an increase in the End-to-End Time for Bronze Customers, while using the Sonly, Tonly and Ponly policies. The T&S policy produces a performance improvement for Bronze customers. This is because with T&S policy, the priority of a Bronze customer can increase in two situations: when this customer is in the “pay” state and when this customer has put an item in the shopping cart.

A comparison between a system that does not distinguish between different customer classes and a system that classifies customers in three categories can be made from the data presented in Table 6-1 and Figure 5-48. FCFS corresponds to a system in which all customers are treated uniformly irrespective of the class they belong to. The results presented in Table 6-1 and Figure 5-48 show that by using appropriate scheduling policies that distinguish among customer classes a substantial increase in revenue can be achieved at higher loads.

Usually, Gold customers are the heaviest buyers, Silver customers are next and the Bronzer customers are the least likely to buy from an e-commerce site. If an e-commerce site’s marketing strategy is to make a large differentiation between different customer classes and increase membership sales, choosing the Ponly policy or the Sonly policy may have better results with regard to maximizing the discrimination between classes. In this case, Gold customers get faster service than the other two classes. For example, consider an e-commerce website that provides an information exchange service in a specific industry. If a customer (such as a company) buys a VIP membership, no matter how many transactions or purchases it makes from this website, this customer always gets the best service. However, if increasing the revenue generated per second from

product sale is the objective, the T&S policy would be the best choice. The T&S policy produces a reasonable differentiated service among the different customer classes while producing the highest revenue. This in turn can motivate Bronze and Silver Customers to purchase a Gold membership. Note that the T&S policy is more complex than other strategies such as the Ponly policy. This may lead to a higher cost of implementation and runtime overhead.

In addition, e-commerce websites can choose their scheduling policy according to their system load. If a website is experiencing a low to moderate Arrival Rate, it might choose the Sonly policy, since its revenue earned is very close to that earned by the T&S policy: the difference between the revenues generated by these two policies is less than 2%. In addition, compared to the T&S policy, it provides a better End-to-End time for Silver and Bronze customers, without significantly hurting the End-to-End time of Gold customers. The End-to-End time for Bronze customers is decreased by 10% and for Silver customers by 18%. The End-to-End time for Gold customers is increased by only 0.1%.

As the Arrival Rate increases, the website can switch from the Sonly to the T&S policy; this will keep the website's revenue growing and provide greater system capacity for handling a heavier workload. The T&S policy can handle the heaviest system load (twice that which the First-Come First-Served policy can handle) among our four priority scheduling policies (see Section 5.2.1).

Although in this thesis the experiments are only based on three levels (Gold, Silver and Bronze), note that more than three levels can easily be implemented by extending the idea of these three levels.

The next section suggests possible directions for future work.

6.3 Future Work

The following directions are worthy of consideration for future work:

- ❑ Adding admission control to an e-commerce website. In our study we did not add an admission control algorithm and customers kept coming into our online store. Even with the most sophisticated scheduling policy, the T&S policy, when the Arrival Rate becomes very high, the system cannot handle such a heavy workload. In order to avoid this situation, adding admission control to prevent too many low-priority customers entering the system at high load might lead to a better performance.
- ❑ Nowadays, many ISPs provide hosting services for multiple companies. A single server can host multiple online stores. In all of our experiments, we had only one virtual bookstore. Also, customers may be categorized based on which store they are visiting in addition to their purchase history. If a given store has paid a high service fee to the ISP, customers who request web pages from this store will have the highest priorities in the server. How to manage the relative priorities of these stores and how to allocate system resources between these stores need to be investigated in the future.
- ❑ The past history of existing customers can be used to classify the customers into classes. With regard to brand-new customers, for whom we don't have any information, the strategy for assigning that new customer to a specific class is important for a future study. Moreover, the behavior of a particular customer may

change over time. A system that dynamically manages customer classes is worthy of future research.

- Although the general conclusions derived for the performance characteristics of the scheduling policies are expected to be valid for a range of workload and system parameters, experimentation with other parameter values and the investigation of the policies under real workload are required. For example, sensitivity analysis studies can also lead to detailed insights regarding the effect of different parameters on the performances of the scheduling policies.

References

- [1] S. Abraham, P. B. Galvin, and G. Gagne. *Operating System Concepts*. 6th Ed. John Wiley & Sons, 2003.
- [2] J. Almeida, M. Dabu, A. Manikutty and P. Cao, "Providing Differentiated Quality-of-Service in Web Hosting Services," in *the proceedings of 1998 SIGMETRICS Workshop on Internet Server Performance*, Madison, WI, Mar 1998, pp. 1-10.
- [3] M. Aron, P. Druschel and W. Zwaenepoel, "Cluster Reserves: A Mechanism for Resource Management in Cluster-based Network Servers," in *the proceedings of 2000 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, Santa Clara, CA, June 2000, pp. 90-101.
- [4] G. Banga and P. Druschel, "Measuring the Capacity of a Web Server under Realistic Loads", *World Wide Web Journal, Special Issue on World Wide Web Characterization and Performance Evaluation*, Vol. 2, No. 1, May 1999, pp. 69-83.
- [5] J. Banks, J. S. Carson II, B. L. Nelson and D. M. Nicol, *Discrete-Event System Simulation*, Prentice-Hall, Inc. 1999.
- [6] N. Bansal and M. Harchol-Balter, "Analysis of SRPT Scheduling: Investigating Unfairness," in *the proceedings of ACM SIGMETRICS Conference*, Cambridge, MA, June 2001, pp. 279-290.
- [7] P. Barford and M. Crovella, "Generating Representative Web Workloads for Network and Server Performance Evaluation", in *the proceedings of 1998 ACM SIGMETRICS Conference*, Madison, WI, June 1998, pp.69-79.
- [8] Y. Diao, J. L. Hellerstein and S. Parekh, "Using Fuzzy Control to Maximize Profits in Service Level Management", *IBM Systems Journal*, volume 41, No. 3, March 2002, pp. 403-420.
- [9] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, et al., "RFC 2068: Hypertext transfer protocol – HTTP/1.1", January 1997, available from: <http://www.faqs.org/rfcs/rfc2068.html>.
- [10] E. Frachtenberg, F. Petrini, "Scheduler Testbed System Design", *Los Alamos National Laboratory (Technical Report-LAUR 01-4694)*, August 2001.
- [11] M. Gong and C. Williamson, "Quantifying the Properties of SRPT Scheduling", in *the proceedings of IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Orlando, FL, October 2003, pp. 126-135.

- [12] K. Gopalan and T. Chiueh, "Multi-Resource Allocation and Scheduling for Periodic Soft Real-Time Applications," in *the proceedings of ACM/SPIE Multimedia Computing and Networking Conference (MMCN 2002)*, San Jose, CA, January 2002, pp. 34-45.
- [13] K. Gopalan and T. Chiueh, "Real-Time Disk Scheduling Using Deadline Sensitive SCAN," Technical Report TR-92, Experimental Computer System Labs, Computer Science Department, SUNY at Stony Brook, January 2001.
- [14] J. Herlocker, J. Konstan, A. Borchers, J. Riedl, "An algorithmic framework for performing collaborative filtering." in *the proceedings of 1999 Conference on Research and Development in Information Retrieval*, Berkeley, CA, August 1999, pp. 230-237.
- [15] R. Jain, *The Art of Computer Systems Performance Analysis*, John Wiley & Sons, 1991.
- [16] E. Jensen, M. Carey, and M. Livny, "A Time-Driven Scheduling Model for Real-Time Operating Systems," in *the proceedings of IEEE 6th Real-Time Systems Symposium*, San Diego, CA, December 1985, pp. 112-122.
- [17] B. Joseph Pine II, Don Peppers, and Martha Rogers, "Do you want to keep your customers forever?" *Harvard Business School Review*, Vol. 73, No. 2, March-April 1995, pp. 103-114.
- [18] J. Konstan, B. Miller, D. Maltz, J. Herlocker, L. Gordon, and J. Riedl, "GroupLens: applying collaborative filtering to usenet news." *Communications of the ACM*, Vol. 40, No. 3, March 1997, pp. 77-87.
- [19] B. Laurie, and P. Laurie, *Apache- The Definitive Guide*, O'Reilly & Associates, 1999.
- [20] P. Li, "QoS-Enabled E-Commerce Web Application", M.C.S Thesis, School of Computer Science, Carleton University, Ottawa, Canada, December 2001.
- [21] Q. Li and B. Moon, "Distributed Cooperative Apache Web Server", in *the proceedings of 10th Int'l World Wide Web Conference*, Hong Kong, May 2001, pp 555-564.
- [22] D. A. Menasce and R. Dodge, "Preserving QoS of E-commerce Sites Through Self-Tuning: A Performance Model Approach," in *the proceedings of the 3rd ACM Conference in Electronic Commerce*, Tampa, October 2001, pp. 224-234
- [23] D. A. Menasce, V. A. F. Almeida, R. C. Fonseca, and M. A. Mendes, "Business-oriented Resource Management Policies for E-Commerce Servers," *Performance Evaluation*, Vol. 42, No. 2-3, September 2000, pp. 223-239.

- [24] D. A. Menasce, V. A. F. Almeida, *Scaling for E-Business Technologies, Models, Performance, and Capacity Planning*, Prentice Hall, 2000.
- [25] D. A. Menasce, V. A. F. Almeida, R. C. Fonseca, and M. A. Mendes, "A Methodology for Workload Characterization for E-commerce Sites," in *the proceedings of 1999 ACM Conference in Electronic Commerce*, Denver, CO, November 1999, pp. 119-128.
- [26] B. Mobasher, R. Cooley and J. Srivastava, "Automatic Personalization Based on Web Usage Mining", *Communications of the ACM*, Volume 43, No. 8, August 2000, pp. 142-151.
- [27] D. Mosberger and T. Jin, "httpperf - A Tool for Measuring Web Server Performance", *ACM SIGMETRICS Performance Evaluation Review*, Vol. 26, No. 3, March 1998, pp. 31-37.
- [28] S. Nadimpalli, "High Performance Web Server", M. Eng. Thesis, Department of Systems and Computer Engineer, Carleton University, Ottawa, Canada, December 2000.
- [29] K. M. Passino and S. Yurkovich, *Fuzzy Control*, Addison-Wesley Longman, 1998.
- [30] F. F. Reichheld and W. E. Sasser, Jr 1990, "Zero Defections: Quality Comes to Services," *Harvard Business School Review*, Vol. 68, No. 5, September-October 1990, pp. 105-111.
- [31] F. F. Reichheld, "Loyalty-Based Management," *Harvard Business School Review*, Vol.71, No. 2, February 1993, pp. 64-73.
- [32] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: An open architecture for collaborative filtering of netnews", in *the proceedings of ACM Conference on Computer-Supported Cooperative Work (CSCW' 94)*, Chapel Hill, NC, October 1994, pp. 175-186.
- [33] J. B. Schafer, J. Konstan, J. Riedl, "Recommender Systems in E-Commerce," in *the proceedings of ACM Conference on Electronic Commerce (EC-99)*, Denver, CO, November 1999, pp. 158-166.
- [34] L. Schrage, "A Proof of the Optimality of the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 16, No. 3, May-June 1968, pp. 678-690.
- [35] U. Shardanand, P. Maes, "Social information filtering: algorithms for automating "word of mouth," in *the proceedings of the Computer-Human Interaction Conference (CHI'95)*, Denver, CO, May 1995, pp. 210-217.

- [36] D. Smith, "A New Proof of the Optimality of the Shortest Remaining Processing Time Discipline", *Operations Research*, Vol. 26, No. 2, March-April 1976, pp. 197-199.
- [37] H. Tang, T. Yang, L. Chu, "Integrated Resource Management for Cluster-based Internet Services," in *the proceedings of the Fifth USENIX Symposium on Operating Systems Design and Implementation (OSDI'02)*, Boston, MA, December 2002, pp. 225-238.
- [38] G. Trent and M. Sake, "WebSTONE: The first generation in HTTP server Benchmarking", February 1995, available from: <http://www.mindcraft.com/webstone/paper.html>.
- [39] A. Wierman and M. Harchol-Balter, "Classifying Scheduling Policies with respect to Unfairness in an M/GI/1." In *the proceedings of ACM SIGMETRICS 2003 Conference on Measurement and Modeling of Computer Systems*, San Diego, CA. June 2003, pp. 238-249
- [40] "Apache Web Server Documentation", available from: <http://www.apache.org>.
- [41] "SPEC. An explanation of the SPECweb96 benchmark", December 1996, available from <http://www.specbench.org/osg/web96/webpaper.html>.