LUND UNIVERSITY

Some topics in web performance analysis

Cao, Jianhua

2004

Some Topics in Web Performance Analysis

Jianhua Cao

LUND UNIVERSITY

Department of Communication Systems
Lund Institute of Technology

ISSN 1101-3931
ISRN LUTEDX/TETS–1066–SE+99P

Printed in Sweden
E-kop
Lund 2004

To Jing

This thesis is submitted to Research Board FIME - Physics, Informatics, Mathematics and Electrical Engineering at Lund Institute of Technology, Lund University in partial fulfillment of the requirements for the degree of Licentiate in Engineering.

Contact information:
Jianhua Cao
Department of Communication Systems
Lund University
P.O. Box 118
SE-221 00 LUND
Sweden

Tel: +46 46 222 90 08
Fax: +46 46 14 58 23
e-mail: jcao@telecom.lth.se

Abstract

This thesis consists of four papers on web performance analysis. In the first paper we investigate the performance of overload control through queue length for two different web server architectures. The simulation result suggests that the benefit of request prioritization is noticeable only when the capacities of the sub-systems match each other. In the second paper we present an M/G/1/K*PS queueing model of a web server. We obtain closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. The model is validated through real measurements. The third paper studies a queueing system with a load balancer and a pool of identical FCFS queues in parallel. By taking the number of servers to infinite, we show that the average waiting time for the system is not always minimized by routing each customer to the expected shortest queue when the information used for decision is stale. In the last paper we consider the problem of admission control to an M/M/1 queue under periodic observations with average cost criterion. The problem is formulated as a discrete time Markov decision process whose states are fully observable. A proof of the existence of the average optimal policy by the vanishing discounted approach is provided. We also show that the optimal policy is nonincreasing with respect to the observed number of customers in the system.

Acknowledgments

I am grateful to Jing for everything I can think of. My deepest gratitude goes to Dr. Christian Nyberg for his luminous guidance and warm encouragement during the course of this research. I am indebted to Prof. Michał Pióro for giving me treasurable advice from time to time. Thanks Prof. Ulf Körner for carefully reading the manuscript and helpful suggestions. Gratitude also goes to my colleague Mikael Andersson for the cooperation made in one of the papers included in this thesis. Finally I want to thank all colleagues at the department and my dear friends here in Sweden.

Jianhua Cao
Lund, April 2004

Contents

1 Introduction

The World Wide Web (WWW) has been the driving force behind the phenomenal growth of the Internet since the 90's. Started with remote file retrieving, the web has evolved into a full-featured platform for many applications such as e-commerce [12]. The research area of web performance analysis is attracting more and more attention as the demand for quality of service guaranties increases in web applications [16, 4, 25].

This thesis consists of four papers on the topic of web performance analysis. The first paper studies the effectiveness of the request based and the session based overload control for a web server. We propose an M/G/1*PS queueing model for a web server in the second paper. The third paper discusses a load balancing strategy using stale state information. In the last paper, we study the optimal admission control to an M/M/1 queue under partial observations with average cost criterion.

The purpose of this introduction is to give a background and motivation for the included papers. The rest of this thesis introduction is organized as follows: in section 2, the architectures of web servers and clusters are reviewed. Section 3 discusses both models and related research issues for web performance analysis. Section 4 lists the four papers included in this thesis. In the last section, we outline some of our future research directions.

2 Architecture of a Web Server and a Cluster

In a simple scenario a web server can be just a networked computer running HTTP software that behaves according to the protocol specification. The most popular software for handling HTTP requests is Apache. The architecture of Apache follows a multi-process programming paradigm [11]. When the Apache receives an HTTP connection request, the request is first parsed by the main process of the Apache program. Then a separate process will be assigned to prepare the requested HTML file. When the requested HTML file includes only static contents, the file is then loaded from disks or caches. When the file includes dynamic contents, such as the result of a database inquiry, Apache forwards the request to one of its modular processes which was ignited when the Apache program was started. The multi-process paradigm allows many processes to prepare HTML files simultaneously, as many as several hundred. When too many processes are generated, the system performance starts to degrade as the cost of context switching between processes soars. It is, however, possible to configure and limit the maximum number of allowed processes in Apache.

The architecture model of Apache is sometimes referred as process-based servers [17]. Other well-known architecture models include: thread-based servers, event-driven servers and in-kernel servers. Threads are similar to processes but share the same address space. The cost of thread creation and context-switching is usually lower than the cost incurred for processes. However poorly programmed threads can crash the whole server. Web servers that use threads include JAWS and Sun's Java Web Server. Servers that use event-driven architectures include Flash and Zeus. With this architecture, a single process is used with non-blocking I/O(or asynchronous I/O). Since there are no context-switch costs and no extra memory consumption which is the case with threads or processes, event-driven servers are usually fast. The down side is, however, like that of thread-based servers, a single failure can halt the whole server. Moreover different operating systems have varying levels of support for asynchronous I/O operations. All previous architectures place the Web server software in user space, in the in-kernel servers such as AFPA and Tux, the HTTP server is tightly integrated with the host's TCP/IP stack. This type of server is extremely fast since potentially expensive transitions to user space are avoided. But this approach is least robust to programming errors. A single server fault can crash the whole machine.

A web site may be just one computer with a fast CPU and lots of memory running Linux, Apache or perhaps even MySQL. This simple solution is very popular for small websites where traffic is usually low. But clearly this one-box-for-all approach isn't scalable for handling a large volume of transactions [8].

A simple scalable solution is to use a load balancing computer as front-end and several identical web servers plus one or two databases behind the scene, [18, 24, 2]. A more sophisticated configuration involves adding application servers for dynamic contents and dedicated servers for searching. In that case, the web servers concentrate on static contents such as files and images. Often there is another load balancer for the application servers. This approach is usually referred to as web clusters. Harkins, for example in [15], describes in great detail an e-commerce web site called etoys.com which employs this architecture.

It is worth mentioning that, employing caching carefully will boost performance dramatically in a web cluster because of the reduced communication delay between servers [5, 6]. More than 90% hit ratio is easily achievable in practice. Harkins reported a 99% hit ratio for etoy.com during the 2002 Christmas season [15].

A multi-tier caching strategy can be implemented as follows for a web cluster that separates the web servers, application servers and search servers. The first level cache

catches the transactions between the web servers and the application servers. The contents of first level cache can be shared among the web servers. The second level cache resides inside the application servers and use IP-multicast to communicate with each other.

3 Models and Research Issues

Different problems in performance analysis are associated with different models which can be roughly classified into two categories, descriptive models and control models. Queueing models of web servers and web traffic are considered descriptive. Admission control, load balancing and scheduling are, on the other hand, linked to control models.

3.1 Queueing System Models of a Web Server

Most queueing models of web servers capture an important behavior of the response time, i.e. that it increases substantially as the server utilization approaches unity [30].

If quantitative characteristics of the throughput and the response time are of interest, an M/G/1/K*PS queue model serves the objective well. It is natural to use processor sharing in the model since different requests are served by different processes which are served in a round-robin fashion. Two parameters of the model, the mean service time and the maximum number of customers in the system can be accurately estimated from measurements in a controlled environment. The first paper of this thesis covers this modeling approach in detail.

In a server, a request can be queued at more than one place which makes it natural to use a queueing network model. Van der Mei, for example [20, 26], proposed a queueing network model consisting of three parts, TCP, HTTP and network IO subsystems, in sequel. The TCP subsystem is modeled as a multi-server system without buffer. The HTTP subsystem is modeled as a finite buffer with multi-servers. The IO subsystem is modeled as a polling system with finite buffer. Even though this model reflects the real transaction flow of an HTTP request, there are no explicit expressions for performance metrics such as throughput and response time. Hence further research and investigation using that model has to rely on simulation to a large extent. The second paper of this thesis used this queueing network model to investigate how the admission control can be affected by the server configuration.

3.2 Web Traffic Models

Noticeable results in web traffic modeling and analysis can be summarized as follows [18, 3, 2, 21, 19],

1. The traffic follows a day and week fluctuation pattern: busiest during the daytime hours, less busy during the evening and nighttime hours and least busy on weekends. But it can be regarded as constant in a short time scale such as an hour.

2. The traffic logs at the HTTP level show a bursty characteristic and a strong correlation of inter-arrival times for a duration of several minutes.

3. The traffic logs in the session level, however, show that inter-arrival times of session initialization request are almost independent.

The arrival process of HTTP requests to a web server is known to be stationary but correlated in a short time scale such as an hour. The origin of the correlation is that a real customer tends to send several requests in sequel when browsing at a web site. This fact motivates a simple request process model that the sequence of initial requests from customers follows a Poisson process and served requests feed back to the server as new requests with a certain probability and delay. The feedback probability and the distribution of the delay are two major components of the session model besides the arrival process of the initial requests. This simple session model is used for describing customer behavior in the first paper of this thesis.

Customer Behavior Model Graph(CBMG), proposed by Menasce et al. [22], is a more sophisticated model for modeling the dependence between requests. Different types of HTTP requests, such as browsing and searching, are represented by different states. The transition probability between states can be estimated from HTTP traffic logs. Exploiting CBMG for the purpose of QoS control has been reported by [23, 7].

3.3 Admission Control

For a simple queueing system, such as the M/M/1 queue, the queue length based overload control which rejects arrivals when the queue length reaches a given threshold is effective to guarantee the response time. However, there are two possible catches to put this simple method into practice. The first one is that many queues are formed inside the system as a web server is a collection of coordinated software and hardware. One can argue, perhaps,

that it is the bottleneck queue that deserves attention. But this argument is flawed since the location of the bottleneck depends the characteristic of the carried workload which varies [3]. Moreover it is preferable in practice that a rejection, if it happens, should be not too early so that a friendly message can be sent to inform the misfortune customers and not too late so that the amount of wasted work is minimized. The second problem in practice is that the overhead of monitoring the queue and adjusting the admission control decision continuously is usually high. Thus the decision has to be made based on incomplete state information where the theory is under-developed. The last paper of this thesis explores this topic.

One major direction of current research in overload control for web servers is taking sessions into account, where a session is a sequence of related requests sent by a customer [9, 10]. The main motivation is that rejecting a browsing request is preferred over abandoning a paying request.

In the first paper of this thesis, we use simulation to show that session based admission control may lose effectiveness when the capabilities of different subsystems such as HTTP processing and network IO are mismatched.

Control theory can also be applied to the overload control of web servers, see e.g. [27, 1, 28].

3.4 Load Balancing

Load balancing is an important architecture component for a web cluster. In theory, an HTTP request should be forwarded to the lightest loaded sever among those in the cluster. In practice, however, the randomized strategies were shown to be stable and easy to implement [24].

The third paper of this thesis investigates why in some cases a seemly "smart" load balancing strategy which makes routing decisions based on incomplete state information is even worse than the random selection.

Load balancing can be at TCP/IP level or at HTTP level. For TCP/IP level load balancing, a single domain name is mapped to multiple servers. Different TCP/IP connections are forwarded to different servers. It is also possible to do load balancing at HTTP application level and that is perhaps the only way when the web server cluster is tightly coupled, i.e. only a subset of the servers in the cluster can accomplish the given task. However, since the HTTP request has to be parsed before it is forwarded, the load balancer itself becomes a potential bottleneck. The current practice for e-commerce sites is to use

a TCP/IP level load balancer for the front end servers and an HTTP level load balancer for application servers.

3.5 Scheduling

The commonly used web server software Apache, assigns a process to each request. The scheduling of different processes is then dependent on the underlying operating system. For Unix and Windows, for example, processes of the same priority, which is the case for all HTTP processing tasks, are served in round robin fashion. This suggests that a processor sharing scheduling model can be used for analyzing the "default" scheduling scheme. Recent work show that the Shortest Remaining Processing Time (SRPT) scheduler can relive the problem of transient overload without starving long jobs [14, 29]. Multi-class priority queues are also proposed in order to get close to SRPT performance in response time and provide much better response time variance property [13].

4 List of Papers

The following papers have been included in this thesis:

1. Jianhua Cao and Christian Nyberg
   On Overload Control through Queue Length for Web Servers
   16th Nordic Teletraffic Seminar (NTS), Aug 2002

2. Jianhua Cao, Mikael Andersson, Christian Nyberg and Maria Kihl
   Web Server Performance Modeling using an M/G/1/K*PS Queue
   International Conference on Telecommunication (ICT), Feb 2003

3. Jianhua Cao and Christian Nyberg
   An Approximate Analysis of Load Balancing Using Stale State Information for Servers in Parallel
   The 2nd IASTED International Conference on Communications Internet and Information Technology (CIIT) Nov 2003

4. Jianhua Cao and Christian Nyberg
   Admission Control to an M/M/1 Queue under Periodic Observations with Average Cost Criterion
   Submitted to Operations Research, April 2004

In the first paper we investigate the performance of overload control through queue length for two different web server architectures. Both of them use finite queue lengths to prevent servers from being overloaded. One architecture prioritizes requests from established sessions while the other treats all requests equally. We use simulation to evaluate the performance of these two types of web servers. The result suggests that the benefit of request prioritization is noticeable only when the capacities of the sub-systems match each other.

In the second paper we present an M/G/1/K*PS queueing model of a web server. The arrival process of HTTP requests is assumed to be Poissonian and the service discipline is processor sharing. The total number of requests that can be processed at the same time is limited to K. We obtain closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. The average of the service time requirement and the maximum number of requests being served simultaneously are model parameters. The parameters are estimated through maximum likelihood estimation. We validated the model through real measurements. The performance metrics predicted by the model fit well to the experimental outcome.

In the third paper, we study a queueing system with a load balancer and a pool of identical FCFS queues in parallel. The arrival process is assumed to be Poisson and the service times have identical independent exponential distributions. The pool of servers informs the load balancer on the number of customers in each server at some regularly spaced time instances. The load balancer routes each customer to the expected shortest queue based on available stale information and elapsed time since the last time instance of system state information updating. We analyze the system performance approximately by taking the number of servers to infinite. We show that the average waiting time for the system is not always minimized by routing each customer to the expected shortest queue when the information used for decision is stale even if prediction of queue length is used.

In the last paper we consider the problem of admission control to an M/M/1 queue under periodic observations with average cost criterion. The admission controller receives the system state information every τ:th second and can accordingly adjust the acceptance probability. For a period of τ seconds, the cost is a linear function of the time average of customer populations and the total number of served customers in that period. The objective is to find a stationary deterministic control policy that minimizes long run average cost. The problem is formulated as a discrete time Markov decision process whose states are fully observable. The model in question generalizes two classical queueing control problems: the open and the closed loop admission control to an M/M/1 queue when taking $\tau \to \infty$

and $\tau = 0$ respectively. A proof of the existence of the average optimal policy by the vanishing discounted approach is provided. Several useful lower and upper bounds of the optimal cost are obtained. We also show that the optimal policy is nonincreasing with respect to the observed number of customers in the system.

5 Future Work

As the web penetrates our everyday life, the following scenario is becoming a commonplace. A web server farm consisting of several hundreds servers of different capabilities and capacities provides several types of web services, such as product inquiry and ordering. Some kind of quality of service or service level agreement between the owner of the web server farm and the web service providers is a necessity for carring out the business. As an extremely simplified example, a possible web service level agreement can be as follows. There are N types of web services. For a service request of type i, $i = 1, \cdots, N$, the availability must be greater than p_i and the response time must be less than W_i, when the average arrival rate for the requests of type i is less than λ_i. Many questions concerning capacity planning, admission control and scheduling may arise immediately and can be considered under the framework of a multi-class multi-server queueing system. Currently we are studying the problem of admission control and load balancing of heterogeneous servers in parallel within the service level agreement settings mentioned above. Often straightforward formulations of the problems lead to nonlinear programs which are difficult to solve even numerically. However, we can relate the nonlinear programs to some linear programs using arguments from queueing theory. Usually these linear programs provide both the upper bound and the lower bound of the objective for the nonlinear program in question. Thus our immediate concern is to characterize the difference of the optimal objective values of two linear programms that approximate the original nonlinear program when the underlying modeling prarameters are taken from some distributions. Moreover we plan to explore similar problems with more elaborated service level agreements and study their solutions using different approximation schemes.

References

[1] Abdelzaher, T. F., Shin, K. G., and Bhatti, N. (2002). Performance guarantees for Web Server End-Systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1).

[2] Arlitt, M., Krishnamurthy, D., and Rolia, J. (2001). Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1).

[3] Arlitt, M. F. and Williamson, C. L. (1997). Internet web servers: Workload characterization and performance implications. *IEEE/ACM Transactions on Networking*, 5(5).

[4] Bhoj, P., Ramanathan, S., and Singhal, S. (2000). Web2K: Bringing QoS to web servers. In *HP Laboratories Palo Alto, Hewlett-Packard Company*. HPL-2000-61.

[5] Bunt, R. B. and Eager, D. L. (1999). Achieving load balance and effective caching in clustered web servers. In *Proceedings of the 4:th International Web caching Workshop*.

[6] Caceres, R., Douglis, F., Feldmann, A., Glass, G., and Rabinovich, M. (1998). Web proxy caching: The devil is in the details. *Performance Evaluation Review*, 26(3):11–15.

[7] Chen, H. and Mohapatra, P. (2003). Overload control in QoS-aware web servers. *Computer Networks*, 42:119–133.

[8] Cherkasova, L. (1999). Scalabe web hosting service. Technical report, HP Laboratories Palo Alto, Hewlett-Packard Company. HPL-1999-52(R.1).

[9] Cherkasova, L. and Phaal, P. (1998). Session based admission control: a mechanism for improving the performance of an overloaded web server. Technical report, Computer Systems Laboratory, Hewlett-Packard Company. HPL-98-119.

[10] Cherkasova, L. and Phaal, P. (1999). Hybrid and predictive admission strategies to improve the performance of an overloaded web server. Technical report, HP laboratories Palo Alto, Hewlett-Packard Company. HPL-98-125(R.1).

[11] Dragoi, O. A. (1999). The conceptual architecture of the Apache web server. Technical report, Department of Computer Science, University of Waterloo.

[12] Feldman, S. (2000). The changing face of e-commerce: Extending the boundaries of the possible. *IEEE Internet Computing*.

[13] Ghosh, S. and Squillante, M. S. (2003). Analysis and control of correlated web server queues. In *Proceedings of SPIE*, volume 5244.

[14] Harchol-Balter, M., Bansal, N., Schroeder, B., and Agrawal, M. (2000). Implementation of {SRPT} scheduling in web servers. Technical report, Department of Computer Science, Carnegie Mellon University. CMU-CS-00-170.

[15] Harkins, P. (2001). Building a large-scale E-commerce site with Apache and mod_perl. Technical report, perl.com.

[16] Iyengar, A., MacNair, E., and Nguyen, T. (1997). An analysis of web server performance. In *Globecom*.

[17] Iyengar, A., Nahum, E., Shaikh, A., and Tewari, R. (2002). Enhancing Web Performance. In *Proceedings of the IFIP World Computer Congress*.

[18] Kwan, T. T., McGrath, R. E., and Reed, D. A. (1995). NCSA's World Wide Web server: Design and performance. *IEEE Computer*, 28(11):68–74.

[19] Liu, Z., Niclausse, N., and Jalpa-Villanueva, C. (2001). Traffic model and performace evaluation of web servers. *Performance Evaluation*, 46:77–100.

[20] Mei, R. D. V. D., Hariharan, R., and Reeser, P. (2001). Web server performance modeling. *Telecommunication Systems*, 16(3,4):361–378.

[21] Menasce, D., Almeida, V., Riedi, R., Ribeiro, F., Fonseca, R., and Jr., W. M. (2003). A hierarchical and multiscale approach to analyze E-business workloads. *Performance Evaluation*, 54:33–57.

[22] Menasce, D. A., Almeida, V. A. F., Fonseca, R., and marco A. Mendes (1999). A methodology for workload characterization of e-commerce sites. In *Proceedings of 1999 ACM conference on Electronic Commerce Deaver, Colorado*.

[23] Menasce, D. A., Almeida, V. A. F., Fonseca, R., and Mendes, M. A. (2000). Business-oriented resource management policies for e-commerce servers. *Performance Evaluation*, 42:223–239.

[24] Mosedale, D., Foss, W., and McCool, R. (1997). Lessons learned administering Netscape's Internet site. *IEEE Internet Computing*, 1(2):28–35.

[25] Nahum, E., Barzilai, T., and Kandlur, D. D. (2002). Performance issues in WWW servers. *IEEE/ACM Transactions on Networking*, 10(1).

[26] Reeser, P. K., van der Mei, R. D., and Hariharan, R. (1999). An analytic model of a web server. In *Proceedings of the International Teletraffic Congress ITC-16*.

[27] Robertsson, A., Wittenmark, B., and Kihl, M. (2003). Analysis and design of admission control in web-server systems. In *American control Conference ACC*.

[28] Sahai, A., Ouyang, J., Machiraju, V., and Wurster, K. (2001). Specifying and guaranteeing quality of service for web services through real time measurement and adaptive control. Technical report, E-services Software Research Department. HP-2001-134.

[29] Schroeder, B. and Harchol-Balter, M. (2002). Web servers under overload: How scheduling can help. Technical report, School of Computer Science, Carnegie Mellon University. CMU-CS-02-143.

[30] Slothouber, L. P. (1995). A model of web server performance. lpslot@biap.com.

On Overload Control through Queue Length for Web Servers

Jianhua Cao and Christian Nyberg

Presented at the 16:th Nordic Teletraffic Seminar, August 2002, Espoo Finland

Abstract. We investigate the performance of overload control through queue length for two different web server architectures. Both of them use finite queue lengths to prevent servers from being overloaded. One architecture prioritizes requests from established sessions while the other treats all requests equally. First, we introduce queueing models for these two systems. Then, we define and explain a new web server performance metric that is a function of session throughput, error rate for connection within sessions and average request response time. Finally, we use simulation to evaluate the performance of these two types of web servers. The result suggests that the benefit of request prioritization is noticeable only when the capacities of the sub-systems match each other.

1 Introduction

The excessive delay of web services is more and more common as the number of Internet users increases everyday. Not only the customers will be unsatisfied but also the service provider will be hurt in the long term. There are two ways to alleviate the problem, namely, infrastructure upgrading and overload control. Infrastructure upgrading is an obvious approach and will bring more traffic and, probably, more profit to the operator. Overload control, however, assures a certain quality of service to a limited amount of customers and sacrifices the rest by throttling the incoming traffic.

Overload control may not be the best thing to do when infrastructure upgrading is possible, but it is still a necessary complement to upgrading approach for two reasons. First, one can not upgrade the system so often as to keep up with the ever increasing Internet traffic. Second, the large variance of Internet traffic will cause the web server overload from time to time even if the web server is engineered to handle the average traffic. So how could overload control be done? A crude overload control mechanism will constantly monitor the server's CPU usage, connection response time, the number of current connections (jobs) and/or the number of queued requests. When certain parameters exceed some predetermined levels, the server starts to reject customers until monitored parameters are back to normal. However, this simple scheme has a problem that all requests are equally treated.

A customer visiting a web site tends to send several requests in sequel. A possible sequence could contain the following commands: browsing, searching, ordering, paying and exiting. We call such a sequence of requests a session. As one can see here, the customer will be very irritated if her paying request is rejected after filling in her credit card number. In general, requests within a session should not be rejected. When a overload control scheme is blind at sessions, it is called request-based overload control (RBOC) otherwise session-based overload control (SBOC).

Different overload control strategies for telephone switches have been studied by Nyberg [5]. The performance of SBOC and RBOC for e-commerce web sites has been studied by Kihl and Widell [2] recently. Several attempts [1, 4, 6] have been tried to model the web servers and have different level of success.

In this paper, we use simulation to study the effectiveness of SBOC and RBOC through queue length. Overload control through queue length means that the control decision is based on the number of queued requests. The result shows that the SBOC through queue length is not necessarily effective when the web server is not carefully configured.

The paper is organized as follows. In Section 2, we introduce the queuing models of a web server. We then define and explain a performance metric for web servers in Section 3. Section 4 gives the simulation results of web servers using SBOC and RBOC with three different configurations. We conclude the paper in Section 4.

2 Queuing Models of Web Servers

The basic queuing model used here is based on [4]. The model consists of three subsystems, TCP, HTTP and IO, in tandem. Fig. 1 shows the structure.

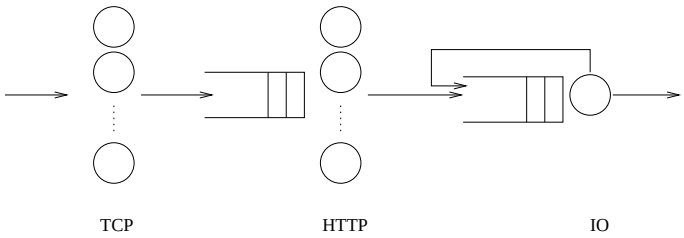


Figure 1: A basic queueing model of web server consists of three subsystems,TCP, HTTP and IO in tandem.

The TCP subsystem is modeled as a multi-server system with zero buffer. The number of TCP servers, m_{tcp} is equal to the maximum number of allowed concurrent TCP connections. The HTTP subsystem is modeled as a multi-server system with a finite buffer. The number of HTTP servers, m_{http}, is equal to the maximum number of allowed concurrent HTTP processes or threads. The IO subsystem is modeled as a processor sharing server with a finite buffer. The buffer size of HTTP subsystem, n_{http}, and the buffer size of the IO subsystem, m_{io}, along with m_{tcp} and m_{http}, are all configurable parameters.

The model works as follows. When a HTTP request arrives, the TCP subsystem will establish a connection between the client and the web server by handshake. So the service time of the TCP subsystem, x_{tcp}, is equal to the the round trip time, t_{rtt}. Note that the TCP subsystem will allow at most m_{tcp} simultaneous connections .

After the connection is established, the request is forwarded to the HTTP subsystem and the TCP server will be released. The request will be processed immediately if there is a free HTTP server available otherwise it will be pushed into the FIFO queue. The service time of the HTTP subsystem, x_{http}, is the total time spending on parsing the HTTP requests and fetching/generating the HTML files. We could reasonably assume that x_{http}

is proportional to the returned file size with coefficient, k(with unit s/KB). As soon as the returned files are compiled, the job will be transfered to the IO subsystem when there is an IO slot available, otherwise the HTTP server that processes the request will be hold. The HTTP server on holding state can not process more HTTP requests until it is freed.

The IO subsystem uses the TCP protocol to send the HTML files. The bandwidth of IO server is shared by all jobs in the queue. The IO server polls jobs in a round robin fashion. The service time of an IO job, x_{io}(with unit seconds), depends on the file size, s_{file}(with unit KB), the bandwidth of the server, w_{server}(with unit KB/s), and the client, w_{client}, the number of concurrent IO jobs, $n_{\text{io}}(\leq m_{\text{io}})$, and the packet loss rate of the connection, p_{loss}.

To simplify the derivation of x_{io}, we use the following four assumptions. First, all the clients use the same maximum segment size, s_{mss} (with unit KB). Second, all the clients have the same packet loss rate of zero, $p_{\text{loss}} = 0$. Third, all the connection have the same TCP flow/congestion control window size, s_{wnd}(with unit KB), all the time. Fourth, the bandwidth of client is less than that of server. So, the approximated IO service time is given by Eq.1.

$$x_{\text{io}} = \left\lceil \frac{s_{\text{file}}}{s_{\text{wnd}}} \right\rceil t_{\text{rtt}} + \frac{s_{\text{file}}}{w_{\text{client}}} + \left\lceil \frac{s_{\text{file}}}{s_{\text{mss}}} \right\rceil \frac{s_{\text{mss}}}{w_{\text{server}}}(n_{\text{io}} - 1) \tag{1}$$

The first item in the right hand side of the equation is the transmission time of the file; the second item is the time for client to receive the file; the third item is the waiting time.

The model described above can be used to predict the throughput, the response time of the web server quite accurately. But it needs some modification when the requests are grouped in sessions, otherwise the arrival process is not a Poisson process any more.

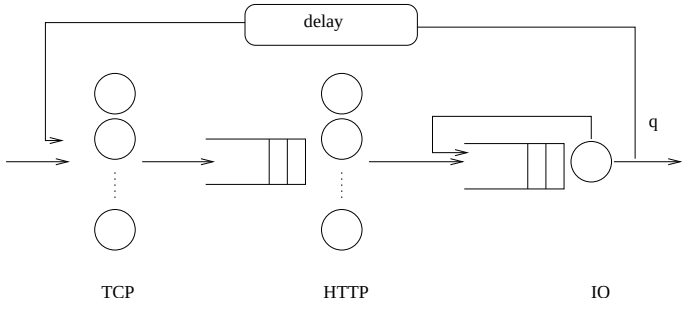


Figure 2: The session is modeled by Bernoulli feedback. If the feedback probability is q, then average number requests in a session is $1/(1-q)$.

In our case, we model the sessions by adding a Bernoulli feedback from the output to the input of the model above, as shown in Fig. 2. So it implies that the number of requests

in a session is geometrically distributed. This fact is justified by the statistical analysis of web traffics [3]. Since the transient analysis is not our major interests, the delay between requests is not modeled explicitly here.

The HTTP server with FIFO queue of finite length is in fact an implementation of RBOC through queue length. Therefore the model above can be used to investigate the performance of RBOC. The HTTP subsystem of SBOC is a bit different from that of RBOC, as illustrated in Fig. 3. Let us call the customers from the feedback the old customers and the others new customers. The SBOC uses two separate queues in the HTTP subsystem: one for new customers and one for old customers. The queue for old customers is assigned with a higher priority in order to break as little number of sessions as possible.

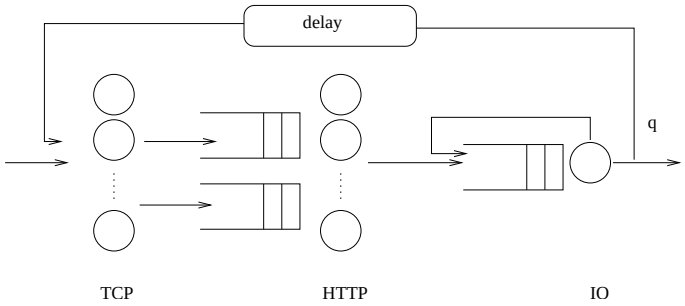


Figure 3: The model of web server using SBOC scheme. Two queues are used for old customers and new customers. The queue for old customers has higher priority than the other one.

3 Web Server Performance Metrics

Three metrics, throughput, H, response time, T, and connection error rate,E are widely used to evaluate the performance of web servers. Throughput is defined as the average number of completed requests (or sessions) per second. Typically, the throughput will reach some limit when the arrival intensity, λ, i.e. the number of arrived customers per second, exceeds some threshold, λ^*. Below that threshold, the throughput will increase linearly with λ. The response time measures the interval between when request is sent and when the requested files are received by the customer. It should be a small constant when $\lambda < \lambda^*$ and increase when λ is close to λ^*. If the web server uses finite buffers, the response time will be a large constant when $\lambda > \lambda^*$. The connection error rate reflects

the percentage of requests (or sessions) that are rejected (or broken) due to the limited capacity of the web server. It should be approximately zero when $\lambda < \lambda^*$, and approach one when λ becomes large.

In order to compare overall performance of RBOC and SBOC, we need a unified metric. In analogy with the definition of network power which is throughput divided by delay, we propose a metric called the power of web server, denoted by P. It is a function of session throughput, error rate for sessions and average request response time:

$$P = \frac{H_{\text{session}}}{T_{\text{request}}}(1 - E_{\text{session}}) \tag{2}$$

where H_{session} is the number of completed sessions per second, T_{request} is the average response time for each accepted request, E_{session} is the probability that a session is being broken.

Since H_{session}, T_{request} and E_{session} are all functions of the arrival intensity λ, and so is the server power, P. We would expect the power of an ideal server to have the following properties: First, it should be proportional to λ when $\lambda < \lambda^*$. Second, it maintains the maximum power, $P(\lambda^*)$, when $\lambda > \lambda^*$. As we can see later, the RBOC cannot maintain the second property and the power of web server using RBOC will drop. So can SBOC maintain maximum power constantly even if the server is heavily loaded? Our answer is "it depends" as the simulation shows.

4 Simulation Results

We use discrete event simulation to investigate the performance of web servers using RBOC and SBOC.

To avoid unnecessary complication, we limit the sources of randomness in our simulation to be two: the customer arrival process assumed to be Poissonian and the session length which is geometrically distributed with mean $1/(1-q) = 6.0$. The following parameters are assumed to be constants: the size of file that clients request, $s_{\text{file}} = 10\text{KB}$; the speed of file feteching, $k = 0.01$ s/KB; the client bandwidth, $w_{\text{client}} = 5.5\text{KB/s}$; the round trip time between client and server, $t_{\text{rtt}} = 0.1\text{s}$; the maximum segment size, $s_{\text{mss}} = 0.5\text{KB}$.

The following parameters in both models are configurable: the number of TCP servers, m_{tcp}; the number of HTTP servers, m_{http}; the maximum queue length of HTTP subsystem, n_{http}; the maximum buffer size of IO subsystem, m_{io}; and the bandwidth of the web server, w_{server}.

Let C denote the capacity of a system, i.e. the maximum number of requests that can be proceeded in one second. The capacities of TCP, HTTP, IO subsystems are then given by:

$$C_{\text{tcp}} = \frac{m_{\text{tcp}}}{x_{\text{tcp}}} = \frac{m_{\text{tcp}}}{t_{\text{rtt}}} \tag{3}$$

$$C_{\text{http}} = \frac{m_{\text{http}}}{x_{\text{http}}} = \frac{m_{\text{http}}}{k \cdot s_{\text{file}}} \tag{4}$$

$$C_{\text{io}} = \frac{m_{\text{io}}}{x_{\text{io}}|_{n_{\text{io}}=m_{\text{io}}}} \tag{5}$$

Given that k, s_{file} and t_{rtt} are constants, the capacities of the sub-systems, C_{tcp}, C_{http}, C_{io}, are then fully determined by the system configuration.

In most configurations, it will usually be the case that $C_{\text{tcp}} > C_{\text{http}}$ and $C_{\text{tcp}} > C_{\text{io}}$. So we limit our investigation to the following three cases: $C_{\text{http}} > C_{\text{io}}$ (case A), $C_{\text{http}} = C_{\text{io}}$(case B), $C_{\text{http}} < C_{\text{io}}$ (case C).

Further we fix the configurations: $m_{\text{tcp}} = 1024$, $n_{\text{http}} = 100$, $m_{\text{io}} = 507$ and $w_{\text{server}} =$ 100Mbits/s, to make C_{io} a constant. The variation of C_{http} is then achieved by adjusting the number of HTTP servers, m_{http}.

We show the different configurations of m_{http} and corresponding subsystem capacities in three cases in Table 1. The simulation results are shown in Fig. 4 and Fig. 5.

In Fig. 4, we plot the session throughput, average request response time and session error rate for systems using RBOC and SBOC in three different cases. The sub-figure (a) and (b) show that RBOC and SBOC give almost identical session throughput. But SBOC gives lower response than RBOC in all three cases as sub-figures (c) and (d) indicate. The sub-figure (e) gives expected behavior of session error rate of RBOC that approaching to one when arrival intensity exceeds some threshold. However the session error rate of SBOC in case A shows the the same pattern. Recall that in case A, the capacity of HTTP subsystem is greater than that of IO subsystem, therefore HTTP servers tend to be held when a job completes. When there are some HTTP servers are held, the effective number of servers will decrease and queues of HTTP subsystem will tend to pile up.

In Fig. 4, we show the power of systems using RBOC and SBOC. Now it is much more straightforward to compare RBOC and SBOC in three different configurations. Let us first consider the case B. It is clear from the simulation results that the power of the system using SBOC does not drop even when the arrival intensity exceeds the threshold λ^*, while it is not the case for the system using RBOC. However in case A, the power of two systems

	case A	case B	case C
m_{http}	24	12	6
C_{tcp}	10240	10240	10240
C_{http}	240	120	60
C_{io}	120.057	120.057	120.057

Table 1: Different configuration of m_{http} and the capacities of subsystems in different cases. case A: $C_{\text{http}} > C_{\text{io}}$; case B: $C_{\text{http}} = C_{\text{io}}$; case C: $C_{\text{http}} < C_{\text{io}}$.

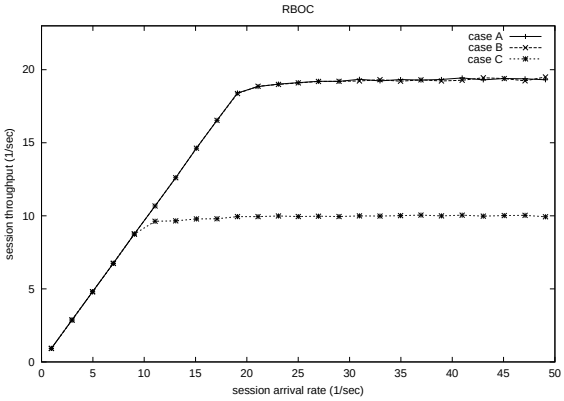
using RBOC and SBOC is the same. The reason is mainly due to that in both systems the session error rate increases as the arrival intensity increases. In case C, we see some advantages of SBOC over RBOC but the gain is not that much as in case B. We also notice that the power of the two systems in case C is the half of that in case B. If checking the Table 1, we will find out that the number of HTTP servers in case C is half of that in case B. Obviously the configuration of case C makes the web server under utilized.

So finally, we can reach a conclusion that SBOC will have its largest advantage over RBOC when the web server is configured in such a way that the capacities of subsystems match each other.

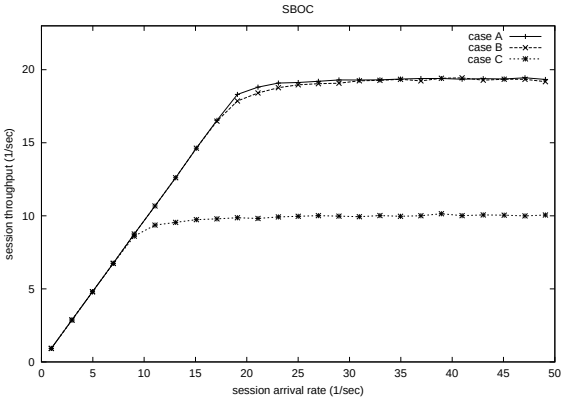
5 Conclusion Remarks

In this paper, we first introduce the queuing model of web servers with SBOC and RBOC respectively. To facilitate the investigation, we define a new performance metric called the power of web server. We use simulation to investigate the performance of two overload control schemes in the cases that the capacity of the HTTP subsystem does/does not match that of the IO subsystem. The result suggests that SBOC is effective when the web server is properly configured so that the capacities of the subsystems match each other.

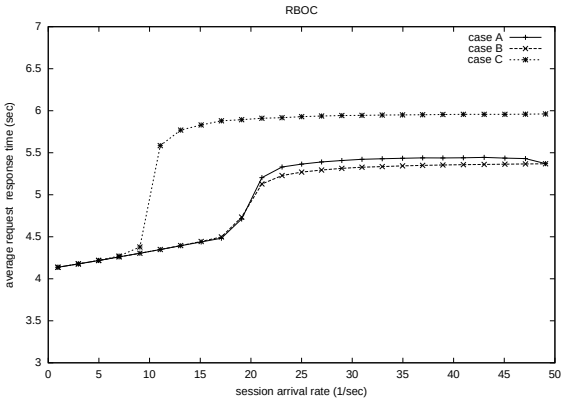
For the future work, first, it is preferred to use analytical model over simulation model in order to find out the region of the system parameters that renders the different overload control schemes effective/ineffective. Second, a more realistic model of the web server is needed. In order to simplify both simulation and analysis, the current web server model ignores the fact that the tasks like HTTP parsing, network and file IO operation are all sharing CPU, network bandwidth and hard-disk. Hence a processor sharing based queueing network model could reveal more interesting dynamics of the web server.



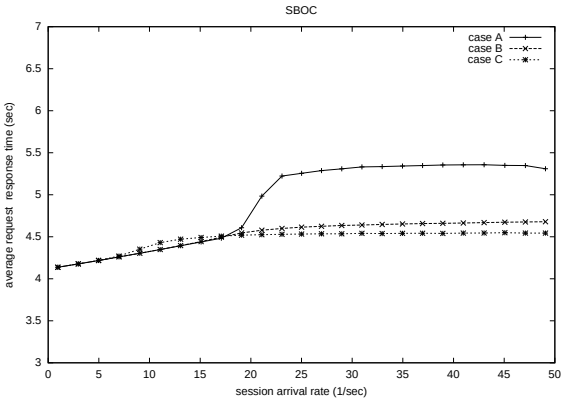
(a) Throughput (RBOC)



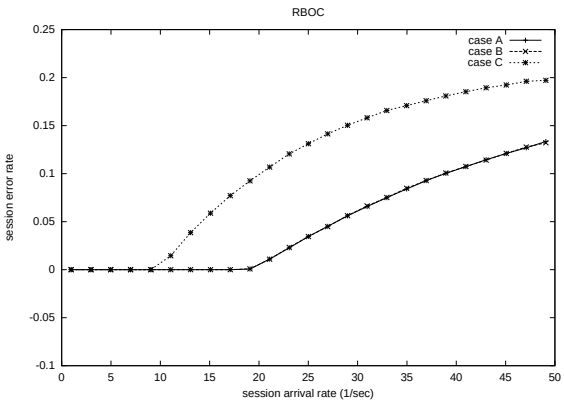
(b) Throughput (SBOC)



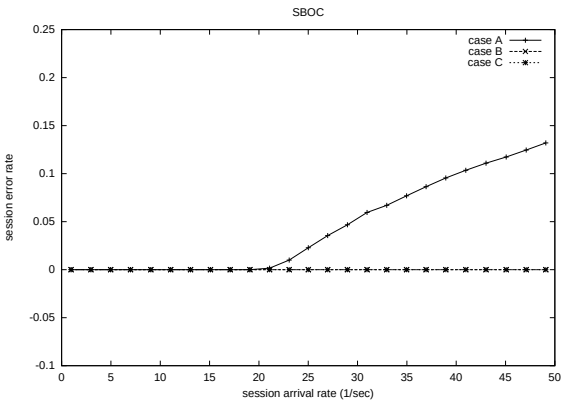
(c) Average Response Time (RBOC)



(d) Average Response Time (SBOC)

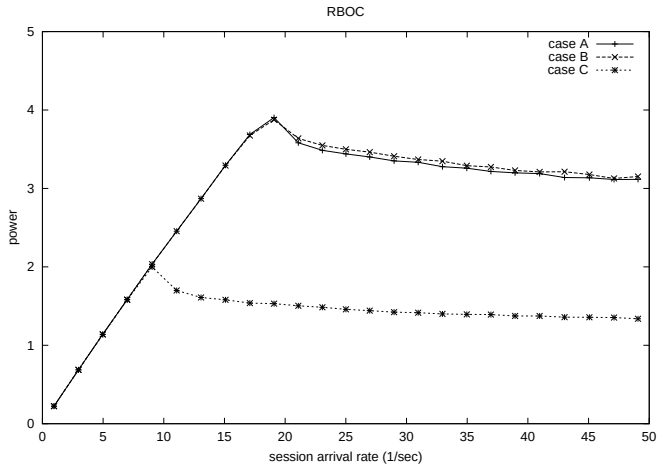


(e) Session Error Rate (RBOC)

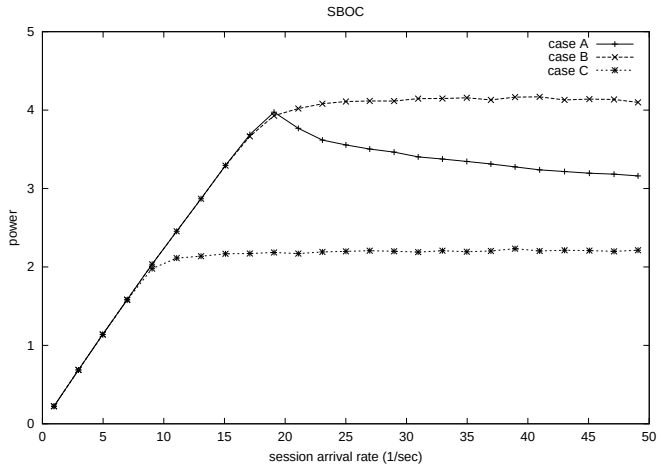


(f) Session Error Rate (SBOC)

Figure 4: The throughput, average request response time and session error rate of web servers using RBOC and SBOC in three different cases.



(a) Power (RBOC)



(b) Power (SBOC)

Figure 5: The power of web servers using RBOC and SBOC in three different cases.

References

[1] Heidemann, J., Orbraczka, K., and Touch, J. (1997). Modeling the performance of http over several transport protocols. *IEEE/ACM Transactions on Networking*, 5(5).

[2] Kihl, M., Widell, N., and Nyberg, C. (2002). Performance modeling of distributed e-commerce sites. In *Networking*.

[3] Liu, Z., Niclasusse, N., and Jalpa-Villanueva, C. (2001). Traffic model and performance evaluation of web servers. *Performance Evaluation*, 46.

[4] Mei, R. D. V. D., Hariharan, R., and Reeser, P. K. (2001). Web server performance modeling. *Telecommunication Systems*, 16(3,4):361–378.

[5] Nyberg, C. (1992). *On Overload Control in Telecommunication Systems*. PhD thesis, Lund University.

[6] Reeser, P. K., van der Mei, R. D., and Hariharan, R. (1999). An analytical model of a web server. In *ITC16*, pages 1199–1280.

Web Server Performance Modeling Using an $M/G/1/K*PS$ Queue

Jianhua Cao, Mikael Andersson, Christian Nyberg and Maria Kihl

Presented at the 10:th International Conference on Telecommunication, February 2003, Tahiti French Polynesia

Abstract. Performance modeling is an important topic in capacity planning and overload control for web servers. We present an M/G/1/K*PS queueing model of a web server. The arrival process of HTTP requests is assumed to be Poissonian and the service discipline is processor sharing. The total number of requests that can be processed at one time is limited to K. We obtain closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. The average of the service time requirement and the maximum number of requests being served simultaneously are model parameters. The parameters are estimated by maximizing the log-likelihood function of the measured average response time. Compared to other models, our model is conceptually simple and it is easy to estimate model parameters. The model has been validated through measurements in our lab. The performance metrics predicted by the model fit well to the experimental outcome.

1 Introduction

Performance modeling is an important part of the research area of web servers. Without a correct model of a web server it is difficult to give an accurate prediction of performance metrics. A validated model is the basis of web server capacity planning, where models are used to predict performance in different settings, see Hu et al. [6] or Menasc and Almeida [12].

Today a web site can receive millions of hits per day and it may become overloaded as the arrival rate exceeds the server capacity. To cope with this, overload control can be used, which means that some requests are allowed to be served by the web server and some are rejected. In this way the web server can achieve reasonable service times for the accepted requests. In overload control investigations for web servers, performance models predict improvements when using a certain overload control strategy, see Widell [16] or Cao and Nyberg [3]. Overload control is a research area of its own, but it is depending on performance models that are valid in the overloaded work region.

Several attempts have been made to create performance models for web servers. Van der Mei et al. [11] modeled the web server as a tandem queuing network. The model was used to predict web server performance metrics and was validated through measurements and simulations. Wells et al. [15] made a performance analysis of web servers using colored Petri nets. Their model is divided into three layers, where each layer models a certain aspect of the system. The model has several parameters, some of which are known. Unknown parameters are determined by simulations. Dilley et al. [5] used layered queuing models in their performance studies. Cherkasova and Phaal [4] used a model similar to the one presented in this paper, but with assumptions of deterministic service times and session-based workload. Beckers et al. [2] proposed a generalized processor sharing performance model for Internet access lines which include web servers. Their model describes the flow-level characteristics of the traffic carried. They established simple relations between the capacity, the utilization of the access line and download times of Internet objects.

However, several of the previous models are complicated. It lacks a simple model that is still valid in the overloaded work region. A simple model renders a smaller parameter space thus easier to estimate, while a complicated model usually contains parameters that are difficult to obtain.

A simple model, like the M/M/1/K or M/D/1/K with a First-Come-First-Served (FCFS) service discipline can predict web server performance quite well. But conceptually it is difficult to assume that the service time distribution is exponential or deterministic

and that the service discipline is always FCFS.

In this paper we describe a web server model that consists of a processor sharing node with a queue attached to it. The total number of jobs in the system is limited. The arrival process to the server is assumed to be Poissonian, whereas the service time distribution is arbitrary. A system like this is called an M/G/1/K*PS queue. The average service time and the maximum number of jobs are parameters that can be determined through a maximum likelihood estimation. We also derived closed form expressions for web server performance metrics such as throughput, average response time and blocking probability.

Our validation environment consists of a server and two computers representing clients connected through a switch. The measurements validate the model. Results show that the model can predict the performance metrics at both lighter loaded and overloaded regions.

The rest of the paper is organized as follows: The next section gives an overview of how a web server works and introduces M/G/1/K*PS queue. In section 3 we describe our new web server model and derive expressions for the performance metrics. We develop the maximum likelihood estimation of the model parameters in Section 4. Our model is validated through experiments in Section 5. Section 5 shows the results and the discussion. The last section concludes our work.

2 Preliminaries

This section describes how web servers work and gives a background on the theory of an M/G/1/K*PS queue.

2.1 Web servers

A web server contains software that offers access to documents stored on the server. Clients can browse the documents in a web browser. The documents can be for example static Hypertext Markup Language (HTML) files, image files or various script files, such as Common Gateway Interface (CGI), Java script or Perl files. The communication between clients and server is based on HTTP [13].

A HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed through a three-way handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, a document request can be issued with a HTTP GET message to the server. The server then replies with a HTTP

GET REPLY message. Finally, the TCP connection is closed by sending TCP FIN and TCP ACK messages in both directions.

Apache [Apache], which is a well-known web server and widely used, is multi-threaded. This means that a request is handled by its own thread or process throughout the life cycle of the request. Other types of web servers e.g. event-driven ones also exist [14]. However, in this paper we consider only the Apache web server. Apache also puts a limit on the number of processes allowed at one time in the server.

2.2 M/G/1/K*PS queue

Consider an M/G/1/K queue with processor sharing service discipline. The arrival of jobs is according to a Poisson process with rate λ. The service time requirements have a general distribution with mean $\bar{x}$. An arrival will be blocked if the total number of jobs in the system has reached a predetermined value K. A job in the queue receives a small quantum of service and is then suspended until every other job has received an identical quantum of service in a round-robin fashion. When a job has received the amount of service required, it leaves the queue. Such a system can also be viewed as a queueing network with one node [8].

The probability mass function (pmf) of the total number of jobs in the system has the following expression,

$$P[N = n] = \frac{(1-\rho)\rho^n}{(1-\rho^{K+1})}, \tag{1}$$

where ρ is the offered traffic and is equal to $\lambda\bar{x}$. We note that an M/M/1/K*FCFS queue has the same pmf [9, 10]. However the service time distribution of the M/M/1/K*FCFS queue must be exponential and its service discipline must be FCFS.

3 Web Server Model

We model the web server using an M/G/1/K*PS queue as Fig. 1 shows. The requests arrive according to a Poisson process with rate λ. The average service requirement of each request is $\bar{x}$. The service can handle at most K requests at a time. A request will be blocked if the number has been reached. The probability of blocking is denoted as P_b. Therefore the rate of blocked requests is given by λP_b.

From (1) we can derive the following three performance metrics, average response time, throughput and blocking probability.

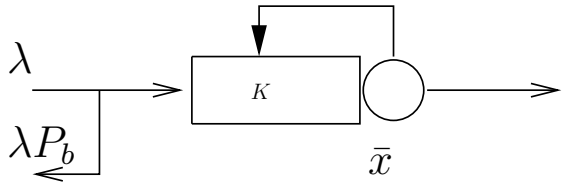


Figure 1: An M/G/1/K-PS model of web servers

The blocking probability P_b is equal to the probability that there are K jobs in the system, i.e. the system is full,

$$P_b = P[N = K] = \frac{(1-\rho)\rho^K}{(1-\rho^{K+1})}. \tag{2}$$

The throughput H is the rate of completed requests. When web server reaches equilibrium, H is equal to the rate of accepted requests,

$$H = \lambda(1 - P_b). \tag{3}$$

The average response time T is the expected sojourn time of a job. Following the Little's law, we have that

$$T = \frac{E[N]}{H} = \frac{\rho^{K+1}(K\rho - K - 1) + \rho}{\lambda(1-\rho^K)(1-\rho)} \tag{4}$$

4 Parameter Estimation

There are two parameters, $\bar{x}$ and K, in our model. We assume that the average response time for a certain arrival rate can be estimated from measurements. The estimations, $\hat{\bar{x}}$ and $\hat{K}$, are obtained by maximizing the likelihood function of the observed average response time.

Let T_i be the average response time predicted from the model and $\hat{T}_i$ be the average response time estimated from the measurements when the arrival intensity is $\lambda_i, i = 1 \ldots m$. Since the estimated response time $\hat{T}$ is the mean of samples, it is approximately a normal distributed random variable with mean T and variance σ_T^2/n when the number of samples n is very large. Hence, the model parameter pair $(\bar{x}, K)$ can be estimated by maximizing

the log-likelihood function

$$\log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma_i^2/n_i}} \exp\left[-\frac{\left(\hat{T}_i - T_i\right)^2}{2\sigma_i^2/n_i} \right]. \tag{5}$$

Maximizing the log-likelihood function above is equivalent to minimize the weighted sum of square errors as follows,

$$\sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{\sigma_i^2/n_i}. \tag{6}$$

As an approximation, the estimated variance of response time, ${\hat{\sigma}_i}^2$, can be used instead of σ_i^2.

Now, the problem of parameter estimation becomes a question of optimization,

$$(\hat{\bar{x}}, \hat{K}) = \underset{(\bar{x},K)}{\arg\ \min} \sum_{i=1}^{m} \frac{\left(\hat{T}_i - T_i\right)^2}{{\hat{\sigma}_i}^2/n_i} \tag{7}$$

The optimization can be solved in various ways, such as steepest decent, conjugate gradient, truncated Newton and even brute force searching. In this paper, we used a brute force approach. The optimum parameter is selected by examining every point of the discretized parameter space.

5 Experiments

Our validation experiments used one server computer and two client computers connected through a 100 Mbits/s Ethernet switch. The server was a PC Pentium III 1700 MHz with 512 MB RAM. The two clients were both PC Pentium III 700 with 256 MB RAM.

All computers used RedHat Linux 7.3 as operating system. Apache 1.3.9 [Apache] was installed in the server. We used the default configuration of the Apache, except for the maximum number of connections. The client computers were installed with a HTTP load generator, which was a modified version of S-Client [1]. The S-Client is able to generate high request rates even with few client computers by aborting TCP connection attempts that take too long time. The original version of S-Client uses deterministic waiting time between requests. We used exponential distributed waiting time instead. This makes the arrival process Poissonian [7].

Table 1: The configuration of four experiments

	$N_{\mathrm{r}} = 1000$	$N_{\mathrm{r}} = 2000$
$N_{\mathrm{conn,max}} = 75$	A1	B1
$N_{\mathrm{conn,max}} = 150$	A2	B2

Table 2: Estimated Parameters of the Model

	A1	A2	B1	B2
$\hat{\tilde{x}}$	0.00708	0.00708	0.00866	0.00834
$\hat{K}$	208	286	215	298

The clients were programmed to request dynamically generated HTML files from the server. The CGI script was written in Perl. It generates a fix number, N_{r}, of random numbers, adds them together and returns the summation. By varying N_{r}, we can simulate different loads on the web server.

We were interested in the following performance metrics: average response time, throughput, and blocking probability. The throughput was estimated by taking the ratio between the total number of successful replies and the time span of measurement. The response time is the time difference between when a request is sent and when a successful reply is fully received. The average response time was calculated as the sample mean of the response times after removing transients. An HTTP request sent by a client computer will be blocked either when the maximum number of connections, denoted as $N_{\mathrm{conn,max}}$, in the server has been reached or the TCP connection is timed out at the client computer. A TCP connection will be timed out by a client computer when it takes too long time for the server to return an ACK of the TCP-SYN. The blocking probability was then estimated as the ratio between the number of blocking events and the number of connection attempts in a measurement period.

We carried out the experiments in four cases by varying N_{r} and $N_{\mathrm{conn,max}}$. Table 1 shows the configurations of four experiments: A1, A2, B1 and B2. In each case, the performance metrics were collected while the arrival rate (in number of requests/second) was changed from 20 to 300 with step size 20.

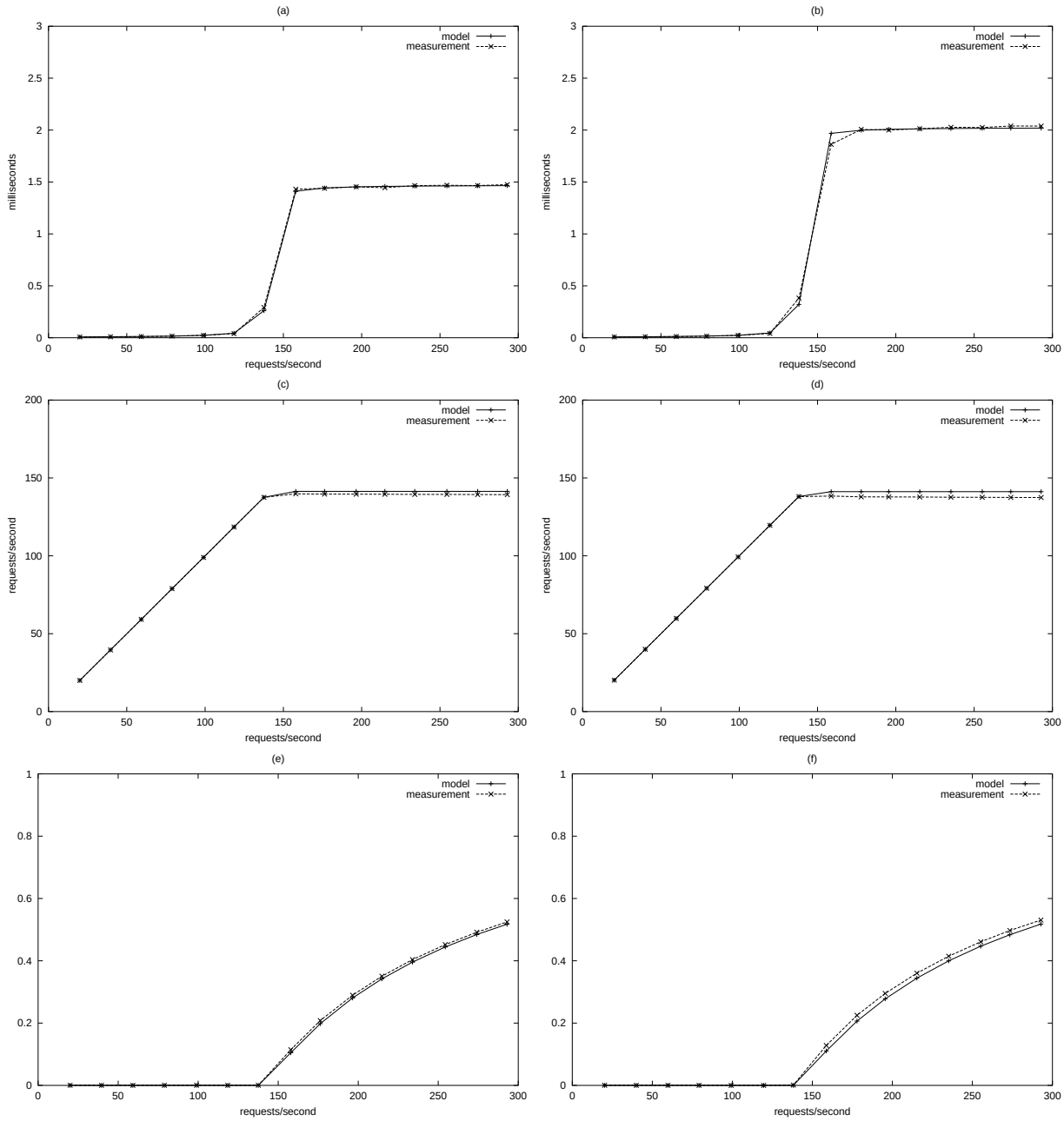


Figure 2: (a) Average response time of A1. (b) Average response time of A2. (c) Throughput of A1. (d) Throughput of A2. (e) Blocking probability of A1. (f) Blocking probability of A2.

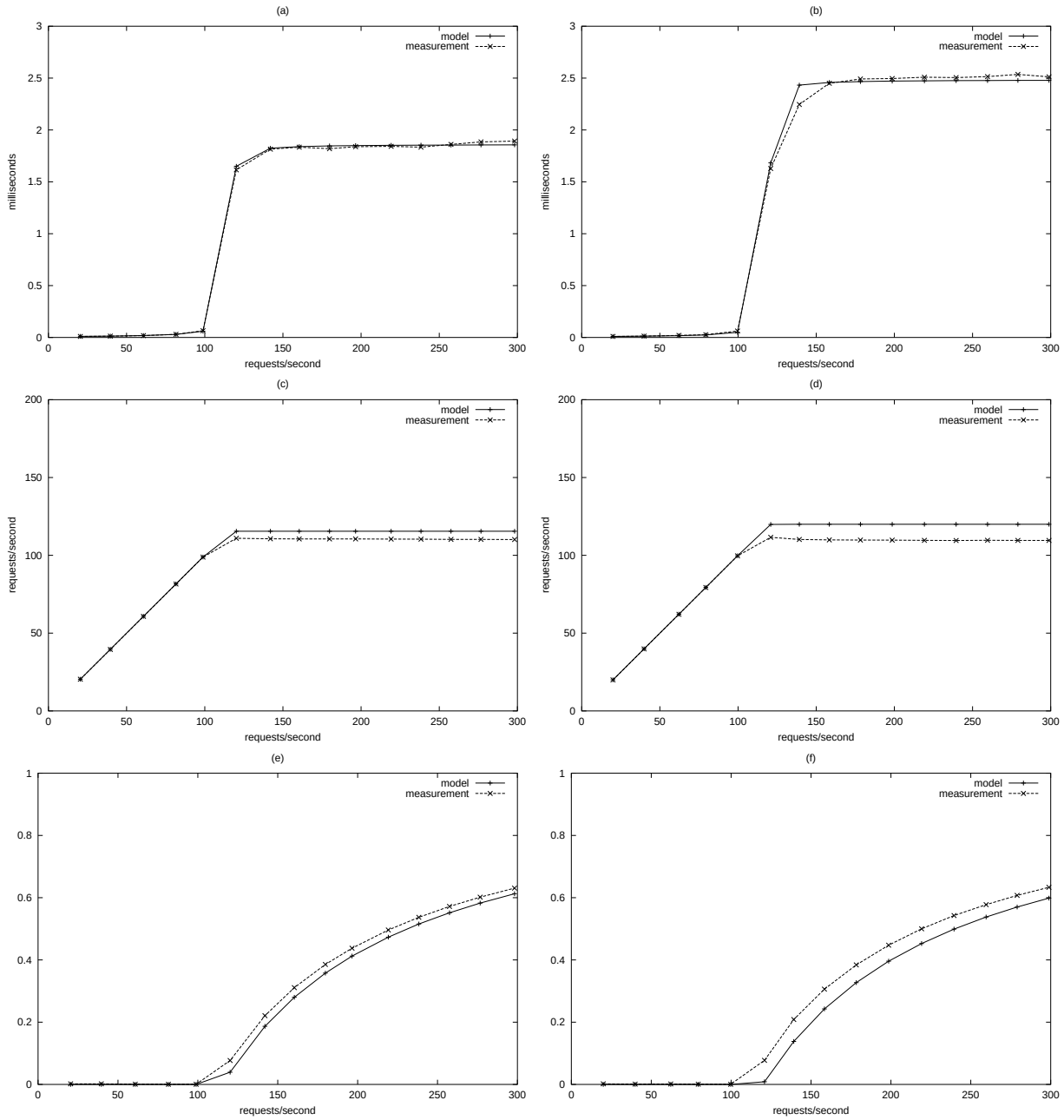


Figure 3: (a) Average response time of B1. (b) Average response time of B2. (c) Throughput of B1. (d) Throughput of B2. (e) Blocking probability of B1. (f) Blocking probability of B2.

6 Results and Discussion

The method developed in section 4 were used to estimate the parameters from the measurements. The results are presented in Table 2.

Using the estimated parameters, we can predict the web server performance and compare it with the measurements. Fig. 2 and 3 show the average response time, the throughput and the blocking probability curves. To facilitate the discussion, we divide four experiments into two groups. The first group called α contains experiments A1 and A2 and the second group β contains B1 and B2.

We notice the following relations in Table 2

$$\hat{\bar{x}}_{\mathrm{A1}} = \hat{\bar{x}}_{\mathrm{A2}} < \hat{\bar{x}}_{\mathrm{B1}} \approx \hat{\bar{x}}_{\mathrm{B2}}.$$

Recall that the same CGI script is used for experiments in the same group. The script for group β is more computational intensive than the one for group α. The script for the group α adds 1000 numbers but the script for the other adds 2000 numbers. However $\bar{x}_{\mathrm{B1}}$(or $\bar{x}_{\mathrm{B2}}$) is not twice as large as $\bar{x}_{\mathrm{A1}}$(or $\bar{x}_{\mathrm{A2}}$). This can be understood as that the time spent on the summations is only a fraction of the sojourn time. Other parts of $\bar{x}$ include the connection setup time, the file transferring time, etc., which can be considered as constants in all experiments.

We find that the estimated K in all experiments is much greater than $N_{\mathrm{conn,max}}$ which is a parameter in the Apache configuration. One may expect that $K \approx N_{\mathrm{conn,max}}$. However, recognize that in our model K is the maximum number of jobs simultaneously in the system. The jobs can be in the HTTP processing phase as well as in the TCP connection setup phase in which the Apache has no control. On the other hand, $N_{\mathrm{conn,max}}$ is the maximum number of jobs handled by the Apache which runs on top of the TCP layer. Therefore K should be greater than $N_{\mathrm{conn,max}}$.

One can reasonably predict that within the same experiment group, α or β, the difference of $\hat{K}$ should be approximately equal to the difference of $N_{\mathrm{conn,max}}$ which is 75. In our experiments, $\hat{K}_{\mathrm{A2}} - \hat{K}_{\mathrm{A1}} = 78$, $\hat{K}_{\mathrm{B2}} - \hat{K}_{\mathrm{B1}} = 83$. There is a reason why the differences are close and greater than 75. When $N_{\mathrm{conn,max}}$ is increased, the average load of CPU will increase. As a result, the TCP listening queue will be visited less frequently by the operating system. This implies that the TCP listening queue size will increase. So the increase of K will be greater than the increase of $N_{\mathrm{conn,max}}$. This explanation is also supported by the fact that the increase of K in the experiment group β is larger than in the group α.

As we mentioned earlier, the CGI script of the group β is more CPU demanding than that of the group α.

Now we turn our attention from the estimated parameters to the predicted performance metrics. The measured and the predicted average response time in all four experiments fit well. This should be of a little surprise because the measured average response times at various arrival rates are used to estimate the parameters of the model.

The predicted blocking probability is slightly less than the measurements in all four experiments. According to (3), the error in the prediction of P_b will also affect the prediction of the throughput. Such divergence is expected since we only use the measured average response time in our parameter estimation.

7 Conclusions

We have presented an M/G/1/K*PS queueing model of a web server. We obtained closed form expressions for web server performance metrics such as average response time, throughput and blocking probability. Model parameters were estimated from the measured average response time. We validated the model through four sets of experiments. The performance metrics predicted by the model fitted well to the experimental outcome.

Future work will include more validation under different types of loads such as network intensive and hard-disk intensive cases. It would also be interesting to see how well the model fits web servers that use an event-driven approach instead of multi-threading.

Acknowledgments

We would like to thank Thiemo Voigt for sharing his code with us and Niklas Widell for useful and interesting discussions. The work has been supported by the Swedish Research Council under contract No. 621-2001-3053.

References

[Apache] Apache. Apache web server. `http://www.apache.org`.

[1] Banga, G. and Druschel, P. (1997). Measuring the capacity of a web server. In *USENIX Symposium on Internet Technologies and Systems*, pages 61–71.

[2] Beckers, J., I.Hendrawan, R.E.Kooij, and van der Mei, R. (2001). Generalized processor sharing performance model for internet access lines. In *9th IFIP Conference on Performance Modelling and Evaluation of ATM and IP Networks*. Budapest.

[3] Cao, J. and Nyberg, C. (2002). On overload control through queue length for web servers. In *16th Nordic Teletraffic Seminar*. Espoo, Finland.

[4] Cherkasova, L. and Phaal, P. (2002). Session-based admission control: A mechanism for peak load management of commercial web sites. *IEEE Transactions on computers*, 51(6):669–685.

[5] Dilley, J., Friedrich, R., Jin, T., and Rolia, J. (1998). Web server performance measurement and modeling techniques. *Performance Evaluation*, 33:5–26.

[6] Hu, J., Mungee, S., and Schmidt, D. (1998). Principles for developing and measuring high-performance web servers over ATM. In *Proceedings of INFOCOM '98, March/April 1998*.

[7] Jain, R. (1991). *The Art of Computer Systems Performance Analysis*. John Wiley & Sons.

[8] King, P. J. B. (1990). *Computer and Communication Systems Performance Modelling*. Prentice Hall.

[9] Kleinrock, L. (1975). *Queueing Systems, Volume 1: Theory*. John Wiley & Sons.

[10] Lam, S. (1977). Queueing networks with population size constraints. *IBM Journal of Research and Development*, 21(4):370–378.

[11] Mei, R. D. V. D., Hariharan, R., and Reeser, P. K. (2001). Web server performance modeling. *Telecommunication Systems*, 16(3,4):361–378.

[12] Menasc, D. A. and Almeida, V. A. F. (2002). *Capacity Planning for Web Services*. Prentice Hall.

[13] Stallings, W. (2000). *Data & Computer Communications*. Prentice Hall. Sixth Edition.

[14] Voigt, T. (2002). Overload behaviour and protection of event-driven web servers. In *In proceedings of the International Workshop on Web Engineering*. Pisa, Italy.

[15] Wells, L., Christensen, S., Kristensen, L. M., and Mortensen, K. H. (2001). Simulation based performance analysis of web servers. In *Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM 2001)*, pages 59–68. IEEE Computer Society.

[16] Widell, N. (2002). Performance of distributed information systems. Technical Report 144, Department of Communication Systems, Lund Institute of Technology. Lic. Thesis.

An Approximate Analysis of Load Balancing Using Stale State Information for Servers in Parallel

Jianhua Cao and Christian Nyberg

Presented at the 2:nd IASTED International Conference on Communications Internet and Information Technology, November 2003, Scottsdale USA

Abstract. That a load balancing strategy using stale information carelessly will incur system performance degradation is easy to verify. However it is not so obvious that routing a customer to the expected shortest queue has the same problem when information for used decision is stale. We consider a queueing system with a load balancer and a pool of identical FCFS queues in parallel. The arrival process is assumed to be Poisson and the service times have identical independent exponential distributions. The pool of servers informs the load balancer the number of customers in each server at some regularly spaced time instances. The load balancer routes each customer to the expected shortest queue based on available stale information and elapsed time since the last time instance of system state information updating. The system performance analysis of this type of model is usually difficult because the involved state space is very large. However when taking the number of servers to the infinite limit, we have a set of differential equations which is easier to handle than the finite case. Using the approximation of infinite number of severs, we show that the average waiting time for the system is not always minimized by routing each customer to the expected shortest queue when information used for decision is stale.

1 Introduction

Load Balancing (LB) improves network performance by distributing traffic efficiently so that individual servers are not overwhelmed by sudden fluctuations in activity [3, 13]. In a large Internet web site, balancing incoming requests evenly among many computers is a critical and sometimes tricky task [16, 17]. Despite fruitful theoretical results on both static and dynamic LB strategies[19, 18], current engineering practices, such as web server farm/cluster architecting, is calling for robust LB algorithms that can exploit imperfect system state information effectively.

Routing a customer to the shortest queue when some mild constraint on the service time distribution is satisfied is a conventional wisdom in LB for identical servers in parallel. However sticking to the rule when the assumption of perfect information is broken will cause performance troubles as a previous study reveals [14]. In this paper, a seemly "good" LB algorithm which routes customers to the expected shortest queue is studied. The queueing system under consideration consists a pool of identical servers in parallel, each with its own queue. Customers arrive according to a Poisson process and immediately upon arrival must join one of the queues thereafter to be served on a first-come first-served basis, with no jockeying or defection allowed. The service times have identical independent exponential distributions and are independent of the arrival process and the decisions of customers. The pool of servers announces the state of the system, i.e. the number of customers waiting and being served at each server, at some regularly spaced time instance called state information update instance. When a customer arrives, the load balancer of the system knows the elapsed time since the last announcement plus the stale state information used for decision. The load balancer routes each customer to the expected shortest queue based on available stale information, which is called the expected shortest queue strategy (ESQS) hereafter. Other well known strategies include random selection and round robin.

The problem of deciding which queue to join when the full and exact state information is available has been studied by many authors. Haight [8] and Kingman [12] considered the dynamics of two servers in parallel when arrivals join the shortest queue. Winston [22] and Weber [20] showed that the shortest queue strategy for N servers in parallel is optimal when the service time distribution has a non-decreasing failure rate and arbitrary arrival. In practice, however, the full and exact state information may be difficult to obtain and maintain, see for example Eager [7] and Zhou [23]. Hjalmtysson and Whitt [11] studied the resource sharing of parallel queues by periodically redistribute customers based on fresh state information. Mitzenmacher [14] considered a system with period state information

update similar to ours. He showed that joining the shortest queue based on stale state information is better than the strategy of random selection when the state information is not too old. Motivated by Mitzenmacher's work, Dahlin [5] proposed an algorithm to exploit the stale state information. His simulation results show that the system employing the algorithm gives shorter waiting times than joining queues at random. Even when the state information is very old, his algorithm will not perform worse than the random strategy.

In the quest for the optimal LB strategy with stale state information, one may first consider the optimality of the strategy that minimizes each customer's expected waiting time. When the distribution of service times is exponential, routing a customer to the expected shortest queue will minimize this customer's expected waiting time. However finding the expected shortest queue is usually very computationally demanding as the available state information is stale. Even if a load balancer has access to unlimited computational resources, the overall system performance is still questionable as the following cases suggest. The fact that decentralized routing based on the expected shortest delay delay may result in poor performance has been known for a long time, see for example Cohen and Kelley [4] and Bersekas [2]. There are several other queueing scenarios where individual optimality does not give a system optimum. Bell and Stidham [1] considered the case that customers only know the service time distribution and the cost of waiting in each server upon arrival to N parallel servers. Whitt [21] found that the average waiting time of the system is not minimized by having each customer minimize his expected waiting time upon arrival to N parallel servers with a certain service time distribution. In this paper numerical examples show that the average queue length and average waiting time of the system is not minimized by routing each customer to the expected shortest queue when the available system state information for load balancing is stale. So this is yet another queueing scenario in which the individual optimum does not coincide with the system optimum.

The rest of this paper is organized as follows. In Section 2 we derive differential equations for the system dynamics. Both the case of finite and infinite number of servers are treated. Section 3 contains the numerical results for the case of infinite number of servers and discussions. Discussions of our result from game theoretical perspective are offered in Section 4. We summarize the paper in Section 5.

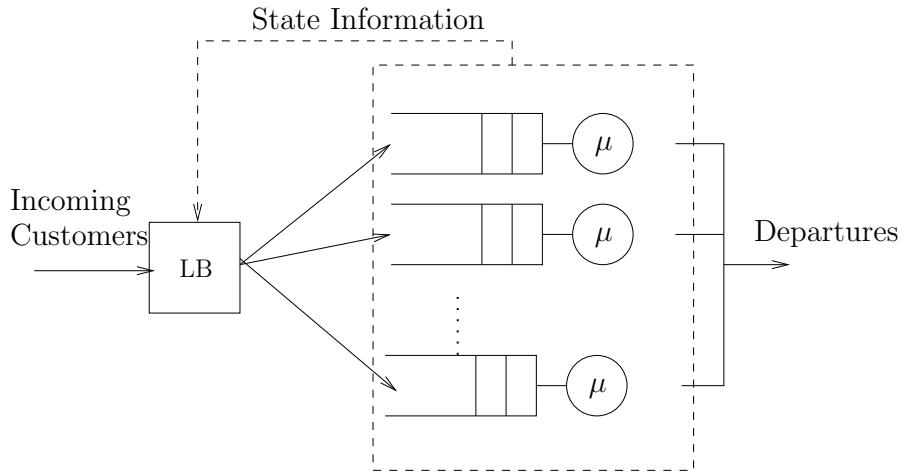


Figure 1: A system consisting of a LB and N identical servers in parallel.

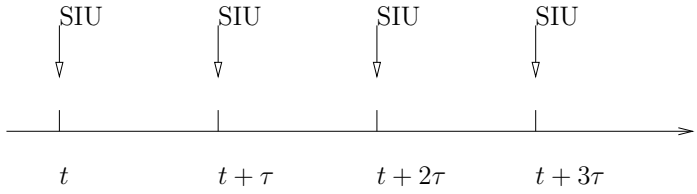


Figure 2: The servers inform the LB their states at regularly spaced SIU instances: t, $t+\tau$, $t+2\tau$, $t+3\tau$, ...

2 Dynamics of the System

2.1 System model

We consider a system consisting N FCFS servers in parallel and a load balancer that dispatches incoming customers to the servers, as Fig. 1 shows. The service rates are assumed to be identical and exponential distributed. The servers inform the LB their state, i.e. the number of customers in each server, at some regularly spaced time instance which is called state information update (SIU) instance subsequently, as Fig. 2 shows. The time between two SIU instances is denoted as τ and assumed to be a constant.

We first consider when there are a finite number of servers in the pool. Because the system of differential equations is difficult to solve analytically or numerically especially when the number of server is large, we then consider the limit situation in which the number of servers is infinite. Even though the analytical solution is still difficult to obtain,

the numerical result reveals some important proprieties of the system dynamics. The infinite server system approximation seems strange at first since an arrival is guaranteed to find an empty server when the state information is fresh. When the state information is stale, an arrival may join the queue of a busy server due to errors in the predication regardless the number of servers.

2.2 Finite number of servers

Let N be the total number of servers in the server pool; $\mathbf{k} = [k_1, k_2, \ldots, k_N]$ be the vector of the number of customers in each server; $\hat{\mathbf{k}} = [\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_N]$ be the vector of the predicted number of customers in each server. For convenience, let δ_i denote $[\underbrace{0, \ldots, 0, 1}_{i}, \underbrace{0, \ldots, 0}_{N-i}]$. The arrival process is assumed to be Poisson. Let λ be the arrival rate per server to the system. The total arrival rate to the system is $N\lambda$.

Since all customers make decisions using prediction, the arrival rate to a server depends not only on how many customers are predicted in this server but also how many customers are predicted in other servers. When the predicted number of customers is distributed as $\hat{\mathbf{k}}$, the arrival rate to the server i at time t is given by,

$$\phi^{(i)}_{\hat{\mathbf{k}}}(t) = \begin{cases} N\lambda/n & \text{if } i \in A \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where $A = \{i : k_i = \min(\hat{k}_1, \hat{k}_2, \ldots, \hat{k}_N)\}$ is the set of servers having less customers than the others in the pool and n is the number of elements in the set A. When a few servers are being predicted to have the same number of customers and to have less customers than any other in the pool, the total arrival rate $N\lambda$ is divided into n portions. Each server that is predicted to have the smallest number of customers receives a portion of $N\lambda/n$.

The state of the system is characterized by $\mathbf{k}$ and $\hat{\mathbf{k}}$ jointly because $\mathbf{k}$ and $\hat{\mathbf{k}}$ may differ except at the moment when SIU happens. Let $p_{\mathbf{k},\hat{\mathbf{k}}}(t)$ be the probability that the system is at the state $(\mathbf{k}, \hat{\mathbf{k}})$. When the system is initially empty,

$$p_{\mathbf{k},\hat{\mathbf{k}}}(0) = \begin{cases} 1 & \text{if } \mathbf{k} = \hat{\mathbf{k}} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

For brevity, we only consider states that are not near the boundary, i.e. when $k_i \geq 1$

and $\hat{k}_i \geq 1$, $k = 1, \ldots, N$ in the following discussion.

Conditioned on the current state $(\mathbf{k}, \hat{\mathbf{k}})$, the probability that the state is unchanged in a small time interval Δt is

$$1 - 2\sum_{i=1}^{N} \phi_{\hat{\mathbf{k}}}^{(i)}(t)\Delta t - 2N\mu\Delta t + o(\Delta t).$$

Conditioned on the current state $(\mathbf{k} - \delta_i, \hat{\mathbf{k}})$, the probability that the state changes to $(\mathbf{k}, \hat{\mathbf{k}})$ in a small time interval Δt is

$$\phi_{\hat{\mathbf{k}}}^{(i)}(t)\Delta t + o(\Delta t).$$

Conditioned on the current state $(\mathbf{k}, \hat{\mathbf{k}} - \delta_i)$, the probability that the state changes to $(\mathbf{k}, \hat{\mathbf{k}})$ in a small time interval Δt is

$$\phi_{\hat{\mathbf{k}}-\delta_i}^{(i)}(t)\Delta t + o(\Delta t).$$

Conditioned on the current state $(\mathbf{k} + \delta_i, \hat{\mathbf{k}})$ or $(\mathbf{k}, \hat{\mathbf{k}} + \delta_i)$, the probability that the state changes to $(\mathbf{k}, \hat{\mathbf{k}})$ in a small time interval Δt is

$$\mu\Delta t + o(\Delta t).$$

Conditioned on all other current state, the probability that the state changes to $(\mathbf{k}, \hat{\mathbf{k}})$ in a small time interval Δt is $o(\Delta t)$.

Thus the system evolves from time t to $t + \Delta t$ according to the following rule,

$$\begin{aligned} p_{\mathbf{k},\hat{\mathbf{k}}}(t + \Delta t) =& p_{\mathbf{k},\hat{\mathbf{k}}}(t)\left(1 - 2\sum_{i=1}^{N} \phi_{\hat{\mathbf{k}}}^{(i)}(t)\Delta t - 2N\mu\Delta t\right) \\ &+ \sum_{i=1}^{N} p_{\mathbf{k}-\delta_i,\hat{\mathbf{k}}}(t)\phi_{\hat{\mathbf{k}}}^{(i)}(t)\Delta t \\ &+ \sum_{i=1}^{N} p_{\mathbf{k},\hat{\mathbf{k}}-\delta_i}(t)\phi_{\hat{\mathbf{k}}-\delta_i}^{(i)}(t)\Delta t \\ &+ \sum_{i=1}^{N} p_{\mathbf{k}+\delta_i,\hat{\mathbf{k}}}(t)\mu\Delta t \\ &+ \sum_{i=1}^{N} p_{\mathbf{k},\hat{\mathbf{k}}+\delta_i}(t)\mu\Delta t + o(\Delta t). \end{aligned} \tag{3}$$

Moving $p_{\mathbf{k},\hat{\mathbf{k}}}(t)$ from the right side to the left, dividing both sides by Δt and letting

$\Delta t \to 0$, we have the following differential equation,

$$\begin{aligned}\frac{d}{dt}p_{\mathbf{k},\hat{\mathbf{k}}}(t) = &-2p_{\mathbf{k},\hat{\mathbf{k}}}(t)\left(\sum_{i=1}^{N}\phi^{(i)}_{\hat{\mathbf{k}}}(t) + N\mu\right)\\ &+\sum_{i=1}^{N}p_{\mathbf{k}-\delta_i,\hat{\mathbf{k}}}(t)\phi^{(i)}_{\hat{\mathbf{k}}}(t)\\ &+\sum_{i=1}^{N}p_{\mathbf{k},\hat{\mathbf{k}}-\delta_i}(t)\phi^{(i)}_{\hat{\mathbf{k}}-\delta_i}(t)\\ &+\sum_{i=1}^{N}p_{\mathbf{k}+\delta_i,\hat{\mathbf{k}}}(t)\mu + \sum_{i=1}^{N}p_{\mathbf{k},\hat{\mathbf{k}}+\delta_i}(t)\mu. \end{aligned} \tag{4}$$

At the SIU instance t, an arrival will have the full and the exact information about how many customers are in each queue, therefore,

$$p_{\mathbf{k},\hat{\mathbf{k}}}(t) = \begin{cases}\lim_{s\uparrow t}\sum_{\mathbf{h}}p_{\mathbf{k},\mathbf{h}}(s) & \text{if } \hat{\mathbf{k}} = \mathbf{k}\\ 0 & \text{otherwise}\end{cases} \tag{5}$$

The system of differential equation is nonlinear because $p_{\mathbf{k},\hat{\mathbf{k}}}$ depends on the state dependent arrival rate. Like most nonlinear differential equations, a closed analytical solution is unlikely to be found. Let us investigate the prospect of a numerical solution. When the number of servers in the pool is large, we have a very large state space. Assume that the system of differential equations is truncated so the maximal number of customers per server is $k_{\max}$. When we have N servers, the total number of states is $(k_{\max}+1)^{2N}$. The number of states increases exponentially as N increases. Therefore (4) is numerically intractable when N is large.

However when we let $N \to \infty$,we can actually have a set of differential equations that are simpler but yet powerful enough to reveal the dynamics of the system.

2.3 Approximated analysis using infinite number of servers

Let $p_{i,j}(t)$ be the percentage of servers having i customers but being predicted to have j customers at time t. Though we do not explicitly model how many customers, real and predicted, in each queue, knowing $p_{i,j}(t)$ is enough to derive the average number of customers per server in the system.

Let $\psi_j(t)$ be the arrival rate to servers that are predicted to have j customers each.

Since new arrivals are routed to servers that are predicted having 0 customers, $\psi_0(t)$ is the ratio between the average arrival rate per server λ and the percentage of servers that are predicted to have 0 customers. Thus,

$$\psi_j(t) = \begin{cases} \lambda / \sum_k p_{k,0}(t) & \text{if } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

When the system is initially empty,

$$p_{i,j}(0) = \begin{cases} 1 & \text{if } i = 0 \text{ and } j = 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The system evolves from time t to time $t + \Delta t$ according to the following rules: when $i = 0$ and $j = 0$,

$$\begin{aligned} p_{0,0}(t + \Delta t) =& p_{0,0}(t)\,(1 - 2\psi_0(t)\Delta t) \\ &+ p_{1,0}(t)\mu\Delta t + p_{0,1}(t)\mu\Delta t + o(\Delta t); \end{aligned} \tag{8}$$

when $i = 0$ and $j \geq 1$,

$$\begin{aligned} p_{0,j}(t + \Delta t) =& p_{0,j}(t)\,(1 - 2\psi_j(t)\Delta t - \mu\Delta t) \\ &+ p_{0,j-1}(t)\psi_{j-1}(t)\Delta t \\ &+ p_{1,j}(t)\mu\Delta t + p_{0,j+1}(t)\mu\Delta t + o(\Delta t); \end{aligned} \tag{9}$$

when $i \geq 1$ and $j = 0$,

$$\begin{aligned} p_{i,0}(t + \Delta t) =& p_{i,0}(t)\,(1 - 2\psi_0(t)\Delta t - \mu\Delta t) \\ &+ p_{i-1,0}(t)\psi_0(t)\Delta t \\ &+ p_{i+1,0}(t)\mu\Delta t + p_{i,1}(t)\mu\Delta t + o(\Delta t); \end{aligned} \tag{10}$$

when $i \geq 1$ and $j \geq 1$,

$$\begin{aligned} p_{i,j}(t + \Delta t) =& p_{i,j}(t)\,(1 - 2\psi_j(t)\Delta t - 2\mu\Delta t) \\ &+ p_{i-1,j}(t)\psi_j(t)\Delta t + p_{i,j-1}(t)\psi_{j-1}(t)\Delta t \\ &+ p_{i+1,j}(t)\mu\Delta t + p_{i,j+1}(t)\mu\Delta t + o(\Delta t). \end{aligned} \tag{11}$$

Moving $p_{0,0}(t)$, $p_{0,j}(t)$, $p_{i,0}(t)$ and $p_{i,j}(t)$ from the right sides of (8),(9), (10) and (11) to the left sides, dividing both sides by Δt and letting $\Delta t \to 0$, we have the following system of differential equations, when $i = 0$ and $j = 0$,

$$\frac{d}{dt}p_{0,0}(t) = -2p_{0,0}(t)\psi_0(t) + p_{1,0}(t)\mu + p_{0,1}(t)\mu; \tag{12}$$

when $i = 0$ and $j = 1$,

$$\begin{aligned}\frac{d}{dt}p_{0,j}(t) = &- p_{0,j}(t)(2\psi_j(t) + \mu) \\ &+ p_{0,j-1}(t)\psi_{j-1}(t) + p_{1,j}(t)\mu + p_{0,j+1}(t)\mu;\end{aligned} \tag{13}$$

when $i = 1$ and $j = 0$,

$$\begin{aligned}\frac{d}{dt}p_{i,0}(t) = &- p_{i,0}(t)(2\psi_0(t) + \mu) \\ &+ p_{i-1,0}(t)\psi_0(t) + p_{i+1,0}(t)\mu + p_{i,1}(t)\mu;\end{aligned} \tag{14}$$

when $i \geq 1$ and $j \geq 1$,

$$\begin{aligned}\frac{d}{dt}p_{i,j}(t) = &- 2p_{i,j}(t)(\psi_j(t) + \mu) \\ &+ p_{i-1,j}(t)\psi_j(t) + p_{i,j-1}(t)\psi_{j-1}(t) \\ &+ p_{i+1,j}(t)\mu + p_{i,j+1}(t)\mu.\end{aligned} \tag{15}$$

At the SIU instance t, an arrival knows the percentage of servers having i customers in the system. So,

$$p_{i,j}(t) = \begin{cases} \lim_{s\uparrow t} \sum_k p_{i,k}(s) & \text{if } j = i \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

The system of differential equations for the case of an infinite number of servers is still nonlinear and analytical intractable. But now it is solvable by numerical methods.

3 Numerical Results

We first truncate the system of differential equations (12), (13), (14) and (15) to a finite number of i:s and j:s, i.e. $0 \leq i \leq i_{\max} = 100$ and $0 \leq j \leq j_{\max} = 100$. The truncated

differential equations are then solved numerically using ODEPACK [10]. For convenience, we assume the service rate of each server $\mu = 1$ in the following discussion. For stability reasons, both for the system itself and the numerical solutions, we choose to show the result when $\lambda = 0.5$.

We are interested in the average number of customers per server $\bar{n}(t)$ which can be calculated as follows once the distribution of the system states $[p_{i,j}(t)]$ is known,

$$\bar{n}(t) = \sum_{i=0}^{i_{\max}} i \sum_{j=0}^{j_{\max}} p_{i,j}(t) \tag{17}$$

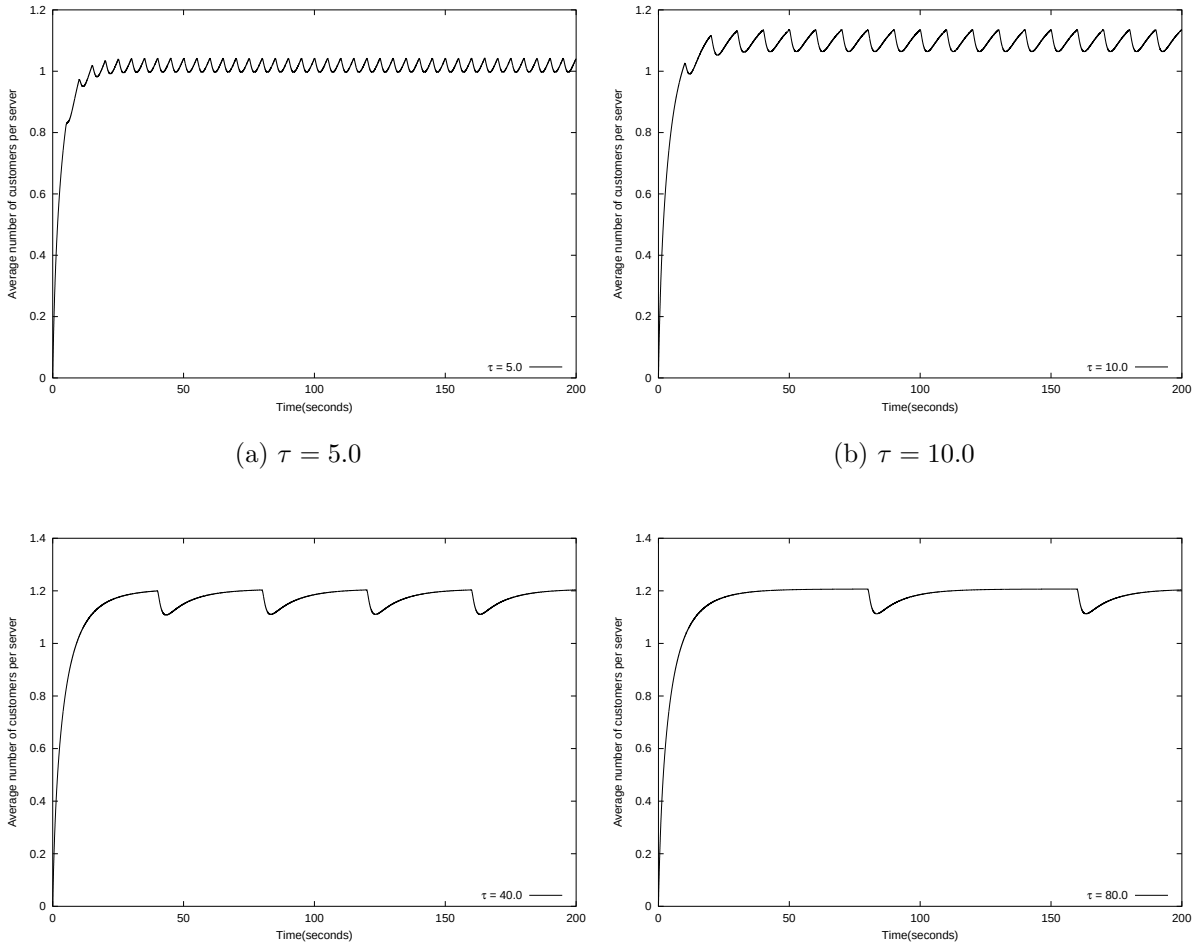
Fig. 3 shows the transient behavior of $\bar{n}(t)$ for different SIU intervals τ when the system is initially empty. From Fig. 3, we see that $\bar{n}(t)$ oscillates. The period of oscillation is equal to the period of the SIU updates. The amplitude of oscillation increases as SIU interval increases.

Fig. 4 shows the transient behavior of $\bar{n}(t)$ between two SIU updates for different SIU intervals τ when the influence of the initial state can be neglected. We scale different SIU intervals to 1 in order to reveal the trend in the curves. From Fig. 4 we see that just before each SIU update instance $\bar{n}(t)$ reaches maximum, and $\bar{n}(t)$ starts to drop right after each SIU update instance. Fig. 4 also shows that for the fixed arrival rate $\lambda = 0.5$, the maximum of $\bar{n}(t)$ increases, but not indefinitely, as the SIU update interval increases.

When the SIU interval is large, the average number of customers in the system is not minimized by routing each customer to the expected shortest queue. In Fig. 4(b) the minimum of the average queue length per server $\bar{n}(t)$ between two SIU instances, which is about 1.11 , is greater than 1.0 which can be obtained if random selection strategy is used. Hence, by Little's law, the average waiting time for the system is not minimized by routing each customer to the expected shortest queue.

Fig. 5 shows how the maximum of $\bar{n}(t)$ increases with arrival rate λ when the SIU interval τ is fixed to 50 seconds. In the same figure, we also plot the average number of customers per server in steady state for the random selection strategy [1]. For the ESQS, the maximum of $\bar{n}(t)$ increases faster and is always greater than the average number of customers per server for the random selection. Therefore it suggests that the system using the ESQS becomes unstable earlier than the system using the random strategy when the arrival rate per server λ is close to 1 or the utilization of servers is close to 1.

[1] When the random selection is used, the arrival process to each server is Poisson. Therefore the average number of customers per server $\bar{n} = \lambda/(1-\lambda)$, where λ is the arrival rate per server.



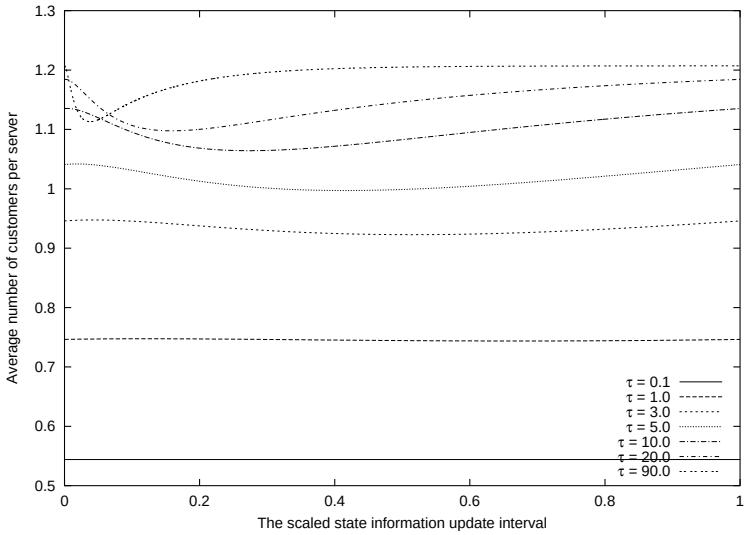
(a) $\tau = 5.0$

(b) $\tau = 10.0$

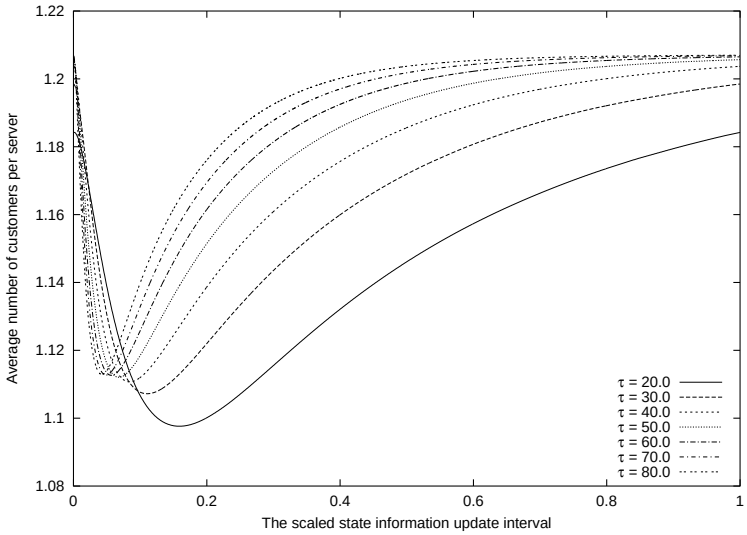
(c) $\tau = 40.0$

(d) $\tau = 90.0$

Figure 3: Transient behavior of the average number of customers per server between time 0 second and 200 seconds for different SIU intervals τ when the system is initially empty.



(a)



(b)

Figure 4: Transient behavior of the average number of customers per server between two SIU instance for different SIU intervals τ when the influence of the initial state can be neglected. We scale different SIU intervals to 1 in order to reveal the trend in curves.

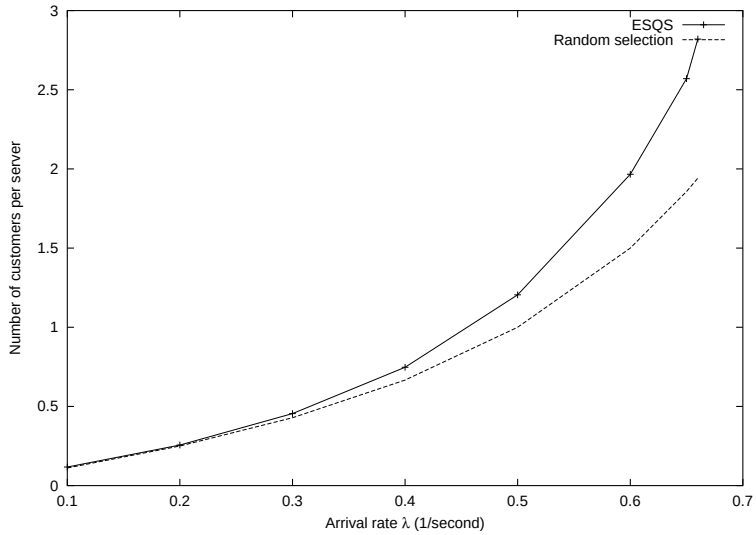


Figure 5: The maximum of the average number of customers per server for the ESQS vs. arrival rate per server when the influence of initial state can be neglected. We also plot the average number of customers per server in steady state when the random selection strategy is used.

4 Discussions

The system under consideration can also be viewed from another perspective. Let us call the system described in Section 2.1 system A. Consider a similar system B. In system B, the load balancer is removed but each arrival decides which queue to join based on stale state information and the elapsed time since last SIU update instance. Now we have a queueing game that will be played by all customers. Every customer tries to minimize his own waiting time. Clearly the optimal decision of each customer also depends on all early arrivals. When almost everyone joins the expected shortest queue based on stale information, no one will benefit a shorter waiting time by deviating the common strategy. In game theory literatures [9, 6, 15], such a situation is refereed as Wardrop equilibrium (for infinite numer of players) or Nash equilibrium (for finite number of players). By definition, system B in Wardrop equilibrium is equivalent to System A.

From game theory point of view, that minimizing the expected waiting time of each customer is not equivalent to minimizing the average waiting time of the system is clear. However when there are finite numer of servers in our system, this fact is difficult to verify through equations that governs the system dynamics. The approximation using infinite number of servers turns out be effective to reveal that in our system Wardop equilibrium

does not bring the system optimal solution.

5 Conclusions

This article studies a LB strategy of routing an arrival to the expected shortest queue among a pool of identical servers in parallel when the available state information is stale. When the number of servers is finite, we derive differential equations that reflect the system dynamics. Because of the analytical and numerical intractability of these differential equations, we consider the limiting situation in which the number of servers is infinite. It turns out that the set of differential equations for an infinite number of servers can be solved numerically. The numerical solution presents some expected behaviors of the system dynamics such as the oscillation and some unexpected behaviors such as the extreme of the average number of customers per server increases as the SIU interval grows. The numerical solution also shows that LB through routing each customer to the expected shortest queue is not always a good strategy in order to achieve the minimum of average waiting time when the available information used for decision is stale. However for the system under consideration the optimum load balancing strategy that minimizes the average waiting time is still unresolved and is a subject of future work.

References

[1] Bell, C. E. and Jr., S. S. (1983). Individual versus social optimization in the allocation of customers to alternative servers. *Management Science*, 29(7):831–839.

[2] Bertsekas, D. P. (1982). Optimal routing and flow control methods for communication networks. In Bensoussan, A. and Lions, J. L., editors, *Analysis and Optimization of Systems*, volume 44 of *Lecture Notes in Control and Information Science*, pages 615–643. Springer-Verlag.

[3] Bourke, T. (2001). *Server Load Balancing*. O'Reilly & Associates. 2nd edition.

[4] Cohen, J. E. and Kelly, F. P. (1990). A paradox of congestion in a queuing network. *Journal of Applied Proabbility*, 27:730–734.

[5] Dahlin, M. (2001). Interpreting stale load information. *IEEE Transactions on Parallel and Distributed System*, 11(10):1033–1047.

[6] Dubey, P. (1986). Inefficiency of Nash equilibria. *Mathematics of Operation Research*, 11:1–8.

[7] Eager, D. L., Lazowska, E. D., and Zahorjan, J. (1986). Adaptive load sharing in homogeneous distributed systems. *IEEE Transactions on Software Engineering*, SE-12(5):662–675.

[8] Haight, F. A. (1958). Two queues in parallel. *Biometrika*, 45(3/4):401–410.

[9] Hassin, R. and Haviv, M. (2003). *To Queue or Not to Queue: Equilibrium Behavior in Queueing Systems*. Kluwer Academic Publishers.

[10] Hindmarsh, A. C. (1983). *Scientific Computing*, chapter ODEPACK, A Systematized Collection of ODE Solvers. North-Holland, Amsterdam.

[11] Hjalmtysson, G. and Whitt, W. (1998). Periodic load balancing. *Queueing Systems*, 30:203–250.

[12] Kingman, J. F. C. (1961). Two similar queues in parallel. *Annals of Mathematical Statistics*, 32(4):1314–1323.

[13] Kopparapu, C. (2002). *Load Balancing Servers, Firewalls, * Caches*. John Wiley & Sons.

[14] Mitzenmacher, M. (2000). How useful is old information? *IEEE Transactions On Parallel and Distributed Systems*, 11(1):6–20.

[15] Myerson, R. B. (1991). *Game Theory, Analysis of Conflicit*. Harvard University Press.

[16] Redbooks, I. (1998). *Load-Balancing Internet Servers*. IBM Corp.

[17] Redbooks, I. (1999). *Load Balancing for eNetwork Communications Servers*. IBM Corp.

[18] Ross, K. W. and Yao, D. D. (1991). Optimal load balancing and scheduling in a distributed computer system. *Journal of the ACM*, 38(3):676–690.

[19] Tantawi, A. N. and Towsley, D. (1985). Optimal static load balancing in distributed computer systems. *Journal of the ACM*, 32(2):445–465.

[20] Weber, R. R. (1978). On the optimal assignment of customers to parallel servers. *Journal of Applied Probability*, 15:406–413.

[21] Whitt, W. (1986). Deciding which queue to join: Some counterexamples. *Operations Research*, 34(1):55–62.

[22] Winston, W. (1977). Optimality of the shortest line discipline. *Journal of Applied Probability*, 14:181–189.

[23] Zhou, S. (1988). A trace-driven simulation study of dynamic load balancing. *IEEE Transactions On Software Engineering*, 14(9):1327–1341.

Admission Control to an M/M/1 Queue under Periodic Observations with Average Cost Criterion

Jianhua Cao and Christian Nyberg

Submitted to Operations Research, April 2004

Abstract.We consider the problem of admission control to an M/M/1 queue under periodic observations with average cost criterion. The admission controller receives the system state information every τ:th second and can accordingly adjust the acceptance probability. For a period of τ seconds, the cost is a linear function of the time average of customer populations and the total number of served customers in that period. The objective is to find a stationary deterministic control policy that minimizes the long run average cost. The problem is formulated as a discrete time Markov decision process whose states are fully observable. By taking the control period τ to 0 or to ∞, the model in question generalizes two classical queueing control problems: the open and the closed loop admission control to an M/M/1 queue. Our contribution includes : (1) a proof of the existence of the average optimal policy by the vanishing discounted approach; (2) several useful lower and upper bounds of the optimal cost; (3) a proof that the optimal policy is nonincreasing with respect to the observed number of customers in the system. Numerical examples are also given.

1 Introduction

We consider the problem of admission control to an M/M/1 queue under periodic observations with average cost criterion. The admission controller receives the system state information every τ:the second and can accordingly adjust the control policy which is the probability that an arrival will be accepted into the system. For a period of τ seconds, the cost for making a particular decision is a linear function of the time average of customer populations waiting in queue and the total number of served customers in that period. The objective is to find a stationary deterministic control policy that minimizes long run average cost.

The problem in question is motivated by a recent study of admission control for an M/G/1 queue with applications to web servers and e-commerce systems [21]. An important assumption in their study is that the controller can monitor the queueing system state only at regular spaced time instances. Their control objective is to keep the the server utilization close to a prescribed target. We keep the assumption of periodic observations unchanged but consider a different objective function and limit ourself to the case of an M/M/1 queue. Our objective function definition which rewards departures and penalizes a long queue resembles to that of Naor [19] and many others, see Stidham Jr. [32], Stidham Jr. [33, 34] and references therein. Along this line of objective function definition, the optimal admission control in the cases of complete observable and non-observable queueing system are well understood see, e.g., Hassin and Haviv [12]. In a brief summary, when the queue length is completely observable, the optimal policy is of threshold type and when the queue length is not observable, the optimal policy accepts customers with a fixed probability. Of course there are many other ways of defining the the objective function. Walrand [37, Example 8.5.8 p.278], e.g., introduced an admission control model for an M/M/1 queue in which the cost is incurred for each rejection as well as for each customer in the queue per time unit. In telecommunication applications, it is a common practice to maximize the throughput (or minimize the rejection probability) while limiting the queueing delay see Altman [1, p. 2].

Many models of queueing system admission control, including ours, are based on Markov decision theory, see[35, Section 4] and references therein. One ramification to the existing admission control problems is reducing the amount of information used for decision. Much work has been done along this direction. [11] also considered a queueing system under periodic observations. There are two types of customers in his system with the same service time distribution and different arrivals processes. The control policy con-

sidered is call gapping control where the low priority customers are blocked for the next period whenever the observed queueing system state exceeds a predetermined limit. [17] studied the discrete time admission control and routing to a queueing system consisting of two parallel queues with delayed queue length information. Their objective is to minimize the total expected discounted cost in which a fixed reward is received for each admitted customers and linear holding cost is incurred for customers waiting in queue. The optimal policy in case of one period information delay is of threshold type. [18] considered a multiple-server loss model where the admission controller is informed when an admitted customer finds all servers are busy but not informed when customers depart. In this system a cost is incurred if a new arrival is blocked and an even larger cost induced if an admitted customer is blocked by servers. They proved that the threshold type policy that blocks for a certain amount of time after an admitted arrival is optimal in case of single servers.

The structure properties of queueing control problems provide guidelines for heuristic algorithm design and, thus, are often considered valuable [35]. In particular, many optimal policies of queueing system control problems turn out be monotonic which is often intuitively clear as, e.g., less should get into the queue if there are many customers already. The general paradigm is to show that the step cost is super/sub-modular and the value function in value iteration is and convex/concave [2]. To avoid further mathematical complication, early works often considered total discounted cost and shy away from average cost criteria. The necessary conditions for the existence of the optimal policy for the average cost Markov decision problems, in particular for models with unbounded cost and enumerable state spaces, are now widely known see e.g. [3][13] and Sennott [27].

The main contributions of this paper can be summarized as follows:

1. We introduce an average cost Markov decision process model that unifies two classical queueing control problems: the open loop admission control and the closed loop admission control. The existence of an average cost optimal policy is proved using the vanishing discount approach.

2. Several lower and upper bounds of the optimal cost are examined. The value of information which is the difference between the upper bound and the lower bounds of the optimal cost, is shown to have a limit as arrival rate increases.

3. We establish that the average cost optimal policy is nonincreasing. This is an important property that can be exploited to accelerate numerical computations.

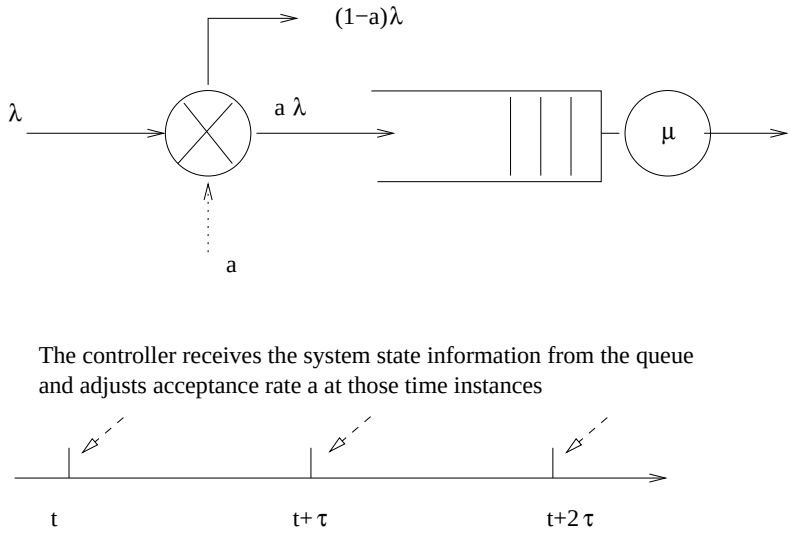


Figure 1: Arrival rate control to an M/M/1 queue with periodic observations.

This paper is organized as follows: In Section 2 we define the Markov decision process model for our problem. In Section 3 we prove the existence of an average optimal solution via the vanishing discount approach. Section 4 relates our model to two well studied problems of admission control for an M/M/1 queue. We show that the average optimal solution is nonincreasing in Section 5. Numerical examples are give in Section 6. Finally, we conclude with some remarks.

2 The Markov Decision Process Model

2.1 Classes of Control Policies and Induced Markov Chains

We control the arrival rate to an M/M/1 queue using percentage blocking. For convenience, we assume that the service rate is 1 in the rest of this paper. The arrival rate to the system is λ which can be less or greater than 1. There is an admission controller that rejects arrivals in order to maintain a reasonable response time for admitted customers. The controller is characterized by the admission probability a, $0 \leq a \leq 1$. We are allowed to adjust the admission probability every τ:th seconds when the updated system state information becomes available. Fig. 1 illustrates the model.

Let $S := \{0, 1, 2, \cdots\}$ be the set of system states. A stationary deterministic control policy is an infinite sequence $(\pi(0), \pi(1), \cdots) =: \pi$, with $0 \leq \pi(i) \leq 1, \forall i \in S$. It specifies

that whenever the observed number of customers in the system is $i \in S$, the admission probability should be adjusted to $\pi(i)$ and remain the same for the next τ seconds. Let Π be the set of stationary deterministic control policies. A more generic control class is, e.g., $\{\pi_k(i,t) \in [0,1] \mid t \in [0,\tau), k \in \mathbb{N}, i \in S\}$, where $\pi_k(i,t)$ is the admission probability at time $k\tau + t$ when the observed number of customers at time $k\tau$ is i . We shall, however, content ourselves with the class of stationary deterministic control policies in this paper.

Let $t_0, t_1, t_2, \cdots$ be the time instances when the admission probability is adjusted and $t_0 = 0, t_{k+1} - t_k = \tau, \forall k \in \mathbb{N}$. Let $X(t) \in S$ be the system state at time t. Clearly $\{X(t_k)\}_{k=0}^{\infty}$ can be regarded as a discrete time Markov chain. For convenience denote $X_k := X(t_k)$.

The transition probability between states i and j is denoted as $P(j \mid i, \pi(i))$. It is actually the probability that at time τ there are j customers in an M/M/1 queue with arrival rate $\pi(i)\lambda$ when there are i customers in the system initially. The explicit expression of this probability in the transform domain (s,z) is reviewed in Appendix A. Once a control policy $\pi \in \Pi$ and the initial state are given, the associated Markov Chain $\{X_k\}$ is fully specified. Notice that the transition probability $P(\cdot \mid \cdot, a)$ is continuous in a.

2.2 One Step Cost

Given a fixed arrival rate, one would like to accept as few customers as possible in order maintain a short mean response time. But, on the other hand, to have a high throughput, the arrival rate ought to be close to the service rate. Hence one has to balance low response time and high throughput. One way to resolve this dilemma is to formulate an constrained optimization problem where the throughput is the objective to be maximized while the response time is subject to a given constraint. Another way is to consider an unconstrained optimization problem whose objective is to minimize a cost which is a function of response time and throughput. This paper discusses the second approach.

To make the definition of cost more precise, we introduce some auxiliary notation.

Let $\bar{N}(t,i,a)$, $\bar{N} : [0,\tau) \times S \times [0,1] \mapsto [0,\infty)$, be the expected number of customers in an M/M/1 queue at time t when initially there are i customers and the admission probability is a.

The time average number of customers in the system between time 0 and τ is

$$\frac{1}{\tau}\int_0^{\tau} \bar{N}(t,i,a)\,dt.$$

The average number of severed customers(not including the rejected customers) between time 0 and τ is

$$\frac{1}{\tau}\left(i + a\lambda\tau - \bar{N}\left(\tau, i, a\right)\right).$$

Let C be the cost to maintain one customer in the system per time unit and R be the reward for a departure. To avoid the trivial case, we assume that $0 < C < R$. The one-step cost, i.e. the time average of net cost incurred in a period of τ seconds between two adjacent control parameter adjustment instances, is denoted as $c\left(i, a\right)$, and is defined as follows,

$$c\left(i, a\right) := \frac{C}{\tau}\int_0^\tau \bar{N}\left(t, i, a\right) dt - \frac{R}{\tau}\left(i + a\lambda\tau - \bar{N}\left(\tau, i, a\right)\right). \tag{1}$$

It is easy to convince oneself that the step cost $c\left(i, a\right)$ is not necessarily monotonic in i or a. Moreover the one-step cost $c\left(i, a\right)$ is unbounded as i increases,

Lemma 1. *$c\left(i, a\right) \to \infty$ as $i \to \infty$.*

Proof. It is enough to show that a low bound of $c\left(i, a\right)$ is increasing linearly in i. Let's consider a system with i real customers and infinitely many phantom customers who queue after those real customers. Only real customers incur a cost C for each time unit and each customer in the system. The system receives reward R for each served customer no matter real or not. In other words, there is an unlimited supply of customers who will bring only profit but not costs since the only cost is serving i real customers. Let d be the time average cost for a period of τ seconds of this system. The following relation then holds

$$c\left(i, a\right) \geq d \geq C\bar{N}\left(\tau, i, 0\right) - R.$$

Clearly $\bar{N}\left(\tau, i, 0\right) \geq \left(1 - e^{-\tau}\right) i$ since serving i customers with one server is not as fast as using i servers working in parallel. □

2.3 Average Cost and β-Discount Cost

The expected long-run average cost or average cost incurred by a policy $\pi \in \Pi$ is defined as

$$J\left(i, \pi\right) := \limsup_{N\to\infty} E_i^\pi\left[\frac{1}{N}\sum_{k=0}^{N-1} c\left(X_k, \pi\left(X_k\right)\right)\right],$$

where i is the initial number of customers in the system .

The optimal average cost for any initial state $i \in S$ is defined as

$$J^*(i) := \inf_{\pi \in \Pi} \{J(i, \pi)\}.$$

A policy π is average cost optimal, or average optimal, if $J(i, \pi) = J^*(i)$ for all $i \in S$.

While the average cost is of primal interest of this paper, β-discounted cost is helpful in our proof of the existence of the average optimal policy.

Given some $\beta \in (0, 1)$, the β-discount cost incurred by a policy $\pi \in \Pi$ is defined as

$$J_\beta(i, \pi) := E_i^\pi \left[\sum_{k=0}^{\infty} \beta^k c(X_k, \pi(X_k)) \right].$$

For any initial state $i \in S$, the optimal β-discount cost is defined as

$$J_\beta^*(i) := \inf_{\pi \in \Pi} \{J_\beta(i, \pi)\}$$

A policy π is β-discount cost optimal, or β-discount optimal, if $J_\beta(i, \pi) = J_\beta^*(i)$ for all $i \in S$.

3 The Existence of Average Cost Optimal Policy

The existence of an average optimal policy for a Markov decision process with finite state space, enumerable action sets and bound costs is well understood [3], but our model is characterized by the enumerable infinite state space S and the unbounded one-step cost $c(i, a)$ (Lemma 1). The existence of average cost optimal policy in such cases are not so obvious as the counterexamples in Ross [22, p.90], Ross [24, p. 142] and Sennott [27, Section 7.1] show.

There are at least three well developed methods to prove the existence of the average optimal policy when the state space is enumerable infinite and step cost is unbounded see Arapostathis et al. [3, Section 5.2 and Section 5.3] for review. The first one is Hordijk's Lyapunov stability condition [15]. Sennott's three necessary conditions [26, 27, 28] can be counted as the second. The third one is Borkar's convex analytic approach [5, 6]. Sennott's conditions are particularly well suited for queueing problems [27] where the step cost often, if not always, grows as more customers accumulate in the queue. Our proof of the existence of the average optimal policy for our particular model is based on Sennott's conditions.

A complete proof of the existence of average optimal policy for a Markov decision

process satisfies Sennott's three conditions relies on the vanishing discounted approach. The general idea is that one treats the average cost case as the limit of the discounted cost problem. The vanishing discounted approach was also used by [4] to prove the existence of average optimal policy for a model with finite state space and action set, and [9] for countable state space, finite action set and bounded costs.

The existence β-discount optimal policy of our problem is easy to check, see Arapostathis et al. [3, Lemma 2.1]. We shall verify Sennott's three conditions in our particular problem in Lemma 2 and prove the existence of average optimal policy using vanishing discount approach in Theorem 4. The proof of Theorem 4 is mainly based on Arapostathis et al. [3, Theorem 5.9].

Lemma 2. *(a) For every $i \in S$ and every $\beta \in (0,1)$, $J_\beta^*(i) < \infty$.*

(b) There exists a nonnegative integer L such that

$$h_\beta(i) := J_\beta^*(i) - J_\beta^*(0) \geq -L$$

(c) There exists a function $M : S \mapsto \mathbb{R}$ such that $h_\beta(i) \leq M(i)$ for all $i \in S$ and $\beta \in (0,1)$ and for every $i \in S$ and $a \in [0,1]$ such that $\sum_j P(j \mid i,a) M(j) < \infty$.

Proof. (a). Clearly there exists a stationary deterministic policy, e.g $\pi := (0,0,\cdots)$ such that the induced MC is ergodic and $J(i,\pi) = 0$ for all $i \in S$. By the Tauberian Theorem see Appendix B, we have

$$\liminf_{\beta\uparrow 1} (1-\beta) J_\beta^*(i) \leq \limsup_{\beta\uparrow 1} (1-\beta) J_\beta^*(i) \leq J(i,\pi)$$

Hence $J_\beta^*(i)$ is finite for all $i \in S$ and $\beta \in (0,1)$.

(b) Consider a queuing system with two M/M/1 queues, one queue with i customers initially but no arrivals, another queue starts empty and the β-discount optimal policy is employed. The discounted cost for the whole system is therefore $J_\beta^*(0) + J_\beta(i,\mathbf{0})$, where $\mathbf{0} = (0,0,\cdots)$ is a policy that rejects all arrivals. Clearly $J_\beta^*(0) + J_\beta(i,\mathbf{0}) \leq J_\beta^*(i)$, hence we have

$$J_\beta^*(i) - J_\beta^*(0) \geq J_\beta(i,\mathbf{0})$$

Let $n := \lfloor R/C \rfloor$. For all $i \in S$, $J_\beta(i,\mathbf{0}) \geq J_\beta(n,\mathbf{0}) \geq -nR$. Therefore $h_\beta(i) \geq -\lfloor R/C \rfloor R$.

(c) Let Π^* be the class of policies inducing irreducible, ergodic MC and $c_{i,0}(\pi)$ the expected total cost of a first passage from i to 0. Clearly Π^* is not empty. Let $M(0) = 0$ and for $i \geq 1$, $M(i) = \inf_{\pi \in \Pi^*} c_{i,0}(\pi)$. We then have $J^*_\beta(i) \leq M(i) + J^*_\beta(0)$ and for all $i \in S$ and $a \in [0,1]$, and some $\pi \in \Pi^*$,

$$\sum_{j \in S} P(j \mid i,a) M(j) \leq \sum_j P(j \mid i,a) c_{i,0}(\pi) = c_{i,0}(\pi) < \infty.$$

□

Lemma 3. *Let M be a function such that $M : S \mapsto \mathbb{R}$ and $|M(i)| < \infty$, for all $i \in S$. Let $\{f_n\}$be an infinite sequence of functions such that $f_n : S \mapsto \mathbb{R}$ and $|f_n(i)| \leq |M(i)|$. Then there exists a subsequence $\{n_k\}$ of $\{n\}$ such that there is a function f, $f : S \mapsto \mathbb{R}$, and $f_{n_k}(i) \to f(i)$for all $i \in S$.*

Proof. From $\{n\}$ we can extract $\{n_{0,k}\}$ such that $\lim_{k\to\infty} f_{n_{0,k}}(0) =: f(0)$. From $\{n_{0,k}\}$ we can extract $\{n_{1,k}\}$ such that $\lim_{k\to\infty} f_{n_{1,k}}(1) =: f(1)$. And so on. Clearly $\{n_{k,k}\}$or simply $\{n_k\}$is the desired subset and f is the desired function. □

This proof is sometimes referred diagonalizing procedure. The sequence$\{n_{k,k}\}$ is called the "Cantor" diagonal sequence. The $\{f_n\}$is said to be point-wise convergent to f.

Theorem 4. *There exists an average optimal policy $\pi \in \Pi$.*

Proof. Let $\{n\}$be a sequence such that $\beta_n \uparrow 1$ and $\beta_n \in (0,1)$. Let π_n be the corresponding β_n-discount optimal policy. By Lemma 3, there exist a subsequence $\{n_k\}$ of $\{n\}$, a policy $\varphi \in \Pi$ and a function h, $h : S \mapsto \mathbb{R}$ such that $\lim_{k\to\infty} \beta_{n_k} \to 1$, $\forall i \in S$, $\lim_{k\to\infty} \pi_{n_k}(i) = \varphi(i)$ and $\lim_{k\to\infty} h_{\beta_{n_k}}(i) = h(i)$, where $h_{\beta_{n_k}}(i) = J^*_{\beta_{n_k}}(i) - J^*_{\beta_{n_k}}(0)$. To simplify the notation $\{n_k\}$ shall be denoted as $\{n\}$ in the rest of this proof. Since $J^*_{\beta_n}(i)$ is finite, denote $\rho(i) := \lim_{\beta_n \uparrow 1} (1-\beta_n) J^*_{\beta_n}(i)$. By Lemma 2(c),

$$|\rho(i) - \rho(0)| \leq \lim_{\beta \uparrow 1} (1-\beta) \max\{|M(i)|, L\} = 0$$

Moreover because $-R \leq \rho(i) \leq C$, $\rho(i)$ is a finite constant. Let $\rho := \rho(i)$.

We shall show that $\inf_{\pi \in \Pi}\{J(i,\pi)\} = \rho$ and that the policy φ is indeed average optimal, i.e. $J(i,\varphi) = \rho, \forall i \in S$,

Since π_n is β_n-DC optimal, $\forall i \in S$

$$(1-\beta_n)\, J^*_{\beta_n}(0) + h_{\beta_n}(i) = c(i,\pi_n(i)) + \beta_n \sum_{j\in S} P(j \mid i,\pi_n(i))\, h_{\beta_n}(j). \tag{2}$$

Since $P(\cdot \mid i,a)$ is continuous in a, by Fatou's lemma [25, p. 23][30, p. 187],

$$\begin{aligned}\liminf_{\beta_n\uparrow 1} \beta_n \sum_{j\in S} P(j \mid i,\pi_n(i))\, h_{\beta_n}(j) &\geq \sum_{j\in S} \liminf_{\beta_n\uparrow 1} \beta_n P(j \mid i,\pi_n(i))\, h_{\beta_n}(j) \\ &= \sum_{j\in S} P(j \mid i,\varphi(i))\, h(j)\end{aligned}$$

Taking "liminf" on the both sides of (2), we have

$$\rho + h(i) \geq c(i,\varphi(i)) + \sum_{j\in S} P(j \mid i,\varphi(i))\, h(j).$$

Let $\{X_k\}$be the MC under control of φ. By the towering property of the conditional expectation and Lemma 2(b), we have

$$\begin{aligned}\rho^* &\geq \lim_{N\to\infty} \frac{1}{N} E_i^{\varphi}\left[\sum_{k=0}^{N-1} c(X_k,\pi)\right] + \frac{1}{N}\left(E_i^{\varphi}[h(X_N)] - h(i)\right) \\ &\geq \lim_{N\to\infty} \frac{1}{N} E_i^{\varphi}\left[\sum_{k=0}^{N-1} c(X_k,\pi)\right] + \frac{1}{N}\left(-L - M(i)\right) \\ &= J(i,\varphi)\end{aligned}$$

Hence $\rho \geq J(i,\varphi)$. On the other hand by Tauberian theorem, and for all $\pi \in \Pi$

$$\begin{aligned}J(i,\pi) &= \limsup_{N\to\infty} \frac{1}{N} E_i^{\pi}\left[\sum_{k=0}^{N-1} c(X_k,\pi)\right] \\ &\geq \limsup_{n\to\infty} (1-\beta_n)\, E_i^{\pi_n}\left[\sum_{k=0}^{\infty} \beta_n^k c(X_k,\pi)\right] \\ &\geq \lim_{\beta\uparrow 1} (1-\beta)\, J^*_{\beta}(i) \\ &= \rho.\end{aligned}$$

Thus $J(i,\varphi) = \rho$ and φ is average optimal. □

In fact we have just proved that if an average cost Markov decision process problem with countable infinite sate space, compact action set and unbounded step cost satisfies

the conditions in Lemma 2, then the average optimal policy exists. Moreover Lemma 2(c) can be loosed in the following form and still guarantee that the average optimal policy exists:

Lemma 5. *There exists a function* $M : S \mapsto \mathbb{R}$ *such that* $h_\beta(i) \leq M(i)$ *for all* $i \in S$ *and* $\beta \in (0,1)$ *and for every* $i \in S$ *there exists* $a \in [0,1]$*and such that* $\sum_j P(j \mid i,a) M(j) < \infty$.

4 Relations with the Two Classical Queueing Control Models

Our model is parametrized by the control interval τ, the arrival rate λ, the cost coefficient C and the reward coefficient R. In two extremes: when $\tau \to \infty$ and $\tau = 0$, our model degenerates into, respectively, the admission control without state information (referred to as open loop control) and with complete information (referred to as closed loop control). In this section we first examine the average optimal costs and policies for the open and the closed loop admission control models. The optimal average costs of the open and the closed control model are then exploited to establish the upper and the lower bounds of the optimal costs for models in general cases $0 \leq \tau$. Finally we show that the value of information which is the difference between the previous established upper and lower bounds increases with respect to the arrival rate but has a limit.

4.1 Open Loop Admission Control

When $\tau \to \infty$, the step cost is independent of i,

$$\lim_{\tau \to \infty} c(i,a) = \begin{cases} C\frac{a\lambda}{1-a\lambda} - Ra\lambda & \text{if } a < \frac{1}{\lambda} \\ \infty & \text{otherwise} \end{cases}$$

On the other hand the step cost becomes the average cost. Therefore the optimal admission control probability a^* is such that

$$a^* = \arg \min_{0 \leq a < \left(1 \wedge \frac{1}{\lambda}\right)} C\frac{a\lambda}{1-a\lambda} - Ra\lambda.$$

which can be solved rather easily:

$$a^* = \begin{cases} 1 & \text{if } \lambda \leq \left(1 - \sqrt{C/R}\right) \\ \frac{1}{\lambda}\left(1 - \sqrt{C/R}\right) & \text{otherwise} \end{cases}$$

The corresponding optimal average cost is $-\left(\sqrt{R} - \sqrt{C}\right)^2$.

4.2 Closed Loop Admission control

When $\tau = 0$

$$c(i,a) = \begin{cases} 0 & \text{if } i = 0 \\ Ci - R\mu & \text{otherwise} \end{cases}$$

Theorem 6. *The average optimal policy φ of the closed loop admission control is of threshold type, i.e. there exists some $n \in \mathbb{N}$ such that*

$$\varphi(i) = \begin{cases} 1 & \textit{if} \quad i < n \\ 0 & \textit{otherwise} \end{cases}$$

Proof. We shall show that the policy $\varphi(i)$is nonincreasing in i and is either 1 or 0.

Notice that the admission control decision is not enforced until a customer arrives. After applying the uniformization procedure [7, p. 3], the original formulation is equivalent to a discrete time problem which is embedded at the arrival and the departure epochs of the original process. The transition probability for the constructed discrete time process is as follows,

$$\tilde{P}(j \mid i,a) = \begin{cases} \frac{a\lambda}{\lambda+1} & \text{if } j = i+1 \\ \frac{(1-a)\lambda}{\lambda+1} & \text{if } j = i \\ \frac{1}{\lambda+1} & \text{if } j = i-1 \\ 0 & \text{otherwise} \end{cases}$$

for $i > 0$ and

$$\tilde{P}(j \mid i,a) = \begin{cases} a & \text{if } j = 1 \\ 1-a & \text{if } j = 0 \\ 0 & \text{otherwise} \end{cases}$$

for $i = 0$.

The step cost is redefined as follows

$$\tilde{c}(i,a) = \begin{cases} \frac{a\lambda}{\lambda+1}C & \text{if } i = 0 \\ \frac{\lambda}{\lambda+1}Ci + \frac{a\lambda}{\lambda+1}C - \frac{1}{\lambda+1}R & \text{if } i > 0 \end{cases}$$

The discrete time process is stochastically equivalent to the original one. But now we have a controlled birth death process with a cost which a linear function of state and action. We can verify that the similar condition prescribed in Lemma 2 is satisfied and therefore by Theorem 4, the existence of average optimal policy is assured.

Let ρ be the optimal average cost and $h(i) := \lim_{\beta \uparrow 1} J_\beta^*(i) - J_\beta^*(0)$ where $J_\beta^*(i)$ is the optimal β-discount cost. The average optimal policy φ is the minimal selector for the average cost optimality equation,

$$\rho + h(i) = \min_a \tilde{c}(i,a) + \sum_{j \in S} \tilde{P}(j \mid i,a)\, h(j), \ \forall i \in S \tag{3}$$

Let $g(i,a) := \tilde{c}(i,a) + \sum_{j \in S} \tilde{P}(j \mid i,a)\, h(j) - h(i)$, it can be shown that $g(i,a)$ is supermodular in (i,a), i.e $g(i,a') - g(i,a)$ is non-decreasing in i for all $a' > a$. Hence the optimal policy must be a non-increasing sequence with respect to i. Moreover $g(i,a)$ is linear in a, hence $\varphi(i) = \arg\min_a g(i,a)$ is either 1 or 0. □

Since the optimal policy is of threshold type, finding the optimal threshold value can be achieved through solving the following optimization problem

$$n = \arg\min_k C\frac{\lambda\left(1 - (1+k)\lambda^k + k\lambda^{1+k}\right)}{(1-\lambda)(1-\lambda^{k+1})} - R\frac{\lambda\left(1-\lambda^k\right)}{(1-\lambda^{1+k})} \tag{4}$$

where the first item in the r.h.s. above is the cost coefficient times the average number of customers in an M/M/1/k queue and the second item in the r.h.s. above is the reward coefficient times the carried traffic intensity.

Notice that as the arrival rate becomes large, the optimal threshold becomes small. In the extreme case when $\lambda \to \infty$, the optimal threshold is 1 and the corresponding average cost is $C - R$.

4.3 Lower and Upper Bounds of the Optimal Average Cost

In general, our model is parametrized by the tuple (λ, R, C, τ). When we study how the solution varies with a particular parameter, e.g. λ, with a little abuse of notion, the associated optimal average cost is denoted as $J^*(\lambda)$ and the average cost under policy $\pi \in \Pi$ is denoted as $J(\lambda, \pi)$ We first establish a simple fact.

Lemma 7. *$J^*(\lambda)$is non-increasing in λ.*

Proof. Assume that $\lambda_1 < \lambda_2$. Let π_1 be the average optimal policy corresponding to $J^*(\lambda_1)$. Let $\varphi \in \Pi$ be a policy such that $\varphi(i) = \frac{\lambda_1}{\lambda_2}\pi_1(i), \forall i \in S$. Then $J^*(\lambda_1) = J(\lambda_1, \pi_1) = J(\lambda_2, \varphi) \geq J^*(\lambda_2)$. □

The optimal average costs for the closed and open loop control models, which are parameterized by $(\lambda, R, C, 0)$ and (λ, R, C, ∞) respectively, provide a lower bound and an upper bound. However, the lower bound does not render an explicit expression except in some special cases such as $\lambda = 1$ or $\lambda = \infty$. The following result is easy to verify,

Theorem 8. *Let J^* be the optimal average cost associated with the model parametrized by (λ, R, C, τ). Then $C - R < J^* \leq -\left(\sqrt{R} - \sqrt{C}\right)^2$.*

The lower bound can be tightened further (almost half), in certain cases and still remain an explicit expression.

Theorem 9. *When $1 \leq R/C \leq 3/2$ and $\lambda \leq 1$, $\frac{(C-R)}{2} \leq J^*$*

Proof. It is sufficient to show that the optimal AC of the closed loop control model J^* is $\frac{C-R}{2}$ when $\lambda = 1$ and $R/C \leq 3/2$. From (4), we have

$$J^* = \min_k C\frac{k}{2} - R\frac{k}{k+1}.$$

When $1 \leq R/C \leq 3/2$, the optimal threshold $n = \arg J^* = 1$ because $C\frac{k}{2} - R\frac{k}{k+1}$ is increasing in k. □

Note that a tighter upper bound of the optimal cost can be obtained from the first step of the value iteration. Due to the not-so-simple definition of the one-step cost function $c(n, a)$, it is difficult to find the explicit expression for this upper bound.

4.4 The Value of Information

We define the value of information as the difference of the optimal average costs between the open and the closed loop control models,

$$V(\lambda) := \lim_{\tau \to \infty} J^*(\lambda, \tau) - J^*(\lambda, 0)$$

when the cost coefficient C and the reward coefficient R are given.

By Lemma 7, $V(\lambda)$ increases in λ. It is, however, interesting to note that the value of information is bounded.

Theorem 10. $\lim_{\lambda \to \infty} V(\lambda) = 2\left(\sqrt{R/C} - 1\right) C$.

Proof. Recall for the open loop control model, one have $\lim_{\tau \to \infty} J^*(\lambda, \tau) = -\left(\sqrt{R} - \sqrt{C}\right)^2$ and for the closed loop control model we have $\lim_{\lambda \to \infty} J^*(\lambda, 0) = C - R$. □

Since the argument above involves nothing more than the optimal solution to the open and the closed control models, the result can be further generalized to a model with K identical exponential servers in parallel. Define the value of information $V_K(\lambda)$ similarly. One can easily show that $\lim_{\lambda \to \infty} V_K(\lambda) = 2\left(\sqrt{R/C} - 1\right) CK$.

5 The Monotonic Property of the Average Cost Optimal Policy

Intuitively when more customers are observed in one control instance, less should be admitted into the system in the next control period of τ seconds. Hence it is reasonable to conjecture that the average optimal policy φ is nonincreasing in i, i.e. $\varphi(i) \geq \varphi(i+1), \forall i \in S$. This fact can be easily verified in the cases of closed loop and open loop admission control. In this section we shall prove that this is true in general when $0 \leq \tau < \infty$. Let $N(t, i, a)$ be the number of customers in a queue (with arrival rate $a\lambda$ and service rate 1) at time t with initial condition $N(0, i, a) = i$. Notice that given t, i, a, $N(t, i, a)$ is a random variable.

A function $f : \mathbb{N} \to \mathbb{R}$ is discrete convex if $f(i+2) - f(i+1) \geq f(i+1) - f(i)$ for all $i \in \mathbb{N}$. Let $\mathfrak{F}$ be the space of discrete convex functions that map from S to $\mathbb{R}$. The following lemma follows.

Lemma 11. *Given $t > 0$, for all $f \in \mathfrak{F}$, $E[f(N(t, i, a))]$ is supermodular in (i, a).*

Proof. We have to show that for all $f \in \mathfrak{F}$, $i \in S$, $0 \leq b < a \leq 1$,

$$E\left[f\left(N\left(t, i+1, a\right)\right)\right] + E\left[f\left(N\left(t, i, b\right)\right)\right] \geq E\left[f\left(N\left(t, i, a\right)\right)\right] + E\left[f\left(N\left(t, i+1, b\right)\right)\right]$$

The following stochastic order relations can be verified by the normalization technique: $N\left(t, i+1, a\right) \geq_{st} N\left(t, i, a\right)$, $N\left(t, i+1, a\right) \geq_{st} N\left(t, i+1, b\right)$, $N\left(t, i+1, b\right) \geq_{st} N\left(t, i, b\right)$, $N\left(t, i+1, a\right) - N\left(t, i, a\right) \geq_{st} N\left(t, i+1, b\right) - N\left(t, i, a\right)$. Since f is discrete convex, we have $f\left(N\left(t, i+1, a\right)\right) - f\left(N\left(t, i, a\right)\right) \geq_{st} f\left(N\left(t, i+1, b\right)\right) - f\left(N\left(t, i, a\right)\right)$ which leads to the claim after taking expectation on both sides. □

Lemma 12. *Given $t > 0$, for all $f \in \mathfrak{F}$. $i \in S$, $0 \leq b < a \leq 1$,*

$$E\left[f\left(N\left(t, i, a\right)\right)\right] + E\left[f\left(N\left(t, i+2, b\right)\right)\right] \geq 2E\left[f\left(N\left(t, i+1, \frac{a+b}{2}\right)\right)\right]$$

Proof. Since f is discrete convex, it is enough to show that

$$N\left(t, i, a\right) + N\left(t, i+2, b\right) \geq_{st} 2N\left(t, i+1, \frac{a+b}{2}\right). \tag{5}$$

Consider a system consisting of two identical but independent M/M/1 queues. The total arrival rate is $(a+b)\lambda$ and initially there are $2i+2$ customers in total in two queues. Suppose we are allowed to distribute $2i+2$ customers into to two queues at time 0 and split the incoming traffic by random routing. Total number of customers in the whole system at time t can be minimized (in stochastic order sense) by placing $i+1$ customers in each queue and splitting traffic evenly. Thus (5) holds. □

Lemma 13. *Define $g : \mathfrak{F} \times S \times [0,1] \mapsto \mathbb{R}$ as follows $g(f)\left(i, a\right) := c\left(i, a\right) + E\left[f\left(N\left(\tau, i, a\right)\right)\right]$, then $g\left(f\right)\left(i, a\right)$is supermodular in (i, a) and $\min_a g\left(f\right)\left(i, a\right)$ is convex in i.*

Proof. (Supermodularity) By Lemma 11, $E\left[N\left(t, i, a\right)\right]$ is supermodular in (i, a). Recall the definition of the step cost

$$c\left(i, a\right) = \frac{C}{\tau}\int_0^\tau E\left[N\left(t, i, a\right)\right] dt - \frac{R}{\tau}\left(i + a\lambda\tau - E\left[N\left(\tau, i, a\right)\right]\right).$$

Hence $c\left(i, a\right)$ is supermodular in (i, a) too. Also from Lemma 11, $E\left[f\left(N\left(\tau, i, a\right)\right)\right]$ is supermodular. Thus $g\left(f\right)\left(i, a\right)$ is supermodular in (i, a).

(Convexity) We have to show that $\forall i \in S$

$$\min_a g\left(f\right)\left(i+2, a\right) + \min_a g\left(f\right)\left(i, a\right) \geq 2\min_a g\left(f\right)\left(i+1, a\right).$$

We shall show that there exist some $w \in [0,1]$ such that

$$\min_a g(f)(i+2,a) + \min_a g(f)(i,a) \geq 2g(f)(i,w) \tag{6}$$

Let $u := \arg\min_a g(f)(i,a)$, $v := \arg\min_a g(f)(i+2,a)$. By the supermodularity of g, $v \leq u$. If $u = v$, then let $w = u = v$, the inequality above is evident. If $v < u$ then let $w = \frac{u+v}{2}$. By Lemma 12,

$$E[N(t,i+2,v)] + E[N(t,i,u)] \geq 2E[N(t,i+1,w)]$$

and

$$E[f(N(t,i+2,v))] + E[f(N(t,i,u))] \geq 2E[f(N(t,i+1,w))].$$

Therefore (6) is valid. □

Theorem 14. *If $\varphi \in \Pi$ is average optimal, then $\varphi(i) \geq \varphi(i+1)$ for all $i \in S$.*

Proof. Consider the value iteration. The value function $V_k : S \mapsto \mathbb{R}$ at the k:th iteration is defined as follows

$$V_{k+1}(i) = \min_a \left\{ c(i,a) + \sum_j V_k(j) P(j \mid i,a) \right\}$$

with $V_0(j) = 0, \forall j \in S$. The average optimal policy is defined as follows

$$\varphi(i) = \lim_{k\to\infty} \arg\min_a c(i,a) + \sum_j V_k(j) P(j \mid i,a), \forall i \in S$$

By induction, we have that for all k, $V_k(i)$ is convex in i and $c(i,a) + \sum_j V_k(j) P(j \mid i,a)$ is supermodular in (i,a). Hence $\varphi(i)$ is nonincreasing in i. □

6 Numerical Examples

In our model, the state space is countable infinite. For numerical calculations, however, the state space must be truncated (cutting off the tail). Rigors analysis of the effects of state space truncation for dynamic programming in general can be found in Fox [10], Whitt [38, 39]. The approximating sequence method described in Sennott [27, Chapter 8] is also very illuminating. In our particular case, we note that the difference in the optimal

cost between the original model and the state space truncated model decreases as the probability of unexpected blocking diminishes. Once the state space is truncated, both the value iteration algorithm and the policy iteration algorithm [20, Section 8.5 and 8.6] can be used to calculate the optimal policy and the optimal average cost. The policy iteration algorithm which is used in our numerical examples is described briefly as follows.

Step 1 Initialization: An arbitrary stationary policy π is chosen.

Step 2 Value determination: For the current policy π, compute the unique solution $\{J, h_i\}$ for the following system of linear equations

$$\begin{aligned} h_i &= c\left(i, \pi\left(i\right)\right) - J + \sum_{j \in S'} h_j P\left(j \mid i, \pi\left(i\right)\right), \quad \forall i \in S' \\ h_0 &= 0 \end{aligned}$$

where S' is the truncated state space.

Step 3 Policy improvement: The new policy $\pi^{'}$is obtained as follows

$$\pi^{'}\left(i\right) = \arg \min_{0 \leq a \leq 1} c\left(i, a\right) - J + \sum_{j \in S'} h_j P\left(j \mid i, a\right) \quad \forall i \in S^{'}.$$

Step 4 Convergence test. Let $d\left(\pi', \pi\right)$ be a distance measure of two policies such as follows $d\left(\pi', \pi\right) = \sum_{i \in S'} \left|\pi^{'}\left(i\right) - \pi\left(i\right)\right|$. If the new policy $\pi^{'}$ is within the give distance ϵ of the old policy π, i.e. $d\left(\pi^{'}, \pi\right) < \epsilon$, the algorithm stops, otherwise goto step 2 with π replaced by $\pi^{'}$.

Note that due the state space truncation, the expression for the average number of customers in the system $\bar{N}\left(i, t, a\right)$ and the transition probability $P\left(j \mid i, a\right)$ must be swapped from the M/M/1 model to the M/M/1/K model where K is the truncated queue size. See Appendix A for more information.

Using the same techniques in proving Lemma 13 and Theorem 14, one can show that the function to be minimized in the policy improvement step is convex and supermodular if the policy chosen in the initialization step is nonincreasing. As a result many convex programming techniques can be used to find the minimum in step 3 and further, the search region for $\pi^{'}\left(i\right)$ can be narrowed from $[0, 1]$ to $\left[0, \pi^{'}\left(i-1\right)\right]$.

In the following discussions, we set the reward coefficient to a constant $R = 100$.

In Fig. 2, 3 and 4, we show how the optimal average cost changes as the control interval τ increases for different combinations of cost coefficients $C = 10, 30, 50$ and arrival rates $\lambda = 0.7, 1.0, 1.5, 2.0$. Notice that when the offered traffic intensity and the cost of waiting

are high, e.g. $C = 50, \lambda = 2.0$, the optimal average cost for models with short control intervals is sensitive to τ. We also plot the theoretical lower and the upper bound of the optimal average cost in these figures.

We have shown in Section 5 that the average optimal policy is nonincreasing. In Fig. 5, we give examples of the average optimal control policies in cases of $\tau = 0.1, 5.0, 10.0$ when $\lambda = 1.0$ and $C = 30$. The optimal control policy in the case of open loop control $\tau \to \infty$ is also shown. If the closed loop control is used (i.e. $\tau = 0.0$) when $\lambda = 1.0, C = 30$, the average optimal policy is of threshold type and the optimal threshold is 1.

The traditional performance metrics such as average response time T and throughout H can easily be calculated once a control policy is given. In Fig. 6 and 7 we show how the average response time and the throughput change with the arrival rate in cases of $C = 10, 30, 50$ when $\tau = 5.0$. For comparison the plot of the optimal average cost versus the arrival rate for the same group of system configurations is shown in Fig. 8. The trade-off between the throughput and the average response time becomes evident in those plots. For example, the average response time becomes lower at the expense of lower throughput when the waiting cost coefficient C is high.

Recall that the value of information is defined as the difference of optimal costs between the open loop controlled model and the closed loop controlled model in Section 4.4. We have showed that the value of information increases but posses a limit as the arrival rate increases. In Fig. 9, we plot the value of information versus arrival rate in the case $C = 10.0$. Moreover, it is interesting that the value of information increases almost linearly when the arrival rate is small.

7 Conclusions

We have presented an admission control model for an M/M/1 queue under periodic observations with average cost criterion. This model degenerates into two well known queueing control problems when the observation interval becomes zero and infinite. The corresponding discrete time Markov decision process is obtained via embedding the state transition instances at the state information update epochs. The state transition probability is related to the transient solution of state probability distribution of an M/M/1 queue. The step cost is linearly proportional to the time average number of customers in the queue and the time average number of departures per time unit between two adjacent control instances. The existence of average optimal policy is proved via the vanishing discount

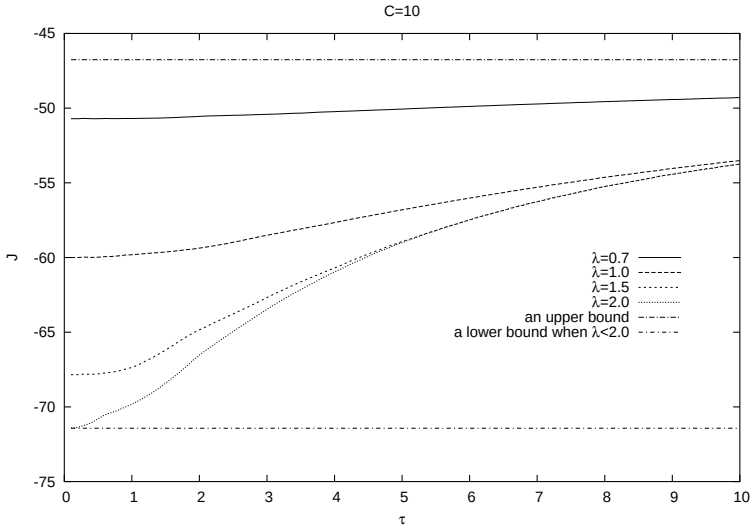


Figure 2: Average optimal cost versus the control interval τ when $C = 10$

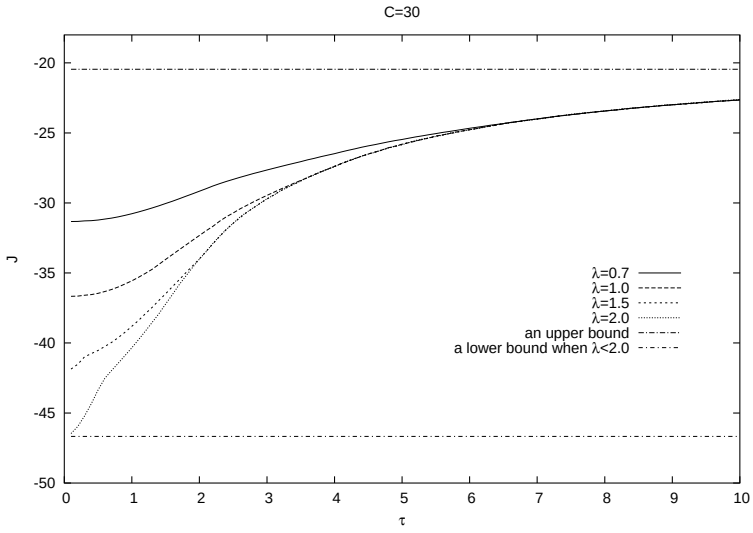


Figure 3: Average optimal cost versus the control interval τ when $C = 30$

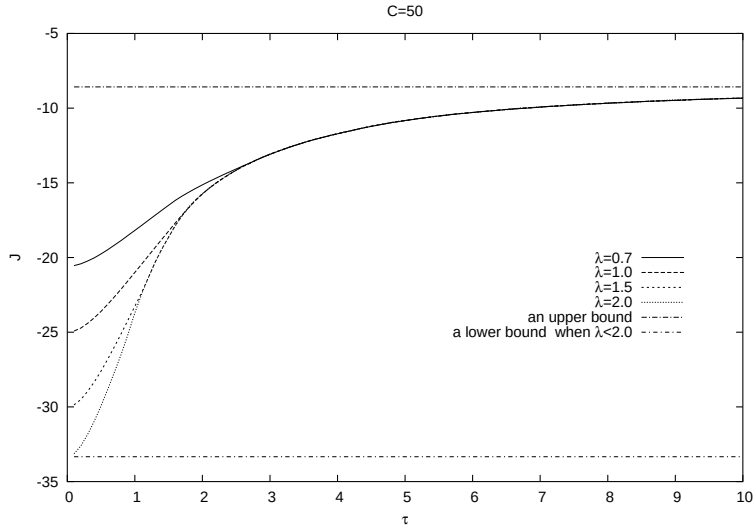


Figure 4: Average optimal cost versus the control interval τ when $C = 50$.

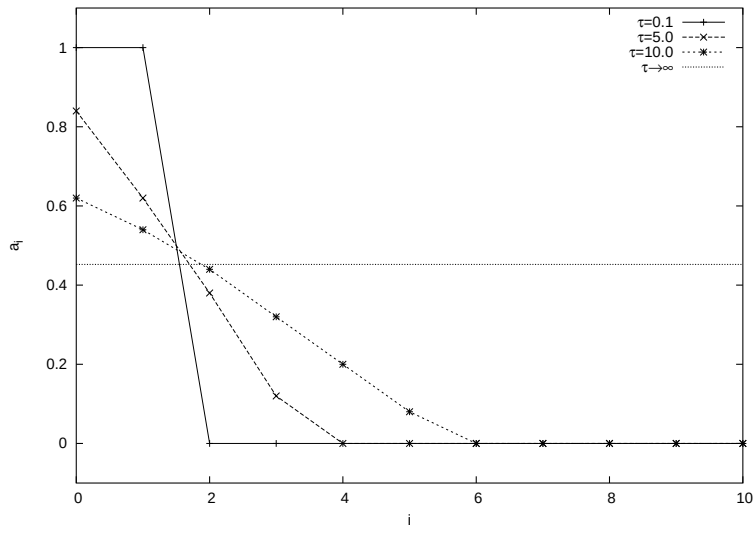


Figure 5: The average optimal control policy when $\lambda = 1.0$ and $C = 30$.

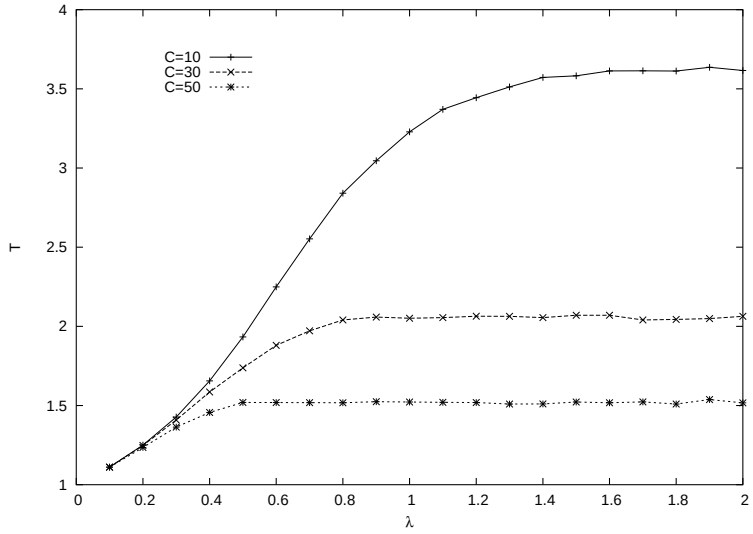


Figure 6: Average Response time versus arrival rate(when $\tau = 5.0$).

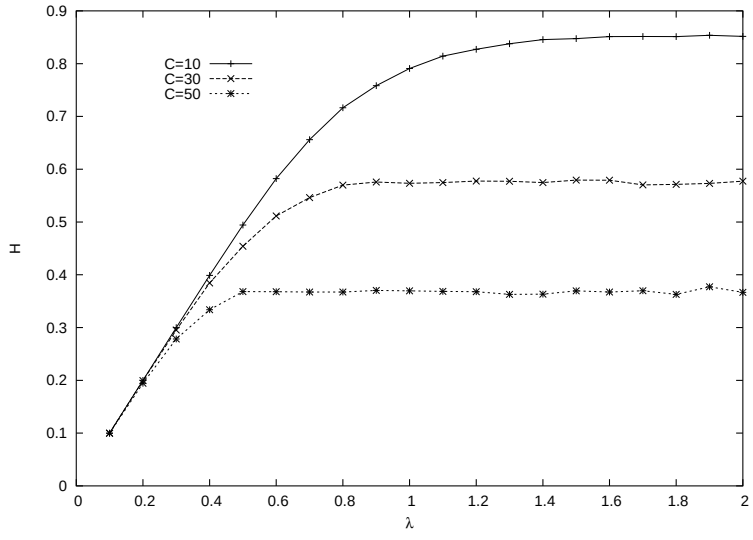


Figure 7: Average throughput versus arrival rate when $\tau = 5.0$.

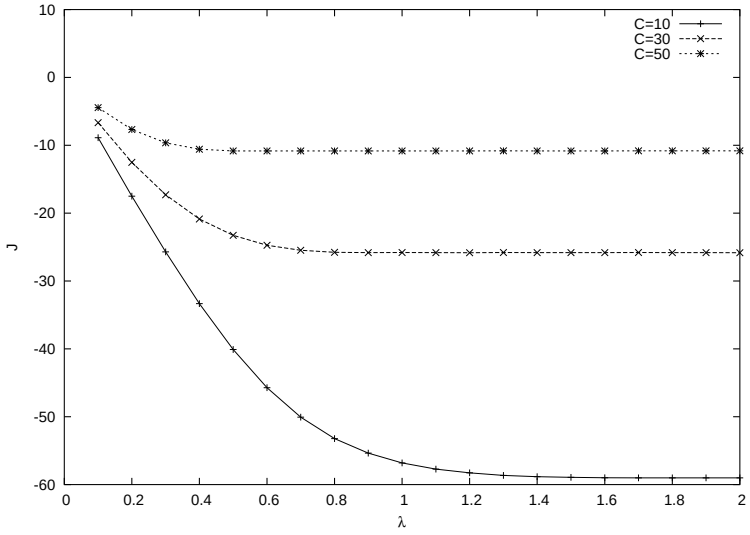


Figure 8: Average optimal cost versus arrival rate when $\tau = 5.0$.

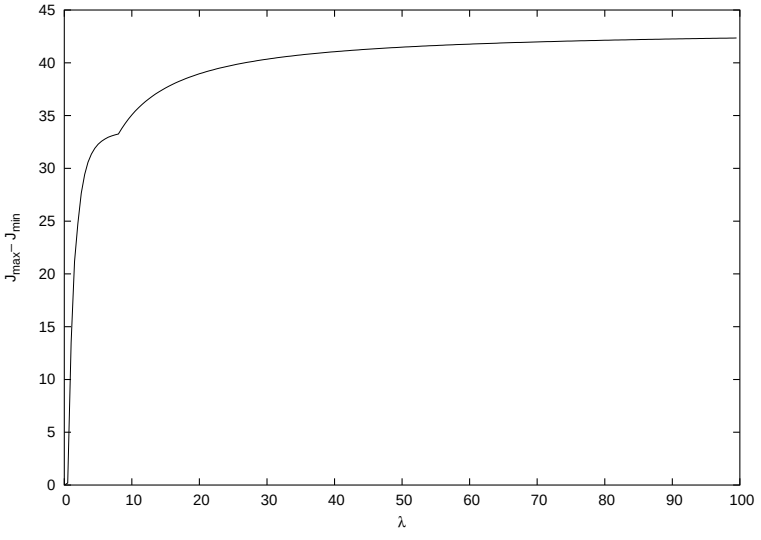


Figure 9: The value of information versus arrival rate

approach. The optimal solutions for the open and the closed loop control models naturally form the upper and the lower bound of the optimal cost for a model with an arbitrary observation interval. We also obtain a tighter lower bound by limit ting the arrival rate to be less than twice the departure rate and the reward coefficient is less than one half of the waiting cost coefficient. The value of information possesses a limit as the arrival rate increases. With help of value iteration algorithm and induction, we proved that the average optimal policy is nonincreasing. Several numerical examples are also provided in a separate section.

Many claims/arguments in the paper are not specific to the M/M/1 queue, e.g. Lemma 2 used in the proof of the existence of average optimal policy; Lemma 11 and 12 in the proof of the monotonic property of optimal policy. However one must proceed with caution when the approach in defining the discrete time Markov decision process for the M/M/1 model in this paper is extended to an M/G/1 or an G/M/1 queue.

A The Transformation Expressions for an M/M/1 Queue and an M/M/1/K Queue

Let $P(i,t)$be the probability that there are i customers in an M/M/1 queueing system (arrival rate $\lambda > 0$ and service rate 1) at time t given that $P(n,0) = 1$. Let $P^*(z,s) := \sum_{i=0}^{\infty} z^i \int_0^{\infty} e^{-st} P(i,t)\, dt$. According to [16, p.77 Eq. 2.162]

$$P^*(z,s) = \frac{z^{n+1} - (1-z) P_0^*(s)}{sz - (1-z)(1-\lambda z)}$$

where $P_0^*(s) = \frac{\alpha^{n+1}}{1-\alpha}$ and $\alpha = \frac{s+1+\lambda-\sqrt{(s+1+\lambda)^2-4\lambda}}{2\lambda}$.

Let $\bar{N}(t,n,\lambda)$ be the average number of customers in the system at time t. Let

$$\bar{N}^*(s,n,\lambda) := \int_0^{\infty} e^{-st} \bar{N}(t,n,\lambda)\, dt.$$

Clearly we have

$$\begin{aligned} \bar{N}^*(s,n,\lambda) &= \lim_{z\to 0} \frac{d}{dz} P^*(z,s) \\ &= \frac{sn + \lambda - 1 + sP_0^*(s)}{s^2}. \end{aligned}$$

In the case of M/M/1/K queue the corresponding transformation expression is given as follows

$$P^*(z,s) = \frac{z^{n+1} - (1-z)P_0^*(s) + \lambda z^{K+1}(1-z)P_K^*(s)}{-\left[\lambda z^2 - s(s+\lambda+1)z + 1\right]},$$

$$\bar{N}(s,n,\lambda) = \frac{sn + \lambda - 1 + sP_0^*(s) - \lambda s P_K^*(s)}{s^2}$$

where

$$P_0^*(s) = \frac{-\left(\alpha_1^{n+1}\alpha_2^{K+1}(1-\alpha_2) - \alpha_1^{K+1}\alpha_2^{n+1}(1-\alpha_1)\right)}{(1-\alpha_1)(1-\alpha_2)\left(\alpha_1^{K+1} - \alpha_2^{K+1}\right)}$$

$$P_K^*(s) = \frac{-\left(\alpha_1^{n+1}(1-\alpha_2) - \alpha_2^{n+1}(1-\alpha_1)\right)}{\lambda(1-\alpha_1)(1-\alpha_2)\left(\alpha_1^{K+1} - \alpha_2^{K+1}\right)}$$

and

$$\alpha_1 = \frac{s+1+\lambda+\sqrt{(s+1+\lambda)^2 - 4\lambda}}{2\lambda},$$

$$\alpha_2 = \frac{s+1+\lambda-\sqrt{(s+1+\lambda)^2 - 4\lambda}}{2\lambda}.$$

Many numerical inversion schemes can be employed in calculating $N(t,n,\lambda)$. We found that Crump's method [8] is particularly stable in our numerical calculations. When we are interested in the set of transient probability distribution function $\{P(i,t), i = 0, \cdots, K\}$ for the M/M/1/K queue, the matrix exponent method can be used to calculate the transition matrix: $P = [P(j,t \mid i,\lambda)] = \exp(Qt)$ where Q is the transition rate matrix. We uses the matrix exponent package called expokit [31] in our numerical investigations.

When $\lambda = 0$, the expression for $P(i,t)$ and $\bar{N}(t,n,\lambda)$ (both for the M/M/1 queue and the M/M/1/K queue) are straightforward,

$$P(i,t) = \begin{cases} 1 - e^{-t}\sum_{k=0}^{n-1}\frac{t^k}{k!} & \text{if } i = 0 \\ \frac{t^{i-k}}{(i-k)!}e^{-t} & \text{otherwise} \end{cases}$$

$$\bar{N}(t,n,0) = e^{-t}\sum_{k=1}^{n}\frac{k\,t^{n-k}}{(n-k)!}$$

B Theorem of Tauberian

The Tauberian theorem is the key in proving an existence of the average cost optimal policy using the vanishing discount approach. It can be found in [3], Heyman and Sobel [14, Proposition 4.7] and Sennott [27, Appendix A4]

Theorem 15. *Let* $\{a_n\}$*be a sequence of nonnegative numbers and* $\beta \in (0,1)$*, then*

$$\begin{aligned}
\liminf_{N\to\infty} \frac{1}{N}\sum_{m=0}^{N-1} a_m &\leq \liminf_{\beta\uparrow 1} (1-\beta)\sum_{n=0}^{\infty}\beta^n a_n \\
&\leq \limsup_{\beta\uparrow 1} (1-\beta)\sum_{n=0}^{\infty}\beta^n a_n \\
&\leq \limsup_{N\to\infty} \frac{1}{N}\sum_{m=0}^{N-1} a_m
\end{aligned}$$

C Supermodularity and Stochastic Orders

C.1 Supermodularity

Supermodularity has been proved itself a sharp tool in establishing the monotonic property of many functions. In particular, it is has been used in many occasions to prove the monotonic property of the optimal control policy on queueing problems, see [35] and references therein. Formal discussion of supermodularity and its application to Markov decision process problems can be found in [36] and Heyman and Sobel [14, Chapter 8, p.368]. The description below is adapted for the particular needs of this paper.

A function $g(i,a)$, $g : \mathbb{N}\times\mathbb{R} \mapsto \mathbb{R}$ is supermodular in (i,a), or simply supermodular, if $g(i,a) - g(i,b)$ is nondecreasing in i for all $a > b$. The following results are particular important for our problem.

Lemma 16. *If* $g(i,a)$*is supermodular in* (i,a)*, then* $\pi(i) = \inf\{\arg\min_a g(i,a)\}$ *is nonincreasing in* i*.*

Proof. Assume that $\pi(i) < \pi(i+1)$ for some i. Since $g(i,a)$ is supermodular in (i,a), we have

$$g(i,\pi(i+1)) - g(i,\pi(i)) \leq g(i+1,\pi(i+1)) - g(i+1,\pi(i)).$$

This implies that

$$g(i+1,\pi(i+1)) = g(i+1,\pi(i))$$

which contradicts the definition of $\pi(i+1)$. □

Lemma 17. *(a) If $f(i,a)$ and $g(i,a)$ are both supermodular in (i,a) then $f(i,a)+g(i,a)$ is supermodular in (i,a).*

(b) If a function $f(i,a,t)$, $f:\mathbb{N}\times\mathbb{R}\times\mathbb{R}\mapsto\mathbb{R}$, is supermodular in (i,a) for all $t\in\mathbb{R}$ and the integral $\int_A f(i,a,t)\,dt$, is properly defined over the set $A\subset\mathbb{R}$, then $\int_A f(i,a,t)\,dt$ is supermodular in (i,a).

C.2 Stochastic Orders

Let X and Y be two random variables such that $P(X>u)\leq P(Y>u)$ for all $u\in(-\infty,+\infty)$. Then X is said to be smaller than Y in the usual stochastic order (denoted by $X\leq_{\text{st}}Y$ or $Y\geq_{\text{st}}X$). More information on stochastic order can found in Ross [23, Chapter 8, p. 251],[29] and Chen and Yao [7, p. 8].

References

[1] Altman, E. (1999). *Constrained Markov Decision Processes*. Chapman & Hall/CRC.

[2] Altman, E. and Stidham Jr., S. (1995). Optimality of monotonic policies for two-action Markovian decision processes, with application to control of queues with delayed information. *Queueing Systems: Theory and Applications*, 21:267–291.

[3] Arapostathis, A., Borkar, V., Fernández-Gaucherand, E., Ghosh, M. K., and Marcus, S. I. (1993). Discrete-time controlled Markov processes with average cost criterion: a survey. *SIAM J. Control and Optimization*, 31(2):282–344.

[4] Blackwell, D. (1962). Discrete dynamic programming. *The Annals of Mathematical Statistics*, 33(2):719–726.

[5] Borkar, V. S. (1988). *Stochastic Differential Systems, Stochastic Control Theory and Applications*, volume 10 of *The IMA Volumes in Mathematics and Its Applications*, chapter Control of Markov chains with long-run average cost criterion, pages 57–77. Springer-Verlag.

[6] Borkar, V. S. (2000). *Markov Decision Processes: Model, Methods and Open Problems*, chapter Convex analytic methods in Markov decision processes. Kluwer 2000.

[7] Chen, H. and Yao, D. D. (2001). *Fundamentals of Queueing Networks: Performance, Asymptotics and Optimization*. Springer.

[8] Crump, K. S. (1976). Numerical inversion of laplace transforms using a fourier series approximation. *Journal of the ACM*, 23(1):89–96.

[9] Derman, C. (1966). Denumerable state Markov decision processes, average cost criterion. *Ann. Math. Statist.*, 37:1545–1553.

[10] Fox, B. L. (1971). Finite-State Approximations to Denumerable-State Dynamic Programs. *J. Math. Anal. Appl*, 34(665–670).

[11] Fukuda, A. (1986). Input regulation control based on periodical monitoring using call gapping control. *Electronics Comm. Japan, Part 1*, 69(11):84–92.

[12] Hassin, R. and Haviv, M. (2003). *To Queue or Not to Queue: Equilibrium Behavior in Queueing Systems*. Kluwer Academic Publishers.

[13] Hernández-Lerma, O. and Lasserre, J. B. (1996). *Discrete-Time Markov Control Processes, Basic Optimality Criteria*. Springer.

[14] Heyman, D. P. and Sobel, M. J. (2004). *Stochastic Model in Operations Research Vol II Stochastic Optimization*. Dover. McGraw-Hill Book Company 1984.

[15] Hordijk, A. (1974). Dynamic Programming and Markov Potential Theory. Technical report, Math. Centre Tract No. 51, Mathematisch Centrum, Amsterdam.

[16] Kleinrock, L. (1975). *Queueing Systems: Volume 1: Theory*. John Wiley & Sons.

[17] Kuri, J. and Kumar, A. (1995). Optimal control of arrivals to queues with delayed queue length information. *IEEE Tran. Automatic Control*, 40(8):1444–1450.

[18] Lin, K. Y. and Ross, S. M. (2003). Admission control with incomplete information of a queueing system. *Operations Research*, 51(4):645–654.

[19] Naor, P. (1969). The regulation of queue size by levying tolls. *Econometrica*, 37(1).

[20] Puterman, M. L. (1994). *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

[21] Robertsson, A., Wittenmark, B., and Kihl, M. (2003). Analysis and design of admission control in Web-server systems. In *American Control Conference 2003*.

[22] Ross, S. M. (1982a). *Introduction to Stochastic Dynamic Programming*. Academic Press.

[23] Ross, S. M. (1982b). *Stochastic Processes*. John Wiley & Sons.

[24] Ross, S. M. (1992). *Applied Probability Models with Optimization Applications*. Dover Pubns. Reprint edition of Holden-Day Inc. 1970.

[25] Rudin, W. (1986). *Real and Complex Analysis*. McGraw-Hill.

[26] Sennott, L. I. (1989). Average cost optimal stationary policies in infinite state Markov decision processes with unbounded costs. *Operations Research*, 37:626–633.

[27] Sennott, L. I. (1998). *Stochastic Dynamic Programming and the Control of Queueing Systems*. John Wiley & Sons.

[28] Sennott, L. I. (2000). *Markov Decision Processes: Model, Methods and Open Problems*, chapter Average reward optimization theory for denumerable state spaces. Kluwer 2000.

[29] Shaked, M. and Shanthikumar, J. G. (1994). *Stochastic Orders and Their Applications*. Academic Press.

[30] Shiryaev, A. N. (1996). *Probability*. Springer Verlag 2nd.

[31] Sidje, R. B. (1998). Expokit: software package for computing matrix exponentials. *ACM Transaction on Mathematical Software*, 24(1):130–156.

[32] Stidham Jr., S. (1985). Optimal control of admission to a queueing system. *IEEE Tran. on Automatic Control*, 30(8).

[33] Stidham Jr., S. (1988). *Stochastic differential systems: stochastic control theory and applications*, chapter Scheduling, Routing, and Flow Control in Stochastic Networks, pages 529–561. Springer-Verlag.

[34] Stidham Jr., S. (2002). Analysis, design and control of queueing systems. *Operations Research*, 50(1):197–216.

[35] Stidham Jr., S. and Weber, R. (1993). A survey of Markov decision models for control of networks of queues. *Queueing Systems*, 13:291–314.

[36] Topkis, D. M. (1978). Minimizing a submodular function on a Lattice. *Operations Research*, 26(2):305–321.

[37] Walrand, J. (1988). *An Introduction to Queueing Networks*. Prentice-Hall Inc.

[38] Whitt, W. (1978). Approximations of dynamic programs I. *Mathematics of Operations Research*, 3(3):231–243.

[39] Whitt, W. (1979). Approximations of dynamic programs, II. *Mathematics of Operations Research*, 4(2):179–185.

Reports on Communication Systems

101. **On Overload Control of SPC-systems**
Ulf Körner, Bengt Wallström, and Christian Nyberg, 1989.
CODEN: LUTEDX/TETS- -7133- -SE+80P

102. **Two Short Papers on Overload Control of Switching Nodes**
Christian Nyberg, Ulf Körner, and Bengt Wallström, 1990.
ISRN LUTEDX/TETS- -1010- -SE+32P

103. **Priorities in Circuit Switched Networks**
Åke Arvidsson, Ph.D. thesis, 1990.
ISRN LUTEDX/TETS- -1011- -SE+282P

104. **Estimations of Software Fault Content for Telecommunication Systems**
Bo Lennselius, Lic. thesis, 1990.
ISRN LUTEDX/TETS- -1012- -SE+76P

105. **Reusability of Software in Telecommunication Systems**
Anders Sixtensson, Lic. thesis, 1990.
ISRN LUTEDX/TETS- -1013- -SE+90P

106. **Software Reliability and Performance Modelling for Telecommunication Systems**
Claes Wohlin, Ph.D. thesis, 1991.
ISRN LUTEDX/TETS- -1014- -SE+288P

107. **Service Protection and Overflow in Circuit Switched Networks**
Lars Reneby, Ph.D. thesis, 1991.
ISRN LUTEDX/TETS- -1015- -SE+200P

108. **Queueing Models of the Window Flow Control Mechanism**
Lars Falk, Lic. thesis, 1991.
ISRN LUTEDX/TETS- -1016- -SE+78P

109. **On Efficiency and Optimality in Overload Control of SPC Systems**
Tobias Rydén, Lic. thesis, 1991.
ISRN LUTEDX/TETS- -1017- -SE+48P

110. **Enhancements of Communication Resources**
Johan M Karlsson, Ph.D. thesis, 1992.
ISRN LUTEDX/TETS- -1018- -SE+132P

111. **On Overload Control in Telecommunication Systems**
Christian Nyberg, Ph.D. thesis, 1992.
ISRN LUTEDX/TETS- -1019- -SE+140P

112. **Black Box Specification Language for Software Systems**
Henrik Cosmo, Lic. thesis, 1994.
ISRN LUTEDX/TETS- -1020- -SE+104P

113. **Queueing Models of Window Flow Control and DQDB Analysis**
Lars Falk, Ph.D. thesis, 1995.
ISRN LUTEDX/TETS- -1021- -SE+145P

114. **End to End Transport Protocols over ATM**
Thomas Holmström, Lic. thesis, 1995.
ISRN LUTEDX/TETS- -1022- -SE+76P

115. **An Efficient Analysis of Service Interactions in Telecommunications**
Kristoffer Kimbler, Lic. thesis, 1995.
ISRN LUTEDX/TETS- -1023- -SE+90P

116. **Usage Specifications for Certification of Software Reliability**
Per Runeson, Lic. thesis, May 1996.
ISRN LUTEDX/TETS- -1024- -SE+136P

117. **Achieving an Early Software Reliability Estimate**
Anders Wesslén, Lic. thesis, May 1996.
ISRN LUTEDX/TETS- -1025- -SE+142P

118. **On Overload Control in Intelligent Networks**
Maria Kihl, Lic. thesis, June 1996.
ISRN LUTEDX/TETS- -1026- -SE+80P

119. **Overload Control in Distributed-Memory Systems**
Ulf Ahlfors, Lic. thesis, June 1996.
ISRN LUTEDX/TETS- -1027- -SE+120P

120. **Hierarchical Use Case Modelling for Requirements Engineering**
Björn Regnell, Lic. thesis, September 1996.
ISRN LUTEDX/TETS- -1028- -SE+178P

121. **Performance Analysis and Optimization via Simulation**
Anders Svensson, Ph.D. thesis, September 1996.
ISRN LUTEDX/TETS- -1029- -SE+96P

122. **On Network Oriented Overload Control in Intelligent Networks**
Lars Angelin, Lic. thesis, October 1996.
ISRN LUTEDX/TETS- -1030- -SE+130P

123. **Network Oriented Load Control in Intelligent Networks Based on Optimal Decisions**
Stefan Pettersson, Lic. thesis, October 1996.
ISRN LUTEDX/TETS- -1031- -SE+128P

124. **Impact Analysis in Software Process Improvement**
Martin Höst, Lic. thesis, December 1996.
ISRN LUTEDX/TETS- -1032- -SE+140P

125. **Towards Local Certifiability in Software Design**
Peter Molin, Lic. thesis, February 1997.
ISRN LUTEDX/TETS- -1033- -SE+132P

126. **Models for Estimation of Software Faults and Failures in Inspection and Test**
Per Runeson, Ph.D. thesis, January 1998.
ISRN LUTEDX/TETS- -1034- -SE+222P

127. **Reactive Congestion Control in ATM Networks**
Per Johansson, Lic. thesis, January 1998.
ISRN LUTEDX/TETS- -1035- -SE+138P

128. **Switch Performance and Mobility Aspects in ATM Networks**
Daniel Søbirk, Lic. thesis, June 1998.
ISRN LUTEDX/TETS- -1036- -SE+91P

129. **VPC Management in ATM Networks**
Sven-Olof Larsson, Lic. thesis, June 1998.
ISRN LUTEDX/TETS- -1037- -SE+65P

130. **On TCP/IP Traffic Modeling**
Pär Karlsson, Lic. thesis, February 1999.
ISRN LUTEDX/TETS- -1038- -SE+94P

131. **Overload Control Strategies for Distributed Communication Networks**
Maria Kihl, Ph.D. thesis, March 1999.
ISRN LUTEDX/TETS- -1039- -SE+158P

132. **Requirements Engineering with Use Cases - a Basis for Software Development**
Björn Regnell, Ph.D. thesis, April 1999.
ISRN LUTEDX/TETS- -1040- -SE+225P

133. **Utilisation of Historical Data for Controlling and Improving Software Development**
Magnus C. Ohlsson, Lic. thesis, May 1999.
ISRN LUTEDX/TETS- -1041- -SE+146P

134. **Early Evaluation of Software Process Change Proposals**
Martin Höst, Ph.D. thesis, June 1999.
ISRN LUTEDX/TETS- -1042- -SE+193P

135. **Improving Software Quality through Understanding and Early Estimations**
Anders Wesslén, Ph.D. thesis, June 1999.
ISRN LUTEDX/TETS- -1043- -SE+242P

136. **Performance Analysis of Bluetooth**
Niklas Johansson, Lic. thesis, March 2000.
ISRN LUTEDX/TETS- -1044- -SE+76P

137. **Controlling Software Quality through Inspections and Fault Content Estimations**
Thomas Thelin, Lic. thesis, May 2000
ISRN LUTEDX/TETS- -1045- -SE+146P

138. **On Fault Content Estimations Applied to Software Inspections and Testing**
Håkan Petersson, Lic. thesis, May 2000.
ISRN LUTEDX/TETS- -1046- -SE+144P

139. **Modeling and Evaluation of Internet Applications**
Ajit K. Jena, Lic. thesis, June 2000.
ISRN LUTEDX/TETS- -1047- -SE+121P

140. **Dynamic traffic Control in Multiservice Networks - Applications of Decision Models**
Ulf Ahlfors, Ph.D. thesis, October 2000.
ISRN LUTEDX/TETS- -1048- -SE+183P

141. **ATM Networks Performance - Charging and Wireless Protocols**
Torgny Holmberg, Lic. thesis, October 2000.
ISRN LUTEDX/TETS- -1049- -SE+104P

142. **Improving Product Quality through Effective Validation Methods**
Tomas Berling, Lic. thesis, December 2000.
ISRN LUTEDX/TETS- -1050- -SE+136P

143. **Controlling Fault-Prone Components for Software Evolution**
Magnus C. Ohlsson, Ph.D. thesis, June 2001.
ISRN LUTEDX/TETS- -1051- -SE+218P

144. **Performance of Distributed Information Systems**
Niklas Widell, Lic. thesis, February 2002.
ISRN LUTEDX/TETS- -1052- -SE+78P

145. **Quality Improvement in Software Platform Development**
Enrico Johansson, Lic. thesis, April 2002.
ISRN LUTEDX/TETS- -1053- -SE+112P

146. **Elicitation and Management of User Requirements in Market-Driven Software Development**
Johan Natt och Dag, Lic. thesis, June 2002.
ISRN LUTEDX/TETS- -1054- -SE+158P

147. **Supporting Software Inspections through Fault Content Estimation and Effectiveness Analysis**
Håkan Petersson, Ph.D. thesis, September 2002.
ISRN LUTEDX/TETS- -1055- -SE+237P

148. **Empirical Evaluations of Usage-Based Reading and Fault Content Estimation for Software Inspections**
Thomas Thelin, Ph.D. thesis, September 2002.
ISRN LUTEDX/TETS- -1056- -SE+210P

149. **Software Information Management in Requirements and Test Documentation**
Thomas Olsson, Lic. thesis, October 2002.
ISRN LUTEDX/TETS- -1057- -SE+122P

150. **Increasing Involvement and Acceptance in Software Process Improvement**
Daniel Karlström, Lic. thesis, November 2002.
ISRN LUTEDX/TETS- -1058- -SE+125P

151. **Changes to Processes and Architectures; Suggested, Implemented and Analyzed from a Project viewpoint**
Josef Nedstam, Lic. thesis, November 2002.
ISRN LUTEDX/TETS- -1059- -SE+124P

152. **Resource Management in Cellular Networks -Handover Prioritization and Load Balancing Procedures**
Roland Zander, Lic. thesis, March 2003.
ISRN LUTEDX/TETS- -1060- -SE+120P

153. **On Optimisation of Fair and Robust Backbone Networks**
Pål Nilsson, Lic. thesis, October 2003.
ISRN LUTEDX/TETS- -1061- -SE+116P

154. **Exploring the Software Verification and Validation Process with Focus on Efficient Fault Detection**
Carina Andersson, Lic. thesis, November 2003.
ISRN LUTEDX/TETS- -1062- -SE+134P

155. **Improving Requirements Selection Quality in Market-Driven Software Development**
Lena Karlsson, Lic. thesis, November 2003.
ISRN LUTEDX/TETS- -1063- -SE+132P

156. **Fair Scheduling and Resource Allocation in Packet Based Radio Access Networks**
Torgny Holmberg, Ph.D. thesis, November 2003.
ISRN LUTEDX/TETS- -1064- -SE+187P

157. **Increasing Product Quality by Verification and Validation Improvements in an Industrial Setting**
Tomas Berling, Ph.D. thesis, December 2003.
ISRN LUTEDX/TETS- -1065- -SE+208P

158. **Some Topics in Web Performance Analysis**
Jianhua Cao, Lic. thesis, June 2004
ISRN LUTEDX/TETS- -1066- -SE+99P